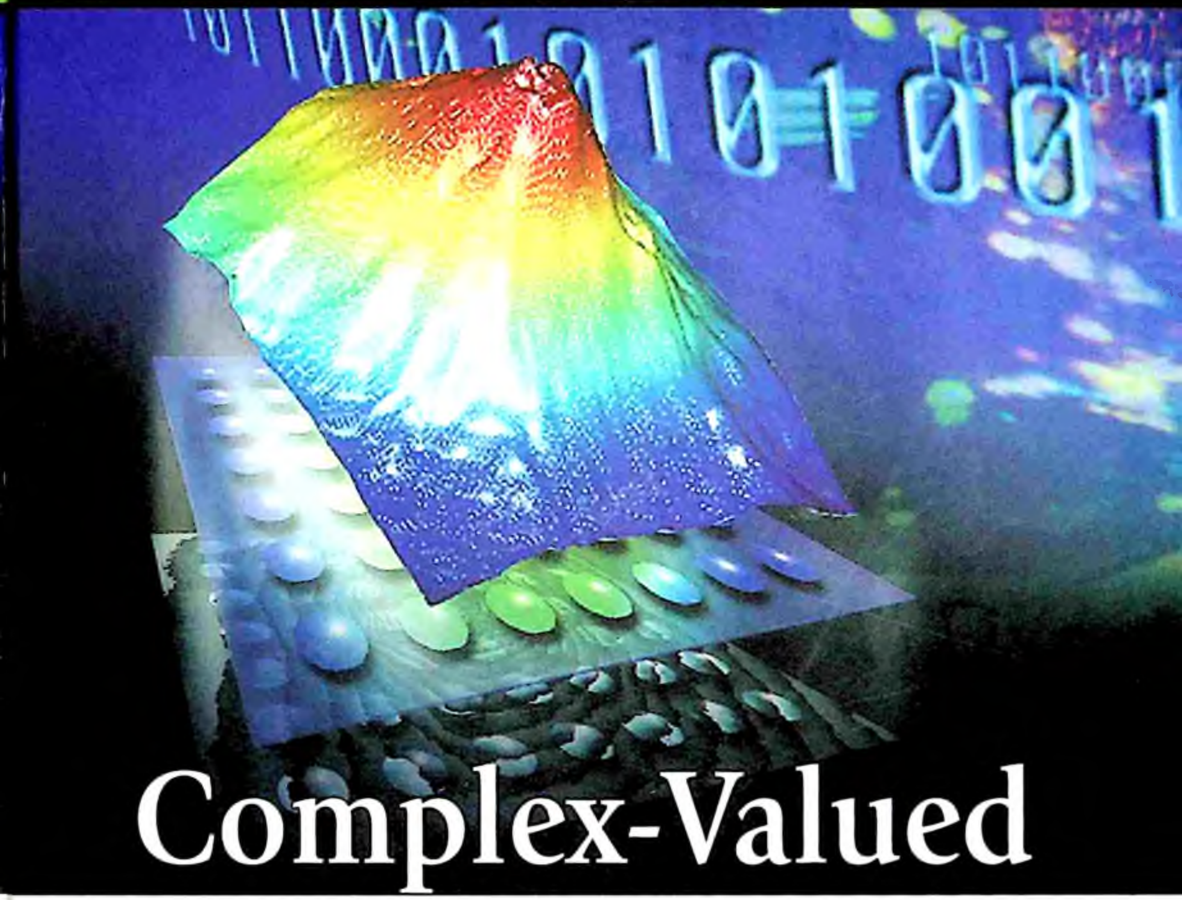
Series on Innovative Intelligence – Vol. 5



Complex-Valued Neural Networks

Theories and Applications

Editor

Akira Hirose

Complex-Valued Neural Networks

Theories and Applications

Series on Innovative Intelligence – Vol. 5

Complex-Valued Neural Networks

Theories and Applications

Editor

Akira Hirose

The University of Tokyo, Japan

World Scientific

NEW JERSEY • LONDON • SINGAPORE • SHANGHAI • HONG KONG • TAIPEI • BANGALORE

Published by

World Scientific Publishing Co. Pte. Ltd.
5 Toh Tuck Link, Singapore 596224
USA office: Suite 202, 1060 Main Street, River Edge, NJ 07661
UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

Cover Illustration: From Fig. 10 (b3) in Chapter 13, p. 298.

COMPLEX-VALUED NEURAL NETWORKS
Theories and Applications

ISBN 981-238-464-2
ISBN 981-238-561-4 (pbk)

Printed in Singapore by World Scientific Printers (S) Pte Ltd

Foreword

The history of complex numbers shows that although slow to be accepted, they have become quite effective in and are an integral part of many areas of engineering and science. The present volume advances applications, theories, and trends in the promising field of complex-valued neural networks.

In their early history complex numbers were despised, deprecated and something to be avoided. Mathematicians looked upon them with suspicion. Gauss gave them new respectability by giving them his seal of approval with the publication of a memoir to the Royal Society of Göttingen in 1831. By the passage of time, complex numbers have become a main staple in mathematics, engineering, and science (Nahin 1998).

There are basically two questions that can be conceivably raised about complex numbers: the first being are they necessary? Can we just live without them? From a mathematical point of view, their usefulness and necessity is enshrined in the Fundamental Theorem of Algebra: *Every polynomial of degree n with complex coefficients (or real as a special case) has n roots in the complex numbers.* Some of those roots may be real, but some of them may have imaginary parts. Thus, the polynomial can be factored into exactly n linear terms. This gives complex numbers a privileged status in mathematics. Over the last century, in other fields which make heavy use of mathematics, such as engineering and science, complex numbers have triumphed. From AC circuit analysis, to control theory, to electromagnetics, to optics, to quantum mechanics, complex numbers are omni-present and ever-used. To avoid them, it can be claimed that one would have to reinvent them or something akin to them. Consider for example quantum mechanics, which is done invariably in the complex domain (Feynman *et al.* 1965). It is possible to be done in the real domain (Stueckelberg 1960), but even in that case an anti-symmetric operator is introduced to basically serve the role of $\sqrt{-1}$.

The second question is "do they represent anything meaningful?" A similar objection can be made, and was made, for negative numbers. After all, who possessed -2 cats? Complex numbers have shown an uncanny ability to model physical quantities, from magnitude and phase of current to probability amplitudes in quantum mechanics. In many aspects, they mirror nature in the way they behave, e.g. in addition and multiplication.

The history of complex neural networks can be traced to (Widrow *et al.* 1975) where the complex LMS algorithm was formulated, and was later widely used in filtering. Slowly in the beginning but increasingly so, complex-valued neural networks are employed to generalize their real domain counterparts and to handle applications that traditionally have been handled in the complex domain. Generalizations are not simply done by changing of a variable, from the real one to the complex one. If done in this way, singularities and other such unpleasant phenomena may arise (Georgiou and Koutsougeras 1992). Neural networks have to be adjusted to the rich structure and environment of the complex domain if they are to operate properly and take advantage of its power.

Complex-valued neural networks have encompassed a wide range of theories and applications: from quantum neural networks to optoelectronics to satellite imaging to communications. The long-awaited present volume, under the able care of Professor Akira Hirose, a well-known pioneer in the field, highlights and brings into focus the rich range of applications, theories, and trends. The diverse material in the chapters that follow, if available, was scattered in literature. Now it is accessible in a single volume, the first in complex-valued neural networks research, which is destined to become a standard manual and reference. Considering the recent research activity and using history as a guide, it is safe to predict that the field will flourish.

References

Georgiou, G.M. and Koutsougeras, C. (1992), "Complex Domain Backpropagation," *IEEE Transactions on Circuits and Systems. II: Analog and Digital Signal Processing*, vol. 39, no. 5, pp. 330–334.

Feynman, R.P., Leighton, R.B. and Sands M. (1965), *The Feynman Lectures on Physics*, vol. 3, Addison-Wesley, Reading Massachusetts.

Nahin, P.J. (1998), *An Imaginary Tale : The Story of* $\sqrt{-1}$, Princeton University Press, Princeton, New Jersey.

Stueckelberg, C.G. (1960), "Quantum theory in real Hilbert space," *Helvetica Physica Acta*, vol. 33, pp. 727-752.

Widrow, B., McCool, J. and Ball, M. (1975), "The complex LMS algorithm," *Proc. IEEE*, pp. 719–720.

Professor George M. Georgiou
Computer Science Department, California State University,
San Bernardino, CA 92407-2397, USA
Email: georgiou@csusb.edu

Preface

In recent years, the complex-valued neural networks have been extending the scope of application in optoelectronics, imaging, remote sensing, quantum neural devices and systems, spatiotemporal analysis of physiological neural systems, and artificial neural information processing. In this first-ever book on the complex-valued networks, the most active scientists at the forefront of the field describe theories and applications from various points of view.

This book provides academic and industrial researchers with a comprehensive understanding of the fundamentals, features and prospects of the powerful complex-valued neural networks. It is also suitable for introducing graduate students to one of the most exciting and exploding fields.

The Chapters are arranged so that theoretical works roughly precede application-oriented ones. However, an application often yields a specific theory besides common complex-valued neural network framework. The fact results in the variety and the plentifulness of this widening area.

Chapter 1 *Complex-Valued Neural Networks: An Introduction* provides a short description on features, applications and a perspective of the field in general. **Chapter 2** *Orthogonal Decision Boundaries and Generalization of Complex-Valued Neural Networks* gives one of the typical complex-valued neural network models and discusses the characteristics of decision boundaries of a single, networked or layered neurons. **Chapter 3** *Complex-Valued Neural Associative Memories: Network Stability and Learning Algorithm* treats the dynamics of associative memories based on energy and compares learning rules. **Chapter 4** *A Model of Complex-Valued Associative Memories and Its Dynamics* presents another viewpoint to the associative memory from which we can regard a set of degenerate vectors as a pattern embedded in the memory. **Chapter 5** *Clifford Networks* a further extension of the complex-valued networks, introduces the Clifford algebras and proposes and analyzes a Clifford-based back-propagation learning rule.

Chapter 6 *Complex Associative Memory and Complex Single Neuron Model* deals with a complex-valued Nagumo-Sato neuron model in the complex space and presents its chaotic behavior. **Chapter 7** *Data-Reusing Algorithm for Complex-Valued Adaptive Filters* analyzes a class of data-reusing learning algorithms for complex-valued adaptive filters and applies an improved algorithm to signal prediction. **Chapter 8** *Instantaneously Trained Neural Networks with Complex Inputs* proposes the 3C algorithm which realizes a time-efficient and resource saving learning in combination with the quaternary encoding technique.

Chapter 9 *Applications of Complex-Valued Neural Networks for Image Processing* describes an image retrieval system where gray-scale images are spatially Fourier transformed and mapped consistently on the unit circle on the complex plane. **Chapter 10** *Memorization of Melodies Using Complex-Valued Recurrent Neural Network* presents a efficient time-sequential data identifier and adaptive generator, named MUSIC, which utilizes the high dynamics stability of recurrent complex-valued networks, and applies to music recall.

Chapter 11 *Complex-Valued Generalized Hebbian Algorithm and Its Applications to Sensor Array Signal Processing* describes an application of Hebbian rule in complex domain to direction-of-arrival problem of sensor arrays. **Chapter 12** *Phasor Model with Application to Multiuser Communication* introduces an associative memory that has a zero resting attractor for application to the multiuser detection in code-division multiple-access communications by realizing active and inactive modes. **Chapter 13** *Adaptive Interferometric Radar Image Processing by Using Complex-Valued Neural Network* presents a lattice neural network for automatic generation of digital elevation map by reducing the number of phase singular points in interferometric synthetic-aperture-radar images.

Chapter 14 *Complex Neural Network Model with Analogy to Self-Oscillation Generated in an Optical Phase-Conjugate Resonator* points out the analogy between the Hopfield network and self-oscillating phase-conjugate resonator and reports experiments using a $BaTiO_3$ crystal and an Argon-ion laser oscillating at 514.5nm. **Chapter 15** *Coherent Lightwave Neural Networks: Use of Frequency Domain* reports a coherent lightwave neural network system whose learning and processing behavior is controllable by using its optical carrier frequency as a modulation key.

I express sincere gratitude to Professor Lakhmi Jain at the University of South Australia. He gave me this wonderful opportunity to collect the first fruits of this field. I am also deeply grateful to Professor Norio Baba at the Osaka Kyoiku University. He was the General Chair of the International Conference on Knowledge-based Intelligent Information and Engineering Systems (KES) 2001 Osaka where not a few researchers in the field gathered for the first time to discuss the complex-valued neural networks in a Special Session. Up to that point, the relevant researchers were connected only by point to point. But after this Special Session, we have got networked to extend the theories and applications and to involve many researchers in neighboring and unexpected but inspiring fields.

I also express my thanks to Dr. K.K.Phua, Chairman, and Ms. Lakshmi Narayan, Senior Editor, and Mr. Loo Chuan Ming of Art Department at World Scientific Publishing Co. for their patience and kind cooperation given for this enterprising publication. With their kind understanding of the significance of this new field, we are fortunate to be able to accelerate our extensive research.

Tokyo, June 2003 Akira Hirose

Contents

Chapter 1

Complex-Valued Neural Networks: An Introduction

Akira Hirose

Complex-valued neural networks deal with complex-valued data with complex-number weights and complex-valued neuron-activation functions. George M. Gerogiou describes clearly in the Foreword the necessity of the complex-valued networks. In this introductory short chapter, we discuss how they are or can be useful and effective. We begin with the role of $i \equiv \sqrt{-1}$ in the quantum mechanics.

According to the quantum mechanics, the motion of an electron is related to the Schrödinger equation:

$$i\hbar \frac{\partial \Psi(\boldsymbol{r}, t)}{\partial t} = -\frac{\hbar^2}{2m} \nabla^2 \Psi(\boldsymbol{r}, t) + V(\boldsymbol{r}) \Psi(\boldsymbol{r}, t) \tag{1}$$

where $\Psi(\boldsymbol{r}, t)$ is the electron's wave function in terms of position $\boldsymbol{r}$ and time t, and $\hbar$, m, $V(\boldsymbol{r}, t)$ and ∇ denote Plank constant divided by 2π, electron mass, potential function and spatial differential operator, respectively.

The probabilistic interpretation argues that the squared absolute value of the solution $|\Psi|^2$ is the probability density of electron existence. The probability is related to ensemble average. However, realistically, through repetitive or long-term experiment of electron observation in an ergodic condition, we find that the electrons obey the probability $|\Psi|^2$. The equation represents experimental results successfully.

The special feature of this equation lies in the fact that it contains the imaginary unit i in an ineliminable manner. Some physicists claim that fundamental equations in physics should not include i because they should consist of only really physical entities. Even if we regard $|\Psi|^2$ as a probability, probabilities in general are discussed by using real number, and again i is not desirable.

However, we can consider the equation as follows. "The probability possesses an amplitude entity $\log(A)$ and a phase entity ϕ. Its spatial and/or temporal evolution obeys

$$\Psi_k(\boldsymbol{r}, t) = e^{\log(A_k(\boldsymbol{r},t))+i\phi_k(\boldsymbol{r},t)} \tag{2}$$

and the principle of superposition holds for plural solutions Ψ_k as

$$\Psi(\boldsymbol{r}, t) = \sum_k \Psi_k = \sum_k e^{\log(A_k(\boldsymbol{r},t))+i\phi_k(\boldsymbol{r},t)} \tag{3}$$

That is to say, the imaginary unit i plays a role in the interaction of amplitude and phase entities as well as the superposition of probability functions." In this sense, i is an operator to connect entities rather than an existence itself. In the complex-valued neural networks, i has the same role to combine plural quantities consistently.

Then, what is the operation of i? In the networks, we multiply input data by synaptic weights. If we pay attention to real and imaginary parts of the data, the multiplication of i, for example, converts the real quantity into imaginary one, and also does the imaginary one into real one with putting a negative sign. In this way the two quantities are exchanged. Multiplication of general complex-valued weight w mixes the real and imaginary quantities in a certain manner determined by the value w. In the network, the muptiplication of w is executed for all the parallel input data elements $\boldsymbol{x} = [x_k]$. As a result, the output maintains a certain vector-direction relation in the complex plane. This is one of the properties that we can utilize effectively to treat two-dimensional information.

If we have a polar coordinate picture, we can regard the multiplication of $w = e^{\log|w| + i \arg(w)}$ as the magnification of the vector length by $|w|$ and the vector rotation of an angle $\arg(w)$. In the problems where the weight has the amplitude and phase entities $|w|$ and $\arg(w)$ in the real world, just like the solutions of Schrödinger equation, the neural operation can directly influence these entities rather than some other apparent phenomena. Actually, (quasi-)periodic signals can be processed in relation to phase because they are expressed as an integration of sinusoidal wave through the Fourier transform and the Fourier synthesis.

Furthermore, we know in the phasor treatment of signals with carrier wave that a temporal differentiation realized by a capacitor C, for example, is analytically equivalent to multiplication of $i\omega C$ where ω is angular frequency, while an integration is to division by $i\omega C$. Such relations are utilized in stabilization of dynamical or time-sequential behavior of neural networks. The phase topology is also related directly to a cyclic metric.

These properties are very important also in the device electronics. The amplitude and phase of an electron probability function is expressed by (1) and modulated by electrical potential, permittivity, magnetic field, and so on. Lightwave and electromagnetic wave also have a similar wave nature. The probability density of a photon is given by the squared absolute value of the wave function. The value multiplied by the energy of a single photon gives the energy of lightwave. It can be modulated by absorption and amplification of media in reality. On the other hand, the phase corresponds to time delay or advance and is modulated by permittivity, permeability and optical path length. In this way, the fundamental particles composing the world interact each other through the amplitude (energy) and the phase (time). (In a special case such as resonance, these two entities have a clear relation, the Kramers-Kronig relation, where the neural holomorphy will become significant.) The complex-valued neural networks are highly expected to reflect such a natural world.

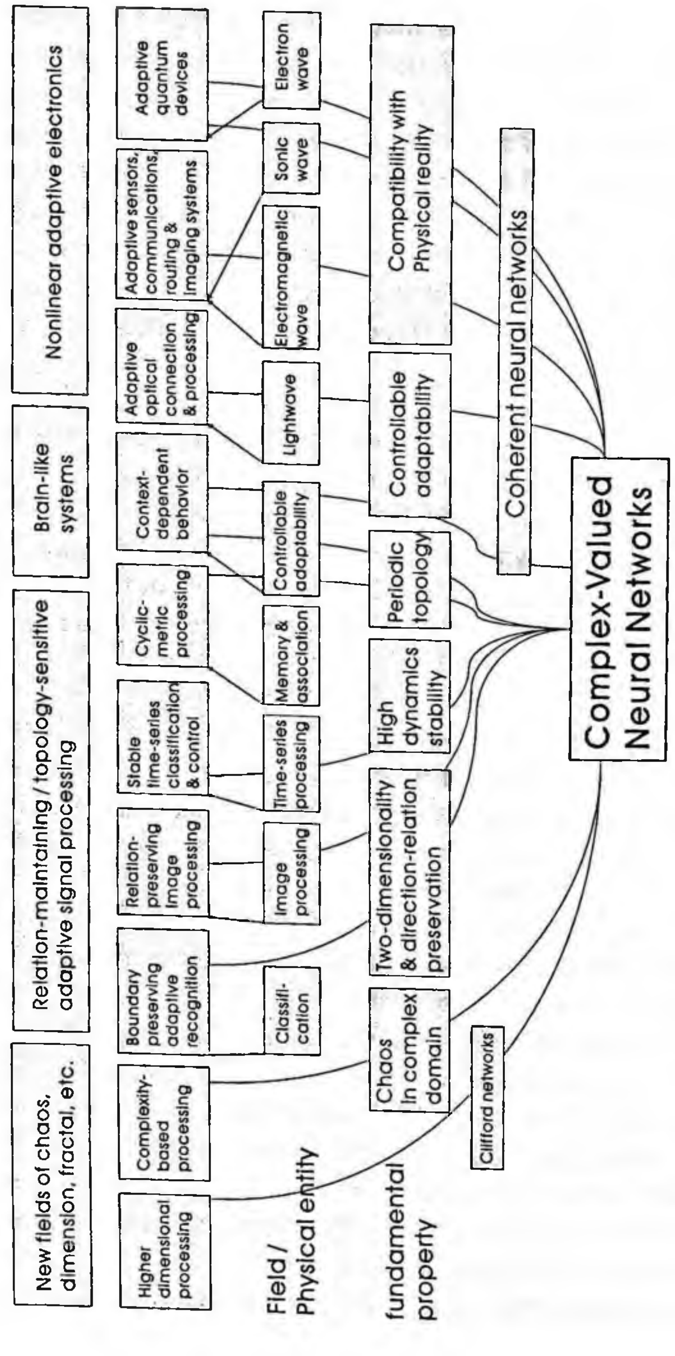

New fields of chaos, dimension, fractal, etc.
Relation-maintaining / topology-sensitive adaptive signal processing
Brain-like systems
Nonlinear adaptive electronics
Higher dimensional processing
Complexity-based processing
Boundary preserving adaptive recognition
Relation-preserving image processing
Stable time-series classification & control
Cyclic-metric processing
Context-dependent behavior
Adaptive optical connection & processing
Adaptive sensors, communications, routing & imaging systems
Adaptive quantum devices
Field / Physical entity
Classification
Image processing
Time-series processing
Memory & association
Controllable adaptability
Lightwave
Electromagnetic wave
Sonic wave
Electron wave
fundamental property
Chaos in complex domain
Two-dimensionality & direction-relation preservation
High dynamics stability
Periodic topology
Controllable adaptability
Compatibility with Physical reality
Clifford networks
Coherent neural networks
Complex-Valued Neural Networks

Figure 1. Fundamental properties and applications of complex-valued neural networks

Figure 1 presents the fundamental properties and application fields based on the discussion above. Most of the following chapters have a close relation to them. Recently the ideas and results have also been discussed in the Special Sessions in Conferences (KES 2001,2002) and (ICONIP 2002).

Further new ideas are also coming out to be presented in other Special Sessions in, for example, (ICANN / ICONIP 2003) and (KES 2003). The program includes quantum neural networks, radar imaging, array antenna signal processing, voice synthesis, spatiotemporal pattern processing, and so forth. These new results will be collected in a sequel. The complex-valued neural networks continue to extend the fields both in theories and applications.

References

Special Sessions on Complex-Valued Neural Networks, International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES),
KES 2001 Osaka (Sept. 6-8, 2001) Proc., N.Baba, L.C.Jain, R.J.Howlett, Eds., IOS Press, Ohmsha (2001) Part 1, pp. 550-580/ KES 2002 Crema (Sept. 16-18, 2002) Proc., E.Damiani, R.J.Howlett, L.C.Jain, N.Ichalkaranje, Eds., IOS Press, Ohmsha (2002) Part 1, pp. 623-647.

Special Session on Complex-Valued Neural Networks, International Conference on Neural Information Processing (ICONIP) 2002 Singapore (Nov. 18-22, 2002) Proc., Lipo Wang, et al., Eds., Vol.3, pp. 1074-1103.

Special Session on Complex-Valued Neural Networks: Theories and Applications, International Conference on Artificial Neural Networks (ICANN) 2003 Istanbul/ International Conference on Neural Information Processing (ICONIP) 2003 Istanbul (June 26-29, 2003) to be held.

Special Session on Complex-Valued Neural Networks, International Conference on Knowledge-Based Intelligent Information Engineering Systems (KES) 2003 Oxford (Sept. 3-5, 2003) to be held.

Author's address

Akira Hirose: Department of Electrical and Electronic Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
ahirose@ee.t.u-tokyo.ac.jp

Chapter 2

Orthogonal Decision Boundaries and Generalization of Complex-Valued Neural Networks

Tohru Nitta

This chapter presents some results of an analysis on the decision boundaries of the complex-valued neural networks whose weights, threshold values, input and output signals are all complex numbers. The main results can be summarized as follows. (a) Decision boundary of a single complex-valued neuron consists of two hypersurfaces which intersect orthogonally, and divides a decision region into four equal sections. Decision boundary of a three-layered complex-valued neural network has this as a basic structure, and its two hypersurfaces intersect orthogonally if net inputs to each hidden neuron are all sufficiently large. (b) Most of the decision boundaries in the 3-layered complex-valued neural network intersect orthogonally when the network is trained using the Complex-BP algorithm. As a result, the orthogonality of the decision boundaries improves its generalization ability. (c) Furthermore, the average of the learning speed of the Complex-BP is several times faster than that of the Real-BP. The standard deviation of the learning speed of the Complex-BP is smaller than that of the Real-BP. It seems that the complex-valued neural network and the related algorithm are natural for learning of complex-valued patterns for the above reasons.

1 Introduction

It is expected that complex-valued neural networks, whose parameters (weights and threshold values) are all complex numbers, will have applications in fields dealing with complex numbers such as telecommunications, speech recognition and image processing with the Fourier transformation. When using the existing method for real numbers, we must apply the method individually to their real and imaginary parts. On the other hand, complex-valued neural networks allow us to directly process data. Moreover complex-valued neural networks enable us to automatically capture good rotational behavior of complex numbers. For example, the fading equalization technology is an application domain suitable for the complex-valued neural network. Channel equalization in a digital communication system can be viewed as a pattern classification problem. The digital communication system receives a transmitted signal sequence with additive noise, and tries to estimate the true transmitted sequence. A transmitted signal can take one of the following four possible complex values: $-1 - i, -1 + i, 1 - i$ and $1 + i$ ($i = \sqrt{-1}$). Thus, the complex-valued neural network is suitable for this domain.

The back-propagation algorithm (called here, *Real-BP*) (Rumelhart *et al.* 1986) is an adaptive procedure which is widely used in training a multi-layer perceptron for a number of classification applications in areas such as speech and image recognition. The *Complex-BP* algorithm is a complex-valued version of the Real-BP, which was proposed by several researchers independently in the early 1990's (Kim and Guest 1990, Nitta and Furuya 1991, Benvenuto and Piazza 1992, Georgiou and Koutsougeras 1992, Nitta 1993, Nitta 1997). The Complex-BP algorithm can be applied to multi-layered neural networks whose weights, threshold values, input and output signals are all complex numbers. This algorithm enables the network to learn complex-valued patterns naturally, and has an *ability to transform geometric figures* as its inherent property, which may be related to the Identity Theorem in complex analysis (Nitta and Furuya 1991, Nitta 1993, Nitta 1997). Miyauchi et al. made an attempt to apply the Complex-BP (Nitta and Furuya 1991, Nitta 1993, Nitta 1997) in the computer vision field (Miyauchi *et al.* 1992, Miyauchi *et al.* 1993, Watanabe *et al.* 1994). They successfully used the ability to transform geometric figures of the Complex-BP network to complement the 2D velocity vector field on an image, which was derived from a set of images and called an *optical flow*. To specifically, the ability to transform geometric figures was applied to the estimation of optical flows. An optical flow is a 2D vector field indicating how an object moves. Generally, it is difficult to obtain a complete optical flow for a real image. Accordingly, to complement incomplete optical flow, the ability to transform geometric figures of the Complex-BP network was applied. They built a 1-n-1 complex-valued BP network, input a start point of a 2D vector, and got the network to learn to output its corresponding end point. Then, they input the coordinates of the missing part(s) of an optical flow to estimate its end point. The ability to transform geometric figures of the Complex-BP has the possibility of being widely used in other fields such as robot navigation and weather forecasting, because a 2D vector field appears in various actual scenes.

As we have seen above, the complex-valued neural network is useful and has lots of merits from the point of view of applications. It is important to clarify the characteristics of the complex-valued neural networks in order to promote the real applications.

This chapter makes clear the differences between the real-valued neural network and the complex-valued neural network by analyzing their fundamental properties from the view of network architectures, and clarifies the utility for the complex-valued neural network which the properties discovered in this chapter bring about. The main results can be summarized as follows. (a) Decision boundary of a single complex-valued neuron consists of two hypersurfaces which intersect orthogonally, and divides a decision region into four equal sections. Decision boundary of a three-layered complex-valued neural network has this as a basic structure, and its two hypersurfaces intersect orthogonally if net inputs to each hidden neuron are all sufficiently large. (b) Most of the decision boundaries in the 3-layered complex-valued neural network intersect orthogonally when the network is trained using the Complex-BP algorithm. As a result, the orthogonality of the decision boundaries improves its generalization ability. (c) Furthermore, the average of the learning speed of the Complex-BP is several times faster than that of the Real-BP. The standard deviation of the learning speed of the Complex-BP is smaller than that of the Real-BP. It seems that the complex-valued neural network is natural for learning of complex-valued patterns for the above reasons.

This chapter is organized as follows: Section 2 describes the complex-valued neural network and the related Complex-BP algorithm. Section 3 deals with the theoretical analyses of decision boundaries of the complex-valued neural network model. The simulation results are given in Section 4. Section 5 is devoted to the discussion on the results obtained in this chapter. Finally, we give some conclusions.

2 The Complex-Valued Neural Network

This section describes the complex-valued neural network used in the analysis. First, we will consider the following complex-valued neuron. The input signals, weights, thresholds and output signals are all complex numbers. The net input U_n to a complex-valued neuron n is defined as:

$$U_n = \sum_m W_{nm} X_m + V_n, \tag{1}$$

where W_{nm} is the (complex-valued) weight connecting complex-valued neurons n and m, X_m is the (complex-valued) input signal from complex-valued neuron m, and V_n is the (complex-valued) threshold value of neuron n. To obtain the (complex-valued) output signal, convert the net input U_n into its real and imaginary parts as follows: $U_n = x + iy = z$, where i denotes $\sqrt{-1}$. The (complex-valued) output signal is defined to be

$$f_C(z) = f_R(x) + i f_R(y), \tag{2}$$

where $f_R(u) = 1/(1 + \exp(-u)), u \in \boldsymbol{R}$ ($\boldsymbol{R}$ denotes the set of real numbers), that is, the real and imaginary parts of an output of a neuron mean the sigmoid functions of the real part x and imaginary part y of the net input z to the neuron, respectively. Note that f_C is not holomorphic, because the Cauchy-Riemann equations do not hold.

A complex-valued neural network consists of such complex-valued neurons described above. The Complex-BP learning rule (Nitta and Furuya 1991, Nitta 1993, Nitta 1997) has been obtained by using a steepest descent method for such (multi-layered) complex-valued neural networks.

3 Orthogonality of Decision Boundaries in the Complex-Valued Neural Network

Decision boundary is a boundary by which the pattern classifier such as the Real-BP classifies patterns, and generally consists of hypersurfaces. Decision boundaries of real-valued neural networks have been examined empirically by Lippmann (1987). This section mathematically analyzes the decision boundaries of the complex-valued neural network.

3.1 A Case of a Single Neuron

We first analyze the decision boundary of a single complex-valued neuron (i.e., the number of hidden layers is zero).

Let the weights denote $\boldsymbol{w} = {}^t[w_1 \cdots w_n] = \boldsymbol{w}^r + i\boldsymbol{w}^i$, $\boldsymbol{w}^r = {}^t[w_1^r \cdots w_n^r]$, $\boldsymbol{w}^i = {}^t[w_1^i \cdots w_n^i]$, and let the threshold denote $\theta = \theta^r + i\theta^i$. Then, for n input signals (complex numbers) $\boldsymbol{z} = {}^t[z_1 \cdots z_n] = \boldsymbol{x} + i\boldsymbol{y}$, $\boldsymbol{x} = {}^t[x_1 \cdots x_n]$, $\boldsymbol{y} = {}^t[y_1 \cdots y_n]$, the complex-valued neuron generates

$$X + iY = f_R\left([{}^t\boldsymbol{w}^r \quad -{}^t\boldsymbol{w}^i] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} + \theta^r\right) + i f_R\left([{}^t\boldsymbol{w}^i \quad {}^t\boldsymbol{w}^r] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} + \theta^i\right) \tag{3}$$

as an output. Here, for any two constants $C^R, C^I \in (0, 1)$, let

$$X(\boldsymbol{x}, \boldsymbol{y}) = f_R\left([{}^t\boldsymbol{w}^r \quad -{}^t\boldsymbol{w}^i] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} + \theta^r\right) = C^R, \tag{4}$$

$$Y(\boldsymbol{x}, \boldsymbol{y}) = f_R\left([{}^t\boldsymbol{w}^i \quad {}^t\boldsymbol{w}^r] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} + \theta^i\right) = C^I. \tag{5}$$

Note here that expression (4) is the decision boundary for the *real part* of an output of the complex-valued neuron with n-inputs. That is, input signals $(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{R}^{2n}$ are classified into two decision regions $\{(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{R}^{2n} | X(\boldsymbol{x}, \boldsymbol{y}) \geq C^R\}$ and $\{(\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{R}^{2n} | X(\boldsymbol{x}, \boldsymbol{y}) < C^R\}$ by the hypersurface given by expression (4). Similarly, expression (5) is the decision boundary for the *imaginary part*. The normal vectors $Q^R(\boldsymbol{x}, \boldsymbol{y})$ and $Q^I(\boldsymbol{x}, \boldsymbol{y})$ of the decision boundaries ((4), (5)) are given by

$$\begin{aligned} Q^R(\boldsymbol{x}, \boldsymbol{y}) &= \left(\frac{\partial X}{\partial x_1} \cdots \frac{\partial X}{\partial x_n} \, \frac{\partial X}{\partial y_1} \cdots \frac{\partial X}{\partial y_n}\right) \\ &= f_R'\left([{}^t\boldsymbol{w}^r \quad -{}^t\boldsymbol{w}^i] \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} + \theta^r\right) \cdot [{}^t\boldsymbol{w}^r \quad -{}^t\boldsymbol{w}^i], \end{aligned} \tag{6}$$

$$
\begin{aligned}
Q^I(x,y) &= \left(\frac{\partial Y}{\partial x_1} \cdots \frac{\partial Y}{\partial x_n}\frac{\partial Y}{\partial y_1} \cdots \frac{\partial Y}{\partial y_n}\right) \\
&= f'_R\left([{}^t w^i \quad {}^t w^r]\begin{bmatrix} x \\ y \end{bmatrix} + \theta^i\right) \cdot [{}^t w^i \ {}^t w^r]. \qquad (7)
\end{aligned}
$$

Noting that the inner product of expressions (6) and (7) is zero, we can find that the decision boundary for the real part of an output of a complex-valued neuron and that for the imaginary part intersect orthogonally.

It can be easily shown that this orthogonal property also holds true for the other types of the complex-valued neurons proposed in (Kim and Guest 1990, Benvenuto and Piazza 1992, Georgiou and Koutsougeras 1992).

Generally, a real-valued neuron classifies an input real-valued signal into two classes (0, 1). On the other hand, a complex-valued neuron classifies an input complex-valued signal into four classes ($0,\ 1,\ i,\ 1+i$). As described above, the decision boundary of a complex-valued neuron consists of two hypersurfaces which intersect orthogonally, and divides a decision region into four equal sections. Thus, it can be considered that a complex-valued neuron has a natural decision boundary for complex-valued patterns.

3.2 A Case of a Three-Layered Network

Next, we examine the decision boundary of a three-layered complex-valued neural network (i.e., it has one hidden layer). Consider a three-layered complex-valued neural network with L input neurons, M hidden neurons, and N output neurons. We use $w_{ji} = w^r_{ji} + iw^i_{ji}$ for the weight between the input neuron i and the hidden neuron j, $v_{kj} = v^r_{kj} + iv^i_{kj}$ for the weight between the hidden neuron j and the output neuron k, $\theta_j = \theta^r_j + i\theta^i_j$ for the threshold of the hidden neuron j, $\gamma_k = \gamma^r_k + i\gamma^i_k$ for the threshold of the output neuron k. Then, for L input (complex-valued) signals $z = {}^t[z_1 \cdots z_L] = x + iy,\ x =$

${}^t[x_1 \cdots x_L]$, $y = {}^t[y_1 \cdots y_L]$, the net input U_j to the hidden neuron j is given by

$$\begin{aligned} U_j &= U_j^r + iU_j^i \\ &= \left[\sum_{i=1}^{L}(w_{ji}^r x_i - w_{ji}^i y_i) + \theta_j^r\right] + i\left[\sum_{i=1}^{L}(w_{ji}^i x_i + w_{ji}^r y_i) + \theta_j^i\right]. \end{aligned} \tag{8}$$

Hence, the output H_j of the hidden neuron j is given by

$$H_j = H_j^r + iH_j^i = f_R(U_j^r) + if_R(U_j^i). \tag{9}$$

And also, the net input S_k to the output neuron k is given by

$$\begin{aligned} S_k &= S_k^r + iS_k^i \\ &= \left[\sum_{j=1}^{M}(v_{kj}^r H_j^r - v_{kj}^i H_j^i) + \gamma_k^r\right] + i\left[\sum_{j=1}^{M}(v_{kj}^i H_j^r + v_{kj}^r H_j^i) + \gamma_k^i\right]. \end{aligned} \tag{10}$$

Hence, the output O_k of the output neuron k is given by

$$O_k = O_k^r + iO_k^i = f_R(S_k^r) + if_R(S_k^i). \tag{11}$$

Here, for any two constants $C^R, C^I \in (0,1)$, let

$$O_k^r(\boldsymbol{x}, \boldsymbol{y}) = C^R, \tag{12}$$
$$O_k^i(\boldsymbol{x}, \boldsymbol{y}) = C^I. \tag{13}$$

The expressions (12) and (13) are the decision boundaries for the real and imaginary parts of the output neuron k in the 3-layered complex-valued neural network, respectively. The normal vectors $Q^R(\boldsymbol{x}, \boldsymbol{y})$, $Q^I(\boldsymbol{x}, \boldsymbol{y})$ of these hypersurfaces ((12), (13)) are given by

$$Q^R(\boldsymbol{x}, \boldsymbol{y}) = \left(\frac{\partial O_k^r}{\partial x_1} \cdots \frac{\partial O_k^r}{\partial x_L}\ \frac{\partial O_k^r}{\partial y_1} \cdots \frac{\partial O_k^r}{\partial y_L}\right), \tag{14}$$

$$Q^I(\boldsymbol{x}, \boldsymbol{y}) = \left(\frac{\partial O_k^i}{\partial x_1} \cdots \frac{\partial O_k^i}{\partial x_L}\ \frac{\partial O_k^i}{\partial y_1} \cdots \frac{\partial O_k^i}{\partial y_L}\right), \tag{15}$$

and their inner product is given by

$$\begin{aligned} Q^R(x,y) \cdot {}^tQ^I(x,y) = & \frac{\partial O_k^r}{\partial x_1} \cdot \frac{\partial O_k^i}{\partial x_1} + \cdots + \frac{\partial O_k^r}{\partial x_L} \cdot \frac{\partial O_k^i}{\partial x_L} \\ & + \frac{\partial O_k^r}{\partial y_1} \cdot \frac{\partial O_k^i}{\partial y_1} + \cdots + \frac{\partial O_k^r}{\partial y_L} \cdot \frac{\partial O_k^i}{\partial y_L}. \end{aligned} \tag{16}$$

Note here that, for any $1 \le i \le L$,

$$\begin{aligned} & \frac{\partial O_k^r}{\partial x_i} \cdot \frac{\partial O_k^i}{\partial x_i} + \frac{\partial O_k^r}{\partial y_i} \cdot \frac{\partial O_k^i}{\partial y_i} \\ = \ & \frac{\partial f_R(S_k^r)}{\partial S_k^r} \cdot \frac{\partial f_R(S_k^i)}{\partial S_k^i} \\ & \cdot \left[\sum_{j=1}^{M} \left(v_{kj}^r w_{ji}^r \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} - v_{kj}^i w_{ji}^i \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} \right) \right] \\ & \cdot \left[\sum_{j=1}^{M} \left(v_{kj}^i w_{ji}^r \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} + v_{kj}^r w_{ji}^i \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} \right) \right] \\ & - \frac{\partial f_R(S_k^r)}{\partial S_k^r} \cdot \frac{\partial f_R(S_k^i)}{\partial S_k^i} \\ & \cdot \left[\sum_{j=1}^{M} \left(v_{kj}^r w_{ji}^r \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} - v_{kj}^i w_{ji}^i \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} \right) \right] \\ & \cdot \left[\sum_{j=1}^{M} \left(v_{kj}^i w_{ji}^r \cdot \frac{\partial f_R(U_j^i)}{\partial U_j^i} + v_{kj}^r w_{ji}^i \cdot \frac{\partial f_R(U_j^r)}{\partial U_j^r} \right) \right]. \end{aligned} \tag{17}$$

Hence, the inner product of the normal vectors is not always zero. Therefore, we can not conclude that the decision boundaries (hypersurfaces) for the real and imaginary parts of the output neuron k in the 3-layered complex-valued neural network intersect orthogonally. However, paying enough attention to expression (17), we can find that if

$$\frac{\partial f_R(U_j^r)}{\partial U_j^r} = \frac{\partial f_R(U_j^i)}{\partial U_j^i} \tag{18}$$

for any $1 \leq j \leq M$, then the inner product is zero. In general, if both $|u_1|$ and $|u_2|$ are sufficiently large, we can consider that $f'_R(u_1)$ is nearly equal to $f'_R(u_2)$. Hence, if, for any $1 \leq j \leq M$, there exist sufficiently large positive real numbers K_1, K_2 such that

$$|U_j^r| > K_1, \qquad (19)$$

$$|U_j^i| > K_2, \qquad (20)$$

then the two decision boundaries ((12), (13)) intersect orthogonally. That is, if, for any $1 \leq j \leq M$, both the absolute values of the real and imaginary parts of the net input (complex number) to the hidden neuron j are sufficiently large, then the decision boundaries intersect orthogonally. Therefore, the following theorem can be obtained.

Theorem If both the absolute values of the real and imaginary parts of the net inputs to all hidden neurons are sufficiently large, then the decision boundaries for the real and imaginary parts of an output neuron in the 3-layered complex-valued neural network intersect orthogonally.

4 Simulation

We present below the simulation results on the decision boundaries of the three-layered complex-valued neural networks trained using the Complex-BP (called *Complex-BP network*) (Nitta and Furuya 1991, Nitta 1993, Nitta 1997) and compare them with those of the three-layered real-valued neural networks trained using the Real-BP (called *Real-BP network*) (Rumelhart *et al.* 1986)

In the experiments, the three sets of (complex-valued) learning patterns shown in Tables 1-3 were used, and the learning constant ε was 0.5. The initial components of the weights and the thresholds were chosen to be random real numbers between -0.3 and 0.3. We judged that learning finished, when

$$\sqrt{\sum_{p}\sum_{k=1}^{N}|T_k^{(p)} - O_k^{(p)}|^2} \;=\; 0.05 \tag{21}$$

held, where $T_k^{(p)}$, $O_k^{(p)} \in \boldsymbol{C}$ denoted the desired output value, the actual output value of the output neuron k for the pattern p, i.e., the left side of expression (21) meant the error between the desired output pattern and the actual output pattern; N denoted the number of neurons in the output layer, $\boldsymbol{C}$ denoted the set of complex numbers. We regarded presenting a set of learning patterns to the neural network as one learning cycle.

Table 1. Learning pattern 1.

Input pattern	*Output pattern*
$-0.03 - 0.03i$	$1 + i$
$0.03 - 0.03i$	i
$0.03 + 0.03i$	0
$-0.03 + 0.03i$	1

Table 2. Learning pattern 2.

Input pattern	*Output pattern*
$-0.03 - 0.03i$	i
$0.03 - 0.03i$	0
$0.03 + 0.03i$	1
$-0.03 + 0.03i$	$1 + i$

Table 3. Learning pattern 3.

Input pattern	*Output pattern*
$-0.03 - 0.03i$	0
$0.03 - 0.03i$	1
$0.03 + 0.03i$	$1 + i$
$-0.03 + 0.03i$	i

We used the four kinds of three-layered Complex-BP networks:

1-3-1, 1-6-1, 1-9-1 and 1-12-1 networks. After training, by presenting the 1,681(=41×41) points in the complex plane $[-1,1] \times [-1,1]$ ($x + iy$, where $x = -1.0, -0.95, \cdots, 0.95, 1.0; y = -1.0, -0.95, \cdots, 0.95, 1.0$), the actual output points formed the decision boundaries. Figure 1 shows an example of the decision boundary of the Complex-BP network. In Figure 1, the number **1** denotes the region in which the real part of the output value of the neural network is **OFF** (0.0-0.5), and the imaginary part **OFF**; the region **2** the real part **ON** (0.5-1.0), and the imaginary part **OFF**; the region **3** the real part **OFF**, and the imaginary part **ON**; the region **4** the real part **ON**, and the imaginary part **ON**. And the decision boundary for the real part (i.e., the boundary that the region "**1+3**" and the region "**2+4**" form) and that for imaginary part (i.e., the boundary that the region "**1+2**" and the region "**3+4**" form) intersect orthogonally.

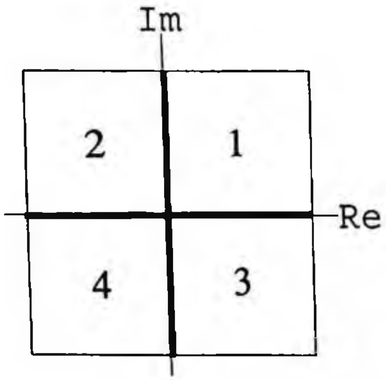


Figure 1. An example of the decision boundary of the 1-12-1 Complex-BP network learned with the learning pattern 1. The meanings of the numerals are as follows. **1**: Real part **OFF**(0.0-0.5), Imaginary part **OFF**, **2**: Real part **ON**(0.5-1.0), Imaginary part **OFF**, **3**: Real part **OFF**, Imaginary part **ON**, and **4**: Real part **ON**, Imaginary part **ON**. The decision boundary for the real part (i.e., the boundary that the region '**1+3**" and the region '**2+4**" form) and that for imaginary part (i.e., the boundary that the region '**1+2**" and the region '**3+4**" form) intersect orthogonally.

We also conducted the corresponding experiments for the Real-BP networks. We chose the 2-4-2 Real-BP network for the 1-3-1

Complex-BP network as a comparison object because the numbers of the parameters (weights and thresholds) were almost the same: the number of parameters for the 1-3-1 Complex-BP network was 20, and that for the 2-4-2 Real-BP network 22 where a complex-valued parameter $z = x + iy$ (where $i = \sqrt{-1}$) was counted as two because it consisted of a real part x and an imaginary part y. Similarly, the 2-7-2, 2-11-2 and 2-14-2 Real-BP networks were chosen for the 1-6-1, 1-9-1 and 1-12-1 Complex-BP networks as their comparison objects, respectively. The numbers of parameters of them are shown in Table 4. In the Real-BP networks, the real component of a complex number was input into the first input neuron, and the imaginary component was input into the second input neuron; the output from the first output neuron was interpreted to be the real component of a complex number, and the output from the second output neuron was interpreted to be the imaginary component. Figure 2 shows an example of the decision boundary of the Real-BP network where the numbers **1**-**4** have the same meanings as those of Figure 1. We can find from Figure 2 that the decision boundary for the real part (i.e., the boundary that the region "**1+3**" and the region "**2+4**" form) and that for imaginary part (i.e., the boundary that the region "**1+2**" and the region "**3+4**" form) do not intersect orthogonally.

Table 4. The number of parameters in the Real-BP and Complex-BP networks.

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
The number of parameters	20	38	56	74
Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
The number of parameters	22	37	57	72

First, we measured the angles between the decision boundary for the real part (i.e., the boundary that the region "**1+3**" and the region "**2+4**" formed) and that for imaginary part (i.e., the boundary that the region "**1+2**" and the region "**3+4**" formed) which were the components of the decision boundary of the output neuron in the Complex-

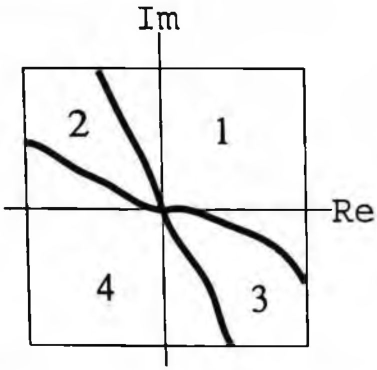


Figure 2. An example of the decision boundary of the 2-14-2 Real-BP network learned with the learning pattern 1. The numbers **1-4** have the same meanings as those of Figure 1. The decision boundary for the real part (i.e., the boundary that the region "**1+3**" and the region "**2+4**" form) and that for imaginary part (i.e., the boundary that the region "**1+2**" and the region "**3+4**" form) do not intersect orthogonally.

BP networks in the visual observation under the experimental conditions described above. The average and the standard deviation of the angles of 100 trials for each of the 3 learning patterns and each of the 4 kinds of network structures were used as the evaluation criterion. Although we stopped learning at the 200,000th iteration, all trials succeeded in converging. And also, we measured the same quantities of the Real-BP networks for the comparison. The results of the experiments are shown in Table 5(a)-(c). We can find from Table 5 that all the average angles for the Complex-BP networks are almost 90 degrees, which are independent of the learning patterns and the network structures, whereas those of the Real-BP networks are around 70-80 degrees. In addition, the standard deviations of the angles for the Complex-BP networks are around 0-5 degrees and those for the Real-BP networks around 20 degrees. Thus, we can conclude from the experimental results that the decision boundary for the real part and that for imaginary part which are the components of the decision boundary of the output neuron in the three-layered Complex-BP networks almost intersect orthogonally, whereas those for the Real-BP networks do not.

Table 5. Comparison of the angles of the decision boundaries (the average and the standard deviation). The unit is *degree*.

(a) Pattern 1

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	90	90	90	90
Standard deviation	0	0	0	0

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	78	72	76	80
Standard deviation	19	22	17	17

(b) Pattern 2

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	89	90	90	90
Standard deviation	6	0	0	0

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	85	77	77	75
Standard deviation	16	20	18	21

(c) Pattern 3

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	90	89	90	90
Standard deviation	0	5	3	0

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	86	76	72	73
Standard deviation	11	20	22	22

It seems that the generalization ability of neural networks can be improved if the decision boundaries of the network intersect orthogonally. Next, we measured the discrimination rate of the Complex-BP network for unlearned patterns in order to clarify how the orthogonality of the decision boundary of the 3-layered Complex-BP network changed its generalization ability.

To specifically, we counted the number of the test patterns for which the Complex-BP network could give the correct output in the same experiments described above on the angles of decision boundaries of

100 trials for each of the 3 learning patterns and each of the 4 kinds of network structures. We defined the correctness as follows: the output value $X + iY (0 \leq X, Y \leq 1)$ of the Complex-BP network for an unlearned pattern $x + iy$ $(-1 \leq x, y \leq 1)$ was correct if $|X - A| < 0.5$ and $|Y - B| < 0.5$, provided that the closest input learning pattern to the unlearned pattern $x + iy$ was $a + ib$ whose corresponding output learning pattern was $A + iB$ $(A, B = 0 \text{ or } 1)$. For example, the output value $X + iY (0 \leq X, Y \leq 1)$ of the Complex-BP network for an unlearned pattern $x + iy$ $(0 \leq x, y \leq 1)$ was correct if both the real and imaginary parts of the output value of the Complex-BP network took value less than 0.5, provided that the corresponding output learning pattern for the input learning pattern $0.03 + 0.03i$ was 0. Then, the average and the standard deviation of the discrimination rate of 100 trials for each of the 3 learning patterns and each of the 4 kinds of network structures were used as the evaluation criterion. The results of the experiments including the Real-BP network case appear in Table 6(a)-(c). The above simulation results clearly suggest that the Complex-BP network has better generalization performance than that of the Real-BP network. We believe that these results are caused by the orthogonality of the decision boundaries.

Table 6. Comparison of the generalization ability (the average and the standard deviation). The unit is *percentage*.

(a) Pattern 1

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	92	95	97	98
Standard deviation	6	5	3	2
Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	88	90	93	93
Standard deviation	8	7	4	4

(b) Pattern 2

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	93	95	96	97
Standard deviation	6	4	4	3

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	88	91	92	93
Standard deviation	8	7	5	6

(c) Pattern 3

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	92	94	97	97
Standard deviation	7	4	3	3

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	87	90	90	92
Standard deviation	9	7	7	6

Finally, we investigated the average and the standard deviation of the learning speed (i.e., learning cycles needed to converge) of 100 trials for each of the 3 learning patterns and each of the 4 kinds of network structures in the experiments described above. The results of the experiments are shown in Table 7(a)-(c). We can find from these experiments that the learning speed of the Complex-BP is several times faster than that of the Real-BP, and the standard deviation of the learning speed of the Complex-BP is smaller than that of the Real-BP.

Table 7. Comparison of the learning speed (the average and the standard deviation). The unit is *learning cycle*.

(a) Pattern 1

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	10770	10178	9766	9529
Standard deviation	438	472	210	144

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	31647	29945	28947	28230
Standard deviation	1268	944	697	566

(b) Pattern 2

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	10608	9932	9713	9539
Standard deviation	418	167	148	110

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	31781	29842	28902	28267
Standard deviation	2126	809	721	576

(c) Pattern 3

Complex-BP network	1-3-1	1-6-1	1-9-1	1-12-1
Average	10746	10055	9713	9502
Standard deviation	740	412	228	188

Real-BP network	2-4-2	2-7-2	2-11-2	2-14-2
Average	34038	29620	28980	28471
Standard deviation	5182	806	603	584

5 Discussion

We have proved that the decision boundary for the real part of an output of a single complex-valued neuron and that for the imaginary part intersect orthogonally in Section 3.1. Since this property is completely different from an usual real-valued neuron, one needs to design the complex-valued neural network for real applications and its learning algorithm taking into account the orthogonal property of the complex-valued neuron whatever the type of the network is (multi-layered type or mutually-connected type).

It is not always guaranteed that the decision boundary of the 3-layered complex-valued neural network has the orthogonality, as we have made clear in Section 3.2. Then, we have derived a sufficient condition for the decision boundaries in the 3-layered complex-valued neural network to intersect orthogonally in Section 3.2 (Theorem). The sufficient condition was as follows: both the absolute values of the real and imaginary parts of the net inputs to all hidden neurons were sufficiently large. This is a *characterization* for the structure of the decision boundaries in the 3-layered complex-valued neural network. The theorem will be useful if a learning algorithm such that both the absolute values of the real and imaginary parts of the net inputs to all hidden neurons become sufficiently large is devised, because the orthogonality of the decision boundaries of the network can improve the generalization ability of 3-layered complex-valued neural networks as we have seen in Section 4. That is, we can utilize the theorem in order to improve the generalization ability of the 3-layered complex-valued neural network. However, the situation in which the theorem is directly useful for the Complex-BP network cannot be considered regrettably for now, because the control of the net input is difficult as long as the Complex-BP algorithm (that is, steepest descent method) is used. The Complex-BP is one of the learning algorithms for complex-valued neural networks. Thus it should be noted that the usefulness of the theorem depends on the learning algorithm used. Although the orthogonality of the decision boundaries in the 3-layered complex-valued neural network can be guaranteed conditionally as described above, we can find from the experiments in Section 4 that most of the decision boundaries in the 3-layered *Complex-BP network* intersect orthogonally. Moreover, it is learned from the experiments that the orthogonality of the decision boundaries in the 3-layered *Complex-BP network* improves its generalization ability. The decision boundary of the complex-valued neural network which consists of two orthogonal hypersurfaces divides a decision region into four equal sections. So, it is intuitively considered that the orthogonality of the decision boundaries improves its generalization ability. Then, we have experimentally proved that it is true.

It had already been reported that the average of the learning speed of the Complex-BP was several times faster than that of the Real-BP (Nitta 1997). In this connection, we could confirm this again in the experiments on the othogonality of the decision boundary and the generalization ability of the 3-layered Complex-BP network in Section 4. It was learned that the standard deviation of the learning speed of the Complex-BP was smaller than that of the Real-BP, which had

Nitta, T. and Furuya, T. (1991), "A complex back-propagation learning," *Transactions of Information Processing Society of Japan*, vol.32, no.10, pp.1319-1329 (in Japanese).

Nitta, T. (1993), "A complex numbered version of the back-propagation algorithm," *Proceedings of World Congress on Neural Networks*, WCNN'93, Portland, July, vol.3, pp.576-579.

Nitta, T. (1997), "An extension of the back-propagation algorithm to complex numbers," *Neural Networks*, vol.10, no.8, pp.1392-1415.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986), *Parallel Distributed Processing*, vol.1, MIT Press.

Watanabe, A., Yazawa, N., Miyauchi, A., and Miyauchi, M. (1994), "A method to interpret 3D motions using neural networks," *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, E77-A(8), pp.1363-1370.

Author's address

Tohru Nitta: Mathematical Neuroinformatics Group,
Neuroscience Research Institute,
National Institute of Advanced Industrial Science and Technology (AIST),
AIST Tsukuba Central 2, 1-1-1 Umezono, Tsukuba-shi, Ibaraki, 305-8568 Japan.
tohru-nitta@aist.go.jp

Chapter 3

Complex-Valued Neural Associative Memories: Network Stability and Learning Algorithm

Donq-Liang Lee

An associative memory model called complex-valued neural network (CVNN) is presented in this chapter. In a CVNN states are represented by quantization values defined on the unit circle of the complex plane. Such a network is able to perform the task of storing and recalling gray-scale images. The stability properties under different updating modes are investigated by using the energy function approach. It is proved that the model will be globally convergent to a fixed point when operating in a asynchronous mode and to a cycle of length at most 2 when operating in a synchronous mode. Then, some existing learning methods for this model are reviewed and discussed. Finally, simulation results are presented to illustrate the performance of this model.

1 Introduction

Conventional neural networks are usually based on two-state neurons, i.e., the states of the networks are usually bipolar (1 and -1) or binary (1 and 0). Although binary representations are widely used in engineering applications for their simplicity, multivalued representation is a much relevant and direct approximation to real world data. The most simple way to store multivalued patterns in a bipolar network is grouping a certain number of neurons into one macro neuron (Cernuschi-Frias 1989, Lee 1999) that functionally represents a single multivalued state. However, this scheme needs a much greater number of neurons and connection weights, which increases the programming complexity. Moreover, it is difficult to implement this method by hardware. In very large scale integration (VLSI) implementations of neural networks, reductions in the number of neurons and in the number of weighting connections are highly desirable. This leads to the development of multivalued neural networks. (e.g., multi-valued exponential associative memories (Chiueh and Tsai 1993); multivalued recurrent nonlinear associative memory (MAREN)(Erdem and Ozturk 1996); multilevel threshold neurons (Zurada *et al.* 1997; Si and Michel 1995)).

In 1996, Jankowski *et al* proposed a complex-valued neural network (CVNN) which is capable of storing and recalling gray-scale images. The CVNN (Jankowski *et al 1996)* is composed of fully connected multistate complex-valued neurons and the information representation is based on amplitude and phase coding. It can be referred to as a modified Hopfield network having complex-signum activation functions and complex weighting connections. In this chapter, the background of the CVNNs was reviewed first. The structure and updating modes of the CVNNs were discussed in section 2. Theorems regarding network stability under different conditions were examined in section 3. Then, some existing learning methods for the CVNNs were reviewed in section 4. Finally, simulation results were presented in section 5 to illustrate the performance of the CVNNs.

2 Network Structure and Updating Modes

2.1 Structure of CVNNs

The complex-valued neural network (CVNN) introduced in this chapter is an autoassociative memory that stores complex-valued prototype vectors X^k, $k = 1, ..., m$, where $X^k = (x_1^k, x_2^k, ..., x_N^k)^T$ and m is the number of the prototype vectors. The components x_i^ks are all quantization values defined by

$$x_i^k \in \exp[j2\pi v/K]_{v=0}^{K-1} \qquad i = 1, ..., N. \tag{1}$$

The resolution factor K divides the complex unit circle into K quantization levels so that $|x_i^k| = 1\ \forall i, k$. A CVNN consists of N fully connected multi-state neurons. Let $X \in \mathbb{C}^N$ and $S \in \mathbb{C}^{N\times N}$ denote the state vector and the connection weight matrix of the CVNN, respectively. The output of each neuron is determined by the following equation:

$$x_i' = \phi\left\{\sum_{j=1}^{N} s_{ij}x_j\right\} \tag{2}$$

in which x_i is the ith component of X; x_i' denotes the next state of x_i. Moreover, $\phi(\cdot)$ is a complex-signum function

$$\phi(z) = \begin{cases} \exp(j0) & 0 \le \arg[ze^{j(\theta_0/2)}] < \theta_0 \\ \exp(j\frac{2\pi}{K}) & \theta_0 \le \arg[ze^{j(\theta_0/2)}] < 2\theta_0 \\ \vdots & \\ \exp\left[j\frac{2\pi(K-1)}{K}\right] & (K-1)\theta_0 \le \arg[ze^{j(\theta_0/2)}] \\ & < K\theta_0 \end{cases} \tag{3}$$

where arg(α) is the phase angle of α, θ_0 is a phase quantum delimited by K: $\theta_0 = 2\pi/K$. (3) means that $\phi(z)$ is the quantization value on the complex unit circle closest to z. The resolution factor K divides the complex unit circle into K separate sectors and each of them has an angle of θ_0. Note that if $K = 2$, the CVNN will be functionally equivalent to the Hopfield network (Hopfield 1982, 1984) in which all neuron states are bipolar real values (i.e., 1 or -1). The only difference is that the former permits complex-valued weighting connections but the latter does not.

2.2 Updating Modes of CVNNs

As in the case of bipolar Hopfield network, there are two updating modes for the CVNNs, i.e., *asynchronous* and *synchronous* modes. A recalling process is an iterative process that starts with an initial vector X_0 presented to the network. It is called asynchronous mode if neuron states are updated one at a time by following (2) with equal probabilities. If the neuron states are updated simultaneously according to the following equation

$$X' = \phi\{SX\} \tag{4}$$

, the CVNN is said to be operated in the synchronous mode. Here X'denotes the next state of X. We show in the following section that process (2) and (4) converge to one of the fixed points X_f in a finite number of iterations if the matrix $S = [s_{ij}]$satisfys specific matrix conditions. Apparently, for associative memory applications this fixed point is desired to be one of the stored vectors. If so, the stored vector is said to be recalled. A fixed point $X_f = [x_{f1}, x_{f2}, ..., x_{fN}]$ has the following properties:

$$x_{fi} = \phi\left\{\sum_{j=1}^{N} s_{ij}x_{fj}\right\}, \qquad i = 1, ..., N. \tag{5}$$

Obviously, a stored vector can be recalled only if it is a fixed point. In the original model (Jankowski *et al* 1996) the m prototype vectors are stored in the weight matrix according to the generalized Hebb rule

$$s_{ij} = \frac{1}{N}\sum_{k=1}^{m} x_i^k \bar{x}_j^k \qquad i, j = 1, ..., N \tag{6}$$

where the over bar means complex conjugate. However, as in the case of bipolar networks (Hopfield 1982, 1984), the encoding rule (6) can not ensure that all the stored vectors are fixed points of (5).

3 Network Stability

In this section some existing stability results were reviewed.

3.1 Stability in Asynchronous Mode

Theorem 1: Given a CVNN with a complex weight matrix $S = [s_{ij}]$ and assume that it is operating in asynchronous mode.

(i) If S is Hermitian with nonnegative diagonal entries

$$s_{ii} \geq 0, \quad i = 1, ..., N, \tag{7}$$

then the CVNN will converge to a fixed point from any given initial state (Jankoski *et al.* 1996);

(ii) if S is not Hermitian but it is weakly diagonally dominant in the sense that

$$\mathrm{Re}(s_{ii}) \geq \frac{1}{2\sin(\pi/K)} \sum_{j \neq i} |s_{ij} - \bar{s}_{ji}|, \quad i = 1, ..., N, \tag{8}$$

then the CVNN will converge to a fixed point from any given initial state (Lee 2001b).

Proof: We first prove (ii) since (i) can be referred to as a special case of (ii). Let $S = W + T$ where W is the cross-connection matrix,

$$W = \begin{pmatrix} 0 & w_{12} & \cdots & w_{1N} \\ w_{21} & 0 & & \vdots \\ \vdots & & \ddots & w_{N-1,N} \\ w_{N1} & \cdots & w_{N,N-1} & 0 \end{pmatrix},$$

$w_{ii} = 0 \; \forall i$, and T is the self-connection matrix,

$$T = \begin{pmatrix} t_{11} & 0 & \cdots & 0 \\ 0 & t_{22} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & t_{NN} \end{pmatrix},$$

$t_{ij} = 0 \; \forall \, i \neq j$. Then, the energy function of a CVNN can be defined as

$$\begin{aligned} E(X) &= -\frac{1}{2}\,\mathrm{Re}[X^*SX] \\ &= -\frac{1}{2}\,\mathrm{Re}[X^*(W+T)X]. \end{aligned}$$

Assume the transition from a state X to its next state $X + \Delta X$,

$$\begin{aligned} \Delta E &= E(X+\Delta X) - E(X) \\ &= -\frac{1}{2}\,\mathrm{Re}[(X+\Delta X)^*(W+T)(X+\Delta X)] \\ &\quad +\frac{1}{2}\,\mathrm{Re}[X^*(W+T)X]. \end{aligned}$$

The objective is to find the condition on S such that $\Delta E \leq 0$, and $\Delta E = 0$ if and only if $\Delta X = 0$. After some algebra one obtains

$$\begin{aligned} \Delta E &= \mathrm{Re}\left\{-\Delta X^*(W+T)X + \frac{1}{2}\Delta X^* W X \right. \\ &\quad \left. -\frac{1}{2}X^* W \Delta X - \frac{1}{2}\Delta X^* W \Delta X - \frac{1}{2}\Delta X^* T \Delta X\right\} \\ &= \mathrm{Re}\left\{-\Delta X^* S X + \frac{1}{2}\Delta X^*(W - W^*)X - \frac{1}{2}\Delta X^* T \Delta X\right\} \end{aligned}$$

Since the network is operated in asynchronous mode, assume $\Delta X = X' - X$ and the difference between X' and X only comes from the

ith component, i.e., $\Delta X = (0, 0, ..., \Delta x_i, ..., 0)^T$ and $\Delta x_i = x_i' - x_i$, so

$$\begin{aligned} \Delta E &= \mathrm{Re}\left\{ (\bar{x}_i - \bar{x}_i') \textstyle\sum_{j=1}^N s_{ij} x_j + \frac{1}{2} \Delta X^* (W - W^*) X \right. \quad (9) \\ &\quad \left. - \frac{1}{2} t_{ii} \left| \Delta x_i \right|^2 \right\} \end{aligned}$$

Now let

$$\sum_{j=1}^N s_{ij} x_j = r_i \exp(j\phi_i), \ \bar{x}_i = \exp(-j\psi_i), \text{ and } \bar{x}_i' = \exp(-j\psi_i'),$$

where r_i and ϕ_i denote the modulus and phase angle of $\sum_{j=1}^N s_{ij} x_j$, respectively. Moreover, let $\psi_i = \arg(x_i)$ and $\psi_i' = \arg(x_i')$. We find

$$\begin{aligned} & \mathrm{Re}\left\{ (\bar{x}_i - \bar{x}_i') \sum_{j=1}^N s_{ij} x_j \right\} \qquad (10) \\ = \ & r_i \, \mathrm{Re}\{\exp[j(\phi_i - \psi_i)] - \exp[j(\phi_i - \psi_i')]\} \\ = \ & r_i [\cos(\phi_i - \psi_i) - \cos(\phi_i - \psi_i')] \le 0 \end{aligned}$$

The last inequality follows from the fact that ψ_i' is the quantization value closest to ϕ_i. Also note that

$$\sum_{j \neq i} (w_{ij} - \bar{w}_{ji}) x_j = \sum_{j \neq i} (s_{ij} - \bar{s}_{ji}) x_j,$$

and

$$|x_i| = 1, \ \ |\Delta \bar{x}_i| = |\Delta x_i|, \text{ and } t_{ii} = s_{ii}, \forall i$$

From (9) and (10) we obtain

$$\Delta E \ \le \ \frac{1}{2} \mathrm{Re} \left\{ \Delta \bar{x}_i \sum_{j \neq i} (w_{ij} - \bar{w}_{ji}) x_j - t_{ii} \left| \Delta x_i \right|^2 \right\}$$

$$\begin{aligned}
&\leq \frac{1}{2}\left\{|\Delta\bar{x}_i|\left|\sum_{j\neq i}(s_{ij}-\bar{s}_{ji})x_j\right| - \mathrm{Re}(s_{ii})\,|\Delta x_i|^2\right\} \\
&\leq \frac{1}{2}\left\{|\Delta\bar{x}_i|\sum_{j\neq i}|s_{ij}-\bar{s}_{ji}| - \mathrm{Re}(s_{ii})\,|\Delta x_i|^2\right\} \\
&\leq 0.
\end{aligned}$$

The last inequality hold if we choose

$$\mathrm{Re}(t_{ii}) \geq \frac{1}{|\Delta x_i|}\sum_{j\neq i}|s_{ij}-\bar{s}_{ji}|$$

and $\Delta E = 0$ if and only if $\Delta x_i = 0$. Since E is bounded from below as long as $|s_{ij}|$ is bounded for all i and j, i.e.,

$$E = -\frac{1}{2}Re\{X^* S X\} \geq -\frac{1}{2}|X^* S X| \geq -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}|s_{ij}|$$

Therefore, in asynchronous mode and starting with any initial vector, a CVNN always converges to a fixed point. Moreover, the minimum of $|\Delta x_i|$ can be obtained by examining Figure 1,.i.e., $|\Delta x_i| \geq 2\sin(\pi/K)$. Hence we obtain (8). The proof of (ii) is complete.

Now consider the case that $s_{ij} = \bar{s}_{ji}\ \forall i \neq j$ (i.e., S is Hermitian), (8) will reduce to

$$\mathrm{Re}(s_{ii}) \geq 0, \quad i = 1, ..., N \tag{11}$$

which is very similar to (7). The difference reflects the fact that the imaginary part of s_{ii} gives no contribution to E. The proof of theorem 1 is thus complete.

Remark 1: If $K = 2$ and S is a real matrix, then (8) reduces to

$$s_{ii} \geq \frac{1}{2}\sum_{j\neq i}|s_{ij} - s_{ji}|, \quad i = 1, ..., N \tag{12}$$

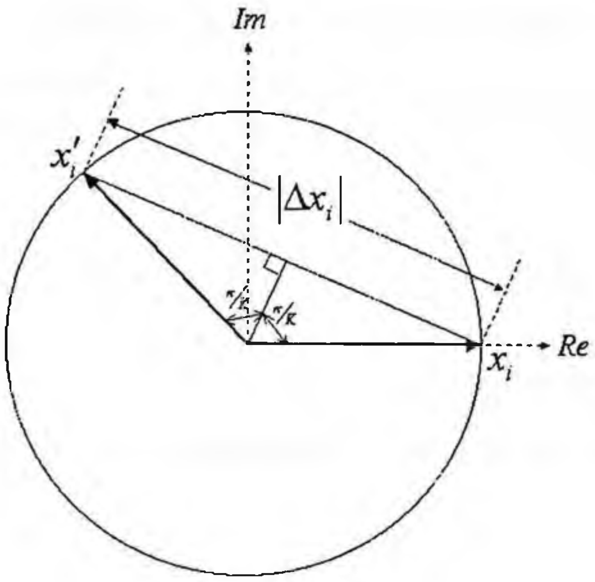


Figure 1. Schematic representation of the relation between $|\Delta\, x_i|$ and K

That is, condition (8) generalizes the stability condition (Xu and Kwong 1995, Xu *et al.* 1996, Lee 1999) of the bipolar Hopfield model (Hopfield 1982, 1984). Moreover, if S is permitted to be a complex matrix, (12) becomes

$$\operatorname{Re}(s_{ii}) \geq \frac{1}{2}\sum_{j\neq i}|s_{ij} - \bar{s}_{ji}|, \quad i = 1, ..., N \tag{13}$$

The meaning behind (13) is that one can design a bipolar real state network (i.e., $X \in \{1, -1\}^N$) by using complex weighting connections, $S \in \mathbb{C}^{N\times N}$. This formalism enhances the design flexibility of neural networks.

3.2 Stability in Synchronous Mode

Theorem 2 (Lee and Wang 1998): Given a CVNN with a complex weight matrix $S = [s_{ij}]$ and assume that it is operating in synchronous mode.

(i) If S is Hermitian and nonnegative definite, then the CVNN will converge to a fixed point from any given initial state;

(ii) if S is only Hermitian, then the CVNN will converge to a fixed point or to a cycle of length 2.

Proof: We first prove (i). Define the following energy function

$$E(X) = -\frac{1}{2}X^*SX.$$

Here no operator Re(·) is necessary since S is Hermitian. The energy change from the current state X to next state X' is

$$\begin{aligned}\Delta E &= E(X') - E(X) \\ &= -\frac{1}{2}(X' - X)^*S(X' - X) - X^*S(X' - X) \\ &\leq -X^*S(X' - X)\end{aligned}$$

The last inequality follows from the fact that S is nonnegative definite. Also note that $-X^*S(X' - X) \in \mathbf{R}$. Applying the same notations as in the proof of Theorem 1 yields

$$\begin{aligned}& -X^*S(X' - X) \\ =\ & -(X' - X)^*SX \\ =\ & -\sum_{i=1}^{N}(\bar{x}_i' - \bar{x}_i)\left\{\sum_{j=1}^{N} s_{ij}x_j\right\} \\ =\ & \sum_{i=1}^{N} r_i \operatorname{Re}\left\{\exp[j(\phi_i - \psi_i)] - \exp[j(\phi_i - \psi_i')]\right\}\end{aligned}$$

$$\begin{aligned} &= \sum_{i=1}^{N} r_i[\cos(\phi_i - \psi_i) - \cos(\phi_i - \psi_i')] \\ &\leq 0 \end{aligned}$$

As in the proof of theorem 1, the last ineqality follows because ψ_i' is the quantization value closest to ϕ_i. Therefore, we conclude that $\Delta E \leq 0$ and $\Delta E = 0$ only if $X' = X$, *i.e.*, $\psi_i' = \psi_i$, $\forall i$. The proof of (i) is complete.

For the prove of (ii), the following energy function is defined,

$$\begin{aligned} E(t) &= -\frac{1}{2}\{[X(t)]^* SX(t-1) + [X(t-1)]^* SX(t)\} \\ &= -\operatorname{Re}\{[X(t)]^* SX(t-1)\} \end{aligned}$$

Here we use $X(t)$ instead of X to represent the current state. Analogously, $X(t-1)$ and $X(t+1)$ will be used to represent the previous state and the next state, respectively. Now the energy E is denoted as a function of time because not only $X(t)$ but also $X(t-1)$ are used to determine the value of E (Goles *et al.* 1985). Next consider the energy change resulting from the state change in X by an alternative form of (4), i.e.,

$$X(t+1) = \phi\{SX(t)\}.$$

The difference between the energies of the next state $X(t+1)$ and the current state $X(t)$ is

$$\begin{aligned} \Delta E &= E(t+1) - E(t) \qquad (14) \\ &= -\frac{1}{2}\{[X(t+1) - X(t-1)]^* SX(t) \\ &\quad +[X(t)]^* S[X(t+1) - X(t-1)]\} \\ &= -\operatorname{Re}\left\{\sum_{i=1}^{N}[\bar{x}_i(t+1) - \bar{x}_i(t-1)]\left(\sum_{j=1}^{N} s_{ij}x_j(t)\right)\right\} \end{aligned}$$

Now let

$$\begin{aligned}\sum_{i=1}^{N} s_{ij}x_j(t) &= r_i \exp[j\phi_i(t)] \\ x_i(t-1) &= \exp[j\psi_i(t-1)]\end{aligned}$$

and

$$x_i(t+1) = \exp[j\psi_i(t+1)]$$

where r_i is the modulus of $\sum s_{ij}x_j(t)$; $\phi_i(t)$, $\psi_i(t-1)$, and $\psi_i(t+1)$ are the phase angles of $\sum s_{ij}x_j(t)$, $x_i(t-1)$, and $x_i(t+1)$, respectively. Note that the phase angle of $\sum s_{ij}x_j(t)$ and $x_i(t+1)$ are identical and equal to $\psi_i(t+1)$. Consequently, (14) can be further represented as

$$\begin{aligned}\Delta E &= -\operatorname{Re}\{\textstyle\sum_{i=1}^{N}\{\exp[-j\psi_i(t+1)] \\ &\quad - \exp[-j\psi_i(t-1)]r_i \exp[j\phi_i(t)]\}\} \\ &= \textstyle\sum_{i=1}^{N} r_i \operatorname{Re}\{\exp[j(\phi_i(t)-\psi_i(t-1))] \\ &\quad - \exp[j(\phi_i(t)-\psi_i(t+1))]\}\} \\ &= \sum_{i=1}^{N} r_i[\cos(\phi_i(t)-\psi_i(t-1)) - \cos(\phi_i(t)-\psi_i(t+1))] \\ &\le 0\end{aligned}$$

The last ineqality follows because $\psi_i(t+1)$ is the quantization value closest to $\phi_i(t)$. Therefore, we conclude that $\Delta E \le 0$ and $\Delta E = 0$ only if $\psi_i' = \psi_i \ \forall i$, that is, $X' = X$. Also note that E is bounded from below as long as $|s_{ij}|$ is bounded for all i and j, i.e.,

$$\begin{aligned}E &= -\operatorname{Re}\{[X(t)]^{*}SX(t-1)\} \\ &\ge -|\,[X(t)]^{*}SX(t-1)\,| \\ &\ge -\sum_{i=1}^{N}\sum_{j=1}^{N}|s_{ij}|\end{aligned}$$

Therefore, in synchronous mode and starting with any initial vector, the CVNN always converges to a fixed point or a limit cycle of period two. The proof of (ii) is complete.

4 Learning Rules

From above observations we know that in a CVNN states are updated according to the update equation; in addition, the network dynamics can be described by using energy function approaches. Iterating via a specific update mode ensures that the corresponding energy descrease so that the network eventually reaches an equilibrium state (fixed point) corresponding to a local energy minimum. A CVNN will behave as an associative memory network if these fixed points correspond to a set of prespecified prototype patterns. Recall that a fixed point $X_f = [x_{f1}, x_{f2}, ..., x_{fN}]$ has the following property:

$$X_f = \phi\{SX_f\}. \tag{15}$$

Obviously, a stored vector can be recalled only if it is a fixed point. In other words, we must find a weight matrix S such that

$$\phi\left\{SX^k\right\} = X^k, \quad \forall\, k = 1, ..., m \tag{16}$$

In this section three existing learning methods were reviewed and discussed.

4.1 Generalized Hebb Rule

In the original CVNN (Jankowski *et al.* 1996) the m prototype vectors are stored in the weight matrix according to the generalized Hebb rule

$$s_{ij} = \frac{1}{N}\sum_{k=1}^{m} x_i^k \bar{x}_j^k \qquad i, j = 1, ..., N \tag{17}$$

Equation (17) can be written in the matrix form as

$$S = \frac{1}{N}\sum_{k=1}^{m} X^k (X^k)^* \tag{18}$$

The weight matrix obtained from (18) is Hermitian and nonnegative definite because the quadratic form

$$\begin{aligned} X^*SX &= \left(\frac{1}{N}\right) X^* \sum_{k=1}^{m} X^k (X^k)^* X \\ &= \left(\frac{1}{N}\right) \sum_{k=1}^{m} X^* X^k (X^k)^* X \\ &= \left(\frac{1}{N}\right) \sum_{k=1}^{m} |X^* X^k|^2 \end{aligned}$$

is always greater than or equal to zero. From the theorems in section 3, the CVNN will always converge to a fixed point under both the asynchronous and the synchronous update modes. However, as in the case of bipolar networks (Hopfield 1982, 1984), the generalized Hebb rule (18) can not ensure that all the stored vectors are fixed points described by (16). Consider the following signal and noise expansion of $\phi\{SX^k\}$, i.e.,

$$\begin{aligned} \phi\{SX^k\} &= \phi\left\{\frac{1}{N}\sum_{i=1}^{m} X^i (X^i)^* X^k\right\} \\ &= \phi\left\{\frac{1}{N}[(X^k)^* X^k] X^k + \frac{1}{N}\sum_{i\neq k}^{m} [(X^i)^* X^k] X^i\right\} \\ &= \phi\left\{X^k + \frac{1}{N}\sum_{i\neq k}^{m} [(X^i)^* X^k] X^i\right\} \end{aligned}$$

The vector in the bracket may be viewed as a signal X^k plus a noise term

$$\frac{1}{N}\sum_{i\neq k}^{m} [(X^i)^* X^k] X^i.$$

Recall that the resolution factor K divides the complex unit circle into K separate sectors and each of them has an angle of θ. Therefore, $\phi\{SX^k\} = X^k$ if and only if all entries of the vector SX^k

locates in the same sectors as their counterparts of X^k and this is happen only when the prototype vectors are mutually (or nearly) orthogonal. Also note that the noise term increases in magnitude as the number of prototype vectors increases. Based on these reasons, the capacity by using generalized Hebb rule is low.

4.2 Pseudoinverse Rule

If we can find an S such that

$$SX^k = X^k, \quad k = 1, ..., m \tag{19}$$

then each of the prototype vectors will be a fixed point of (16). The condition (19) is more conservative than that in (16) since the transformation between prototype vectors is strictly linear in the former case. Let

$$\Sigma = (X^1, X^2, ..., X^m).$$

(19) yields the matrix equation

$$S\Sigma = \Sigma. \tag{20}$$

One solution of this set of Nm linear equations is given by

$$S = \Sigma\Sigma^I \tag{21}$$

where Σ^I denotes the pseudoinverse (Albert 1972) of Σ . For example, if Σ has full column rank, i.e., $(\Sigma^*\Sigma)$ is invertable, Σ^I can be computed by using a simple matrix equation:

$$\Sigma^I = (\Sigma^*\Sigma)^{-1}\Sigma^*.$$

In this case

$$S = \Sigma(\Sigma^*\Sigma)^{-1}\Sigma^*$$

will be a Hermitian and nonnegative definite matrix. As in the case of generalized Hebb rule, a CVNN constructed by pseudoinverse rule will always converge to a fixed point under both the asynchronous and the synchronous modes. However, when the columns of Σ are linearly dependent , Σ^I can be obtained by using the singular value decomposition (SVD) approach. In this case (21) will be the approximation solution of (20) which minimizes the error

$$E = \|S\Sigma - \Sigma\| .$$

Here $\|\cdot\|$ denotes the Euclidean norm operator.

The pseudoinverse rule guarantees to stored specified prototype vectors corresponding to fixed points provided that the prototype vectors are linearly independent. This implies that the capacity of this method is limited to N.

4.3 A Modified Gradient Descent Learning Rule

In the following we proposed a modified learning rule to improve the capacity of CVNN. Analogous to bipolar Hopfield network design (Perfetti 1991), here we search for solutions of (16) with the following two assumptions:

i) S is Hermitian. This guarantees the stability of the designed network under asynchronous mode.

ii) S is zero diagonal. This reduces the number of spurious memories and avoid trivial solutions.

Recall that the objective is to store m prototype vectors X^k, $k = 1, ..., m$, in the CVNN. The matrix S must satisfy the following system of $m \times N$ conditions:

$$x_i^k = \phi\left\{\sum_{j=1}^{N} s_{ij}x_j^k\right\},\ k = 1, ..., m;\ i = 1, ..., N. \qquad (22)$$

Since $\phi(z)$ is a projection function which projects z onto the nearest quantization level on the complex unit circle, the importance is the phase but not the amplitude of z. Keep this in mind, we consider the following cost function (Lee 1991a)

$$J = \sum_{k=1}^{m} \sum_{i=1}^{N} \epsilon(k, i) \tag{23}$$

where

$$\epsilon(k, i) = 1 - \frac{x_{R_i}^k a(k, i) + x_{I_i}^k b(k, i)}{\sqrt{a(k, i)^2 + b(k, i)^2}}. \tag{24}$$

In (24) $x_{R_i}^k$ and $x_{I_i}^k$ denote the real and imaginary part of x_i^k, respectively. Moreover,

$$a(k, i) = \mathrm{Re}\left\{\sum_{j=1}^{N} s_{ij} x_j^k\right\} \tag{25}$$

$$b(k, i) = \mathrm{Im}\left\{\sum_{j=1}^{N} s_{ij} x_j^k\right\} \tag{26}$$

The error $\epsilon(k, i)$ measures the difference between 1 and the cosin of the phase difference between $\arg[x_i^k]$ and $\arg[\sum_{j=1}^{N} s_{ij} x_j^k]$. The smaller the phase difference, the smaller the value of the error.

Since S is Hermitian and zero diagonal, i.e., $s_{ij} = \bar{s}_{ji}$; $s_{ii} = 0$, one has

$$\begin{aligned} &\sum_{j=1}^{N} s_{ij} x_j^k \\ = &\sum_{j=1}^{i-1} \bar{s}_{ji} x_j^k + \sum_{j=i+1}^{N} s_{ij} x_j^k \\ = &\sum_{j<i} (s_{R_{ji}} x_{R_j}^k + s_{I_{ji}} x_{I_j}^k) + \sum_{j>i} (s_{R_{ij}} x_{R_j}^k - s_{I_{ij}} x_{I_j}^k) \end{aligned}$$

$$\begin{aligned} & +j\left\{\sum_{j<i}(s_{R_{ji}}x^k_{I_j} - s_{I_{ji}}x^k_{R_j} + \sum_{j>i}(s_{R_{ij}}x^k_{I_j} + s_{I_{ij}}x^k_{R_j})\right\} \\ = \quad & a(k,i) + jb(k,i) \end{aligned}$$

Note that only s_{pq} $(p < q)$ is to be updated. After learning was terminated, the weight s_{pq} $(p > q)$ can be obtain by using the relation $s_{ij} = \bar{s}_{ji}$. The following gradient descent learning rule is used.

$$s_{pq}(t+1) = s_{pq}(t) - \eta\left\{\frac{\partial J}{\partial s_{R_{pq}}} + j\frac{\partial J}{\partial s_{I_{pq}}}\right\} \; (p < q) \tag{27}$$

where η is a learning coefficient, $0 < \eta < 1$. The gradient in (27) can be derived in the following way. By the chain rule,

$$\begin{aligned} \frac{\partial J}{\partial s_{R_{pq}}} &= \sum_{k=1}^{m}\left\{\frac{\partial J}{\partial \epsilon(k,p)}\frac{\partial \epsilon(k,p)}{\partial s_{R_{pq}}} + \frac{\partial J}{\partial \epsilon(k,q)}\frac{\partial \epsilon(k,q)}{\partial s_{R_{pq}}}\right\} \\ &= \sum_{k=1}^{m}\left\{\frac{\partial \epsilon(k,p)}{\partial s_{R_{pq}}} + \frac{\partial \epsilon(k,q)}{\partial s_{R_{pq}}}\right\} \end{aligned} \tag{28}$$

and

$$\frac{\partial J}{\partial s_{I_{pq}}} = \sum_{k=1}^{m}\left\{\frac{\partial \epsilon(k,p)}{\partial s_{I_{pq}}} + \frac{\partial \epsilon(k,q)}{\partial s_{I_{pq}}}\right\} \tag{29}$$

Note that

$$\begin{aligned} \frac{\partial \epsilon(k,p)}{\partial s_{R_{pq}}} &= \frac{\partial \epsilon(k,p)}{\partial a(k,p)}\frac{\partial a(k,p)}{\partial s_{R_{pq}}} + \frac{\partial \epsilon(k,p)}{\partial b(k,p)}\frac{\partial b(k,p)}{\partial s_{R_{pq}}} \\ &= \epsilon_a(k,p)x^k_{R_q} + \epsilon_b(k,p)x^k_{I_q} \end{aligned} \tag{30}$$

where

$$\epsilon_a(k,p) \triangleq \frac{\partial \epsilon(k,p)}{\partial a(k,p)}; \tag{31}$$

$$\epsilon_b(k,p) \triangleq \frac{\partial \epsilon(k,p)}{\partial b(k,p)}. \tag{32}$$

Similar derivations give

$$\frac{\partial \epsilon(k,q)}{\partial s_{R_{pq}}} = \epsilon_a(k,q)x^k_{R_p} + \epsilon_b(k,q)x^k_{I_p} \tag{33}$$

$$\frac{\partial \epsilon(k,p)}{\partial s_{I_{pq}}} = \epsilon_a(k,p)(-x^k_{I_q}) + \epsilon_b(k,p)x^k_{R_q} \tag{34}$$

$$\frac{\partial \epsilon(k,q)}{\partial s_{I_{pq}}} = \epsilon_a(k,q)x^k_{I_p} + \epsilon_b(k,q)(-x^k_{R_p}) \tag{35}$$

Based on (28)-(35), it follows:

$$\begin{aligned} & \frac{\partial J}{\partial s_{R_{pq}}} + j\frac{\partial J}{\partial s_{I_{pq}}} \qquad (36)\\ = & \sum_{k=1}^{m}\Big\{\epsilon_a(k,p)\bar{x}^k_q + \epsilon_b(k,p)(x^k_{I_q} + jx^k_{R_q}) \\ & + \epsilon_a(k,q)x^k_p + \epsilon_b(k,q)(x^k_{I_p} - jx^k_{R_p})\Big\} \\ = & \sum_{k=1}^{m}\left\{[\epsilon_a(k,p) + j\epsilon_b(k,p)]\bar{x}^k_q + [\epsilon_a(k,q) - j\epsilon_b(k,q)]x^k_p\right\} \end{aligned}$$

where

$$\epsilon_a(k,i) = \frac{b(k,i)\left\{a(k,i)x^k_{I_i} - b(k,i)x^k_{R_i}\right\}}{\left\{a(k,i)^2 + b(k,i)^2\right\}^{3/2}} \qquad i = p \text{ or } q; \tag{37}$$

and

$$\epsilon_b(k,i) = \frac{a(k,i)\left\{b(k,i)x^k_{R_i} - a(k,i)x^k_{I_i}\right\}}{\left\{a(k,i)^2 + b(k,i)^2\right\}^{3/2}} \qquad i = p \text{ or } q; \tag{38}$$

can be obtained from (24), (31), and (32). Moreover, from (3), (22), and (24),

$$\epsilon(k,i) < 1 - \cos(\pi/K), \qquad \forall k, i \tag{39}$$

can be referred to as a criterion to stop the learning rule (27). Becaus all the training vectors are stored as fixed points of (22) as long a (39) is satisfied.

Finally, we summarize this learning algorithm as follows:

Step 1: Set the initial values of $s_{R_{ij}}$ and $s_{I_{ij}}$ to small random numbers, for all i, j.

Step 2: Compute (24) and check if (39) hold. If so, proceed to step 4; otherwise, go to step 3.

Step 3: For each learning cycle, compute $a(k,i)$, $b(k,i)$, $\epsilon_a(k,i)$, and $\epsilon_b(k,i)$ $\forall$ k,i by (25),(26), (31) and (32), respectively. Then update s_{ij} $(i < j)$ by (27) and go to step 2.

Step 4: Update s_{ij} $(i > j)$ by the relation $s_{ij} = \bar{s}_{ji}$, then stop.

5 Computer Simulations

Experimental results are presented here to demonstrate the capacity and the error correction capability of the CVNNs. Though the CVNNs are shown to be globally stable under different matrix conditions and update modes, in the following simulation it is assumed that S is Hermitian and the system is operated in the asynchronous mode.

5.1 Capacity Test

Capacities of CVNNs with two different encoding methods (i.e., the generalized Hebb rule (GHR), and the modified gradient descent rule (MGDR)) are investigated. The network size is fixed at $N = 20$ and the step size is set to $\eta = 0.5$. We varied the number of training vectors m and the resolution factor K. For a fixed K, the components of the training vectors are assigned to complex random values from the K quantization values in (3) with equal probabilities. For each combination (m, K), there are 100 sets of training vectors and each contains m training vectors. A set is called an event. An event is called a successful event if and only if each vector in a training set is a fixed point by a specific training method. For MGDR, the maximum number of learning cycles is set to 1000. In other words, learning will be terminated after 1000 learning cycles regardless of success. Figure 2 illustrates the percentage storage capacity with two different encoding methods. The improvement obtained by using the MGDR is evident

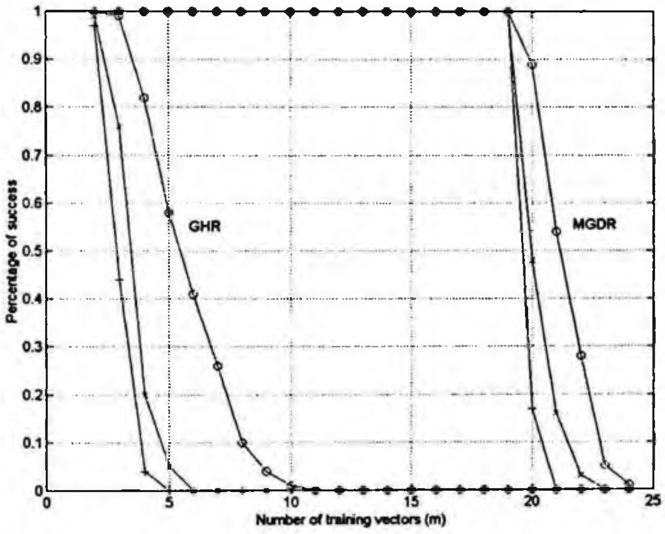


Figure 2. Capacities of the CVNNs with GHR and MGDR (-o-: $K = 4$, -x-: $K = 6$, -+-: $K = 8$).

5.2 Error Correction Capability Test

The error correction capability is also tested once a successful event occurs in the capacity test. The sizes of the training set being studied are $m = 3$, 8, and 16, respectively. After training was carried ou for a set of training vectors, the CVNN was tested with a testing se which contains the noisy versions of the training vectors. The noisy versions of a specific training vector are generated by replacing n components of the original training vector by some random complex values chosen from the K quantization levels. For each combination (m, n, K), 10 independent sets were tested. The percentage of correct recall of CVNN with two different methods at a variety of n were recorded in Figure 3 and Figure 4. The results plotted at a variety of K are also indicated. As seen from Figure 3 and Figure 4, the error correction capability degenerates as K increases. However, the proposed method yields a significant improvement over the Hebb rule training. For example, when $m = 3$ and $K = 4$ the GHR has over 90 percent of success if the number of corrupted components $n \leq 6$. It can be improved to $n \leq 8$ by using the MGDR with the same conditions. Moreover, when $m = 3$ and $n = 5$ the GHR has nearly 92 percent of success if $K = 4$. Under the same conditions, the percent of success can be preserved by using the MGDR even when $K = 8$. These results indicate that the error correction capability of the CVNN is also improved by using the MGDR

6 Conclusion

Using complex representations of the states and the connection weights, respectively, conventional Hopfield networks can be extended to complex-valued neural networks (CVNNs). Since the state variables are represented by complex values on the unit circle of the complex plane, their corresponding phase angles are defined as new variables. This formulation enables the training vectors of CVNNs to permit multivalued components, thereby enhances the design flexibility of the neural networks. The stability conditions of CVNNs under various updating modes have been presented. A learning rule, MGDR, is introduced to enhance the storage performance of CVNN. The weight matrix obtained from MGDR not only guarantees the sta-

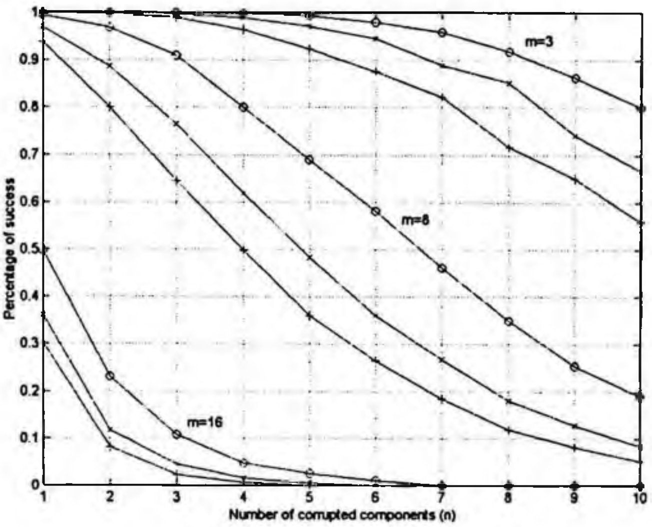


Figure 3. Error correction capability of CVNNs with MGDR (-o-: $K = 4$, -x-: $K = 6$, -+-: $K = 8$).

bility of the designed network, but also reduces the number of spurious memories in state space. Simulation verifies that the capacity of CVNN with MGDR is much larger than that with generalized Hebb rule (GHR). On the other hand, with the aid of the MGDR, the error correction capability of CVNN is also improved. Although the results presented here is based on a Hermitian weight matrix with zero diagonal, a learning rule for non-Hermitian weight matrix with nonzero diagonal can be derived by following a similar way. Moreover, the analysis of CVNNs can also be extended to the heteroassociative memories (Lee and Wang, 1998 and Lee 1998).

Further work in this area is worthwhile. For example, find a learning rule that is capable of shaping and/or expanding the attraction basins of the CVNNs. The analysis would subsequently be more complicated than that in the bipolar network.

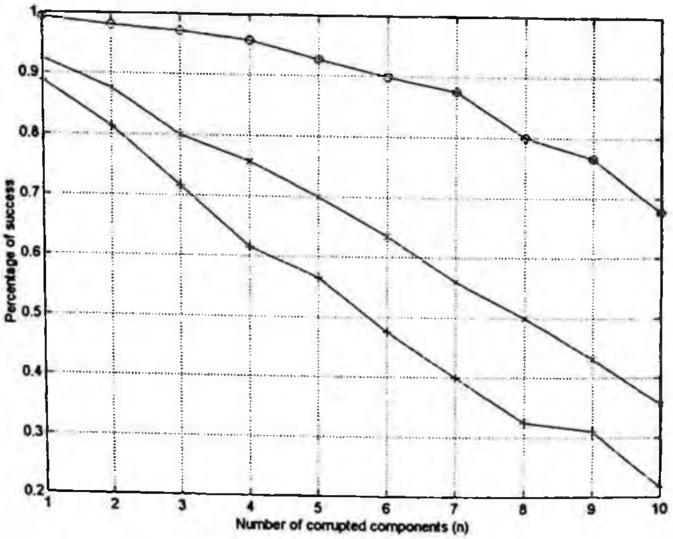


Figure 4. Error correction capability of CVNNs with GHR (-o-: $K = 4$, -x-: $K = 6$, -+-: $K = 8$).

Acknowledgment

Part of the work presented in this chapter was supported by the National Science Council, R.O.C., under Grants NSC 87-2213-E233-002, NSC 89-2218-E-233-001, and NSC 90-2213-E-233-001.

References

[1] Albert, A. (1972), *Regression and the Moore-Penrose pseudoinverse*, Academic Press, New York.

[2] Cernuschi-Frías, B. (1989), "Partial simultaneous updating in

Hopfield memories," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, pp. 887-888.

[3] Chiueh, T. D. and Tsai, H. K. (1993), "Multivalued associative memories based on recurrent networks,"*IEEE Trans. Neural Networks,* vol. 4, no. 2, pp. 364-366.

[4] Erdem, M. H. and Ozturk, Y. (1996), "A new family of multivalued networks," *Neural Networks*, vol. 9, no. 6, pp. 979-989.

[5] Goles, E., Fogelman, F., and Pellegrin, D. (1985), "Decreasing energy functions as a tool for studying threshold networks," *Discrete Applied Mathematics,* vol. 12, pp. 261-277.

[6] Hopfield, J. J. (1982), "Neural networks and physical systems with emergent computational abilities," in *Proc. Nat. Acad. Sci.* USA, vol. 79, pp. 2554-2558.

[7] Hopfield, J. J. (1984), "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proc. Nat. Acad. Sci.* USA, vol. 81, pp. 3088-3092.

[8] Jankowski,.S., Lozowski, A., and Zurada, J. M. (1996), "Complex-valued multistate neural associative memory," *IEEE Trans. Neural Networks,* vol. 7, no. 6, pp. 1491-1496

[9] Lee, D. L. and Wang, W. J (1998), "A Multivalued Bidirectional Associative Memory Operating on a Complex Domain," *Neural Networks*, Vol. 11, No. 9, pp. 1623-1635.

[10] Lee, D. L. (1998), "Generalised Intraconnected Bidirectional Associative Memories," *Electronics Letters*, Vol. 34, No. 8, pp. 736-738.

[11] Lee, D. L. (1999), "New stability conditions for Hopfield networks in partial simultaneous update mode," *IEEE Trans. Neural Networks,* vol. 10, no. 4, pp. 975-978.

[12] Lee, D. L. (2001a), "Improving the capacity of complex-valued neural networks with a modified gradient descent learning rule," *IEEE Trans. Neural Networks,"* 12, no. 2, pp. 439-443.

[13] Lee, D. L. (2001b), "Relaxation of the stability condition of the complex-valued neural networks," *IEEE Trans. Neural Networks,* vol. 12, no. 5, pp. 1260-1262.

[14] Morita, M. (1996), "Memory and learning of sequential patterns by nonmonotone neural networks," *Neural Networks,* vol. 9, no. 8, pp. 1477-1489.

[15] Perfetti, R. (1991), "A neural network to design neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, no. 9, pp. 1099-1103.

[16] Si, J. and Michel, A. N. (1995), "Analysis and synthesis of a class of discrete-time neural networks with multilevel threshold neurons," *IEEE Trans. Neural Networks,* vol. 6, no. 1, pp. 105-116

[17] Wang, C. C., Hwang, S. M., and Lee, J. P. (1996), "Capacity analysis of the asymptotically stable multi-valued exponential bidirectional associative memory, "*IEEE Trans. Syst., Man, Cybern.-part B,* vol. 26, no. 5, pp. 733-743.

[18] Xu, Z. B. and Kwong, C. P. (1995), "Global convergence and asymptotic stability of asymmetric Hopfield networks," *J. Math. Appl.,* vol. 191, pp. 405-427.

[19] Xu, Z. B., Hu, G. Q., and Kwong, C. P. (1996), "Asymmetric Hopfield networks: theory and applications," *Neural Networks,* vol. 9, no. 3, pp. 483-501

[20] Zurada, J. M., Cloete, I., and van der Poel, E. (1997), "Generalized Hopfield networks with multiple stable states," *Neurocomputing*, vol. 13, nos. 2-4.

Author's address

Donq-Liang Lee: Dept. Computer Science and Information Engineering, Van Nung Institute of Technology, Chung Li 32056, Taiwan, R.O.C.

Chapter 4

A Model of Complex-Valued Associative Memories and Its Dynamics

Yasuaki Kuroe

This chapter presents a model of associative memories using complex-valued neural networks and studies its qualitative behavior theoretically. The model is a direct extension of the conventional real-valued associative memories of self-correlation type. One of the most familiar models of associative memories is self-correlation type. We are interested in what will become of the conventional real-valued associative memories when they are directly extended in the complex domain. We investigate the structures and asymptotic behavior of solution orbits near each memory pattern. We also discuss a recalling condition of each memory pattern, that is, a condition which assures that each memory pattern is correctly recalled.

1 Introduction

In recent years, there have been increasing research interests of artificial neural networks and many efforts have been made on applications of neural networks to various fields. As applications of the neural networks spread more widely, developing neural network models which can directly deal with complex numbers is desired in various fields. Several models of complex-valued neural networks have been proposed and their abilities of information processing have been investigated.

One of the most useful and most investigated areas of applications of neural networks addresses implementations of associative memories. Among them associative memories of self-correlation type are easy to implement and have been extensively studied. Some models of complex-valued associative memories of self-correlation type have been proposed (Hashimoto, Kuroe and Mori 2000, Hirose 1992, Kuroe, Hashimoto and Mori 2001a, Kuroe, Hashimoto and Mori 2001b, Nemoto and Kono 1991, Noest 1988).

The purpose of this chapter is to present a model of self-correlation type associative memories using complex-valued neural networks and to investigate its qualitative behavior theoretically (Hashimoto, Kuroe and Mori 2000, Kuroe, Hashimoto and Mori 2001a, Kuroe, Hashimoto and Mori 2001b). The presented model is a direct extension of the real-valued model of the continuous-valued associative memories of synchronous and self-correlation type. We are interested in what will become of the conventional real-valued associative memories when they are directly extended in the complex domain. It will be shown that the model of complex-valued associative memories possesses characteristic features that the conventional real-valued model does not possess. In the model, though each memory vector is an equilibrium point of the network, it is not isolate one, but one of the points belonging to an equilibrium point set which is one-to-one corresponding to the memory vector. These equilibrium sets are considered as the memory patterns in stead of the corresponding vectors. The model of complex-valued associative memories is a nonlinear dynamical system and its qualitative behavior is studied. In particular, we investigate the structures and asymptotic behavior of solution orbits near each memory pattern. Main analysis tool is the center manifold theory. We also discuss a recalling condition, that is, a condition which assures that each memory pattern is correctly recalled.

In the following, the imaginary unit is denoted by i ($i^2 = -1$). The n-dimensional complex (real) space is denoted by $C^n(R^n)$ and the set of $n \times m$ complex (real) matrices is denoted by $C^{n \times m}(R^{n \times m})$. For $A \in C^{n \times m}$ ($a \in C^n$), its real and imaginary parts are denoted by A^R (a^R) and A^I (a^I), respectively.

2 Complex-Valued Associative Memory of Self-Correlation Type

2.1 Model

Let m be the number of memory patterns to be stored and each memory pattern be an N dimensional complex vector, denoted by $s^{(\gamma)} \in C^N$, $\gamma = 1, 2, \cdots, m$. Suppose that the set of memory patterns satisfies the following orthogonal relations.

$$s^{(\gamma)*} s^{(l)} = \begin{cases} N, & \gamma = l \\ 0, & \gamma \neq l \end{cases} \tag{1}$$

$$|s_j^{(\gamma)}| = 1, \quad j = 1, 2, \cdots, N \tag{2}$$

for all $\gamma, l = 1, 2, \cdots, m$ where s^* is the conjugate transpose of s and $s_j^{(\gamma)}$ is the jth element of $s^{(\gamma)}$. Consider a complex-valued neural network described by difference equations of the form:

$$x_j[t+1] = f(\sum_{k=1}^{N} w_{jk} x_k[t]), \qquad j = 1, 2, \cdots, N \tag{3}$$

where $x_j[t] \in C$ is the output of the jth neuron at time t, $w_{jk} \in C$ is the connection weight from the kth neuron to the jth neuron and $f(\cdot)$ is the activation function which is a nonlinear complex function ($f : C \rightarrow C$). Figure 1 shows the schematic diagram of the complex-valued neural network (3).

Let us determine the weight matrix $W = \{w_{jk}\} \in C^{N \times N}$ and the activation function $f(\cdot)$ so that the neural network (3) can store the set of memory vectors and act as an associative memory.

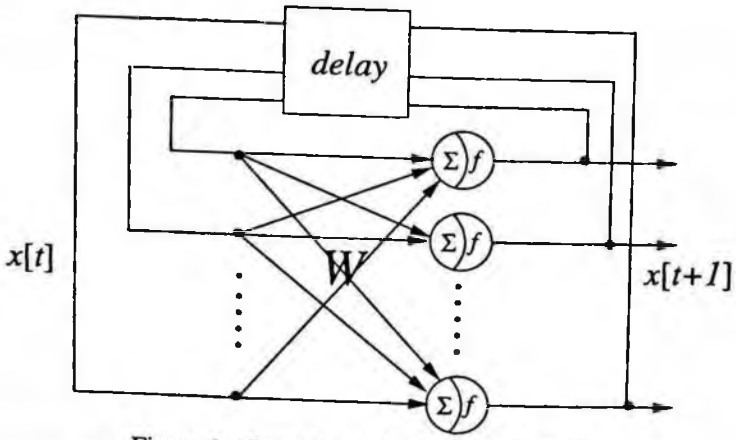


Figure 1. Complex-valued neural network.

The weight matrix W is determined after the model of conventional real-valued associative memories of self-correlation type as follows. Taking account of the orthogonal structure of the set of memory vectors $\{s^{(\gamma)}\}$, we determine the weight matrix W by the sum of the autocorrelation matrix of each memory vector $s^{(\gamma)}$:

$$W = \frac{1}{N}\sum_{\gamma=1}^{m} s^{(\gamma)} s^{(\gamma)*} \tag{4}$$

One of the important factors to characterize behavior of a complex-valued neural network is its activation function $f(\cdot)$. In the real-valued neural networks, the activation is usually chosen to be a smooth and bounded function such as a sigmoidal function. In the complex region, however, there are several possibilities in choosing an activation function because of a variety of complex functions. As a candidate of the activation function $f(\cdot)$, we will choose a complex function which satisfies the conditions:

(i) $f(\cdot)$ is a smooth and bounded function by analogy with the sigmoidal function of real-valued neural networks, and

(ii) each memory vector $s^{(\gamma)}$ becomes an equilibrium point of (3).

The condition (ii) is accomplished by choosing a function which satisfies

$$f(s_j^{(\gamma)}) = s_j^{(\gamma)}, \ \ j = 1, 2, \cdots, N, \ \ \gamma = 1, 2, \cdots, m. \tag{5}$$

In regard to the condition (i), we recall the Liouville's theorem, which says that 'if $f(z)$ is analytic at all $z \in C$ and bounded, then $f(z)$ is a constant function'. Since a suitable $f(z)$ should be bounded, it follows from the theorem that if we choose an analytic function for $f(z)$, it is constant, which is clearly not suitable (Georgiou and Koutsougeras 1992). We choose

$$f(z) = \frac{\eta z}{\eta - 1 + |z|}, \ \ \eta - 1 > 0, \ z \in C \tag{6}$$

as the activation function, where η is a real number satisfying $\eta - 1 > 0$. The function (6) is not analytic, but has the continuous partial derivatives $\partial f^R / \partial z^R$, $\partial f^R / \partial z^I$, $\partial f^I / \partial z^R$ and $\partial f^I / \partial z^I$ and is bounded:($|f(z)| < \eta, \forall z \in C$ ($|z| < \infty$)). Note also that the function (6) satisfies (5), which makes all the memory vectors $s^{(\gamma)}$, $\gamma = 1, 2, \cdots, m$ satisfying (1) and (2) being equilibrium points of the complex-valued neural network (3). Note also that the activation function (6) does not vary the phase of its argument.

2.2 Memory Patterns as Equilibrium Sets

In the model of complex-valued associative memories (3), (4) and (6), each memory vector $s^{(\gamma)}$ to be stored becomes an equilibrium point of the network. But it will be seen that $s^{(\gamma)}$ is not an isolated equilibrium point. This property is not found in the conventional real-valued associative memories. For each memory vector $s^{(\gamma)}$, we consider the vector $e^{i\alpha} s^{(\gamma)}$ where α is a real number ($\alpha \in R$). By using the function (6) we can check that $e^{i\alpha} s^{(\gamma)}$ satisfies the following equation

$$e^{i\alpha}s_j^{(\gamma)} = f(\sum_{k=1}^{N} w_{jk}e^{i\alpha}s_k^{(\gamma)}), \qquad j = 1, 2, \cdots, N \tag{7}$$

for any real number α. This implies that $e^{i\alpha}\boldsymbol{s}^{(\gamma)}$ is also an equilibrium point of the network. Hence each memory vector $\boldsymbol{s}^{(\gamma)}$ to be stored is not an isolated equilibrium point but a point in the equilibrium set defined by

$$\Phi^{(\gamma)} = \{e^{i\alpha}\boldsymbol{s}^{(\gamma)} : \forall\alpha \in \boldsymbol{R}\} \subset \boldsymbol{C}^N \tag{8}$$

which is a closed curve in the complex N dimensional state space $\boldsymbol{C}^N$. Considering this fact, we can treat the model as associative memories as follows. For each memory vector $\boldsymbol{s}^{(\gamma)}$, we identify all the points in the equilibrium set $\Phi^{(\gamma)}$ and regard $\Phi^{(\gamma)}$ as a memory pattern (Figure 2). Hence the model (3), (4) and (6) is an associative memory which has equilibrium sets $\Phi^{(\gamma)}$, $\gamma = 1, 2, \cdots, m$ as memory patterns.

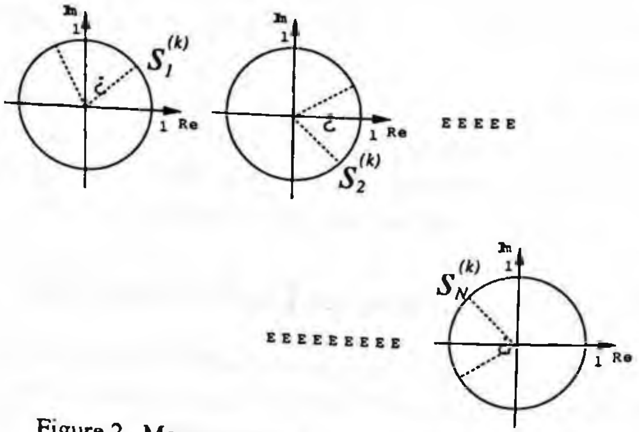


Figure 2. Memory pattern as an equilibrium set $\Phi^{(k)}$.

3 Qualitative Analysis of Behaviors Near Memory Patterns

In this section we will study qualitative behavior of the model of the complex-valued associative memories (3),(4) and (6). Especially, we investigate the structures and asymptotic behavior of solution orbits near each memory pattern $\Phi^{(\gamma)}$.

3.1 Linearized Model Near Memory Patterns

The qualitative behavior of a nonlinear dynamical system near an equilibrium point can be studied via linearization with respect to that point. We will derive the linearized model of the complex-valued neural network (3), (4) and (6) at a point $e^{i\alpha}s^{(\gamma)}$ in each memory pattern $\Phi^{(\gamma)}$. Note that the activation function $f(z)$ given by (6) is not differentiable with respect to z, but its real and imaginary parts, f^R and f^I, are continuously partially differentiable with respect to z^R and z^I. It is, therefore, possible to derive the linearized model by separating the model (3) into its real and imaginary parts.

Choose a real number α arbitrarily and let $q^{(\gamma)} = e^{i\alpha}s^{(\gamma)}$. We separate the model (3) into its real and imaginary parts and linearize them around each equilibrium point $q^{(\gamma)}$. Let $\Delta x_i[t] := x_i[t] - q_i^{(\gamma)}$ and define $y[t] \in R^{2N}$ by

$$y[t] = (\Delta x_1^R[t], \Delta x_2^R[t], \cdots, \Delta x_N^R[t], \Delta x_1^I[t], \Delta x_2^I[t], \cdots, \Delta x_N^I[t])^T \tag{9}$$

where $(\cdot)^T$ denotes the transpose of $(\cdot)$. The linearized model is given by

$$y[t+1] = J(q^{(\gamma)})y[t], \quad \gamma = 1, 2, \cdots, m \tag{10}$$

where $J(q^{(\gamma)}) = F(q^{(\gamma)})Y \in R^{2N\times 2N}$. The matrices $F(q^{(\gamma)}) \in R^{2N\times 2N}$ and $Y \in R^{2N\times 2N}$ are given as follows.

$$F(q^{(\gamma)}) = \begin{bmatrix} F_{RR} & F_{RI} \\ F_{IR} & F_{II} \end{bmatrix} \qquad Y = \begin{bmatrix} W^R & -W^I \\ W^I & W^R \end{bmatrix}$$

where

$$\begin{aligned}
F_{RR} &= \operatorname{diag}\left(\left.\frac{\partial f^R}{\partial z^R}\right|_{z=q_1^{(\gamma)}}, \left.\frac{\partial f^R}{\partial z^R}\right|_{z=q_2^{(\gamma)}}, \cdots, \left.\frac{\partial f^R}{\partial z^R}\right|_{z=q_N^{(\gamma)}}\right) \\
F_{RI} &= \operatorname{diag}\left(\left.\frac{\partial f^R}{\partial z^I}\right|_{z=q_1^{(\gamma)}}, \left.\frac{\partial f^R}{\partial z^I}\right|_{z=q_2^{(\gamma)}}, \cdots, \left.\frac{\partial f^R}{\partial z^I}\right|_{z=q_N^{(\gamma)}}\right) \\
F_{IR} &= \operatorname{diag}\left(\left.\frac{\partial f^I}{\partial z^R}\right|_{z=q_1^{(\gamma)}}, \left.\frac{\partial f^I}{\partial z^R}\right|_{z=q_2^{(\gamma)}}, \cdots, \left.\frac{\partial f^I}{\partial z^R}\right|_{z=q_N^{(\gamma)}}\right) \\
F_{II} &= \operatorname{diag}\left(\left.\frac{\partial f^I}{\partial z^I}\right|_{z=q_1^{(\gamma)}}, \left.\frac{\partial f^I}{\partial z^I}\right|_{z=q_2^{(\gamma)}}, \cdots, \left.\frac{\partial f^I}{\partial z^I}\right|_{z=q_N^{(\gamma)}}\right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial f^R}{\partial z^R} &= \eta\frac{(\eta-1)|z| + {z^I}^2}{|z|(\eta-1+|z|)^2}, \\
\frac{\partial f^I}{\partial z^I} &= \eta\frac{(\eta-1)|z| + {z^R}^2}{|z|(\eta-1+|z|)^2} \\
\frac{\partial f^R}{\partial z^I} &= \frac{\partial f^I}{\partial z^R} = \eta\frac{-z^R z^I}{|z|(\eta-1+|z|)^2}.
\end{aligned}$$

3.2 Structure of Solution Orbits Near Memory Patterns

We now analyze the qualitative behavior and structure of the solution orbits near each memory pattern. This can be done by investigating the eigenvalues and eigenvectors of the coefficient matrix $J(\boldsymbol{q}^{(\gamma)})$ of the linearized model (10). The following theorem holds.

Theorem 1 *For an arbitrary real number β, the matrices $J(\boldsymbol{q}^{(\gamma)})$ and $J(e^{i\beta}\boldsymbol{q}^{(\gamma)})$ are orthogonally similar, that is, there exists an orthogonal matrix T such that*

$$J(e^{i\beta}\boldsymbol{q}^{(\gamma)}) = TJ(\boldsymbol{q}^{(\gamma)})T^{-1}. \tag{11}$$

Proof. Choose the orthogonal matrix T as

$$T = T(\beta) = \begin{bmatrix} I\cos(\beta) & -I\sin(\beta) \\ I\sin(\beta) & I\cos(\beta) \end{bmatrix} \tag{12}$$

where I is the unit matrix. It is easy to check that (11) holds for the matrix T. □

The theorem implies that all the eigenvalues of the matrix $J(\boldsymbol{q}^{(\gamma)})$ are the same for all point $\boldsymbol{q}^{(\gamma)} = e^{i\alpha}\boldsymbol{s}^{(\gamma)} \in \Phi^{(\gamma)}$ and the corresponding eigenvectors relate with each other by the orthogonal matrix. Furthermore each matrix $J(\boldsymbol{q}^{(\gamma)})$ has the following characteristic properties with respect to its eigenstructure.

Theorem 2 *Let $\lambda_i(J(\boldsymbol{q}^{(\gamma)}))$ be the ith eigenvalue of $J(\boldsymbol{q}^{(\gamma)})$. $\lambda_i(J(\boldsymbol{q}^{(\gamma)}))$, $i = 1, 2, \cdots, 2N$ are all real and less than or equal to one:*

$$|\lambda_i(J(\boldsymbol{q}^{(\gamma)}))| \leq 1, \quad i = 1, 2, \cdots, 2N \tag{13}$$

for all $\gamma = 1, 2, \cdots, m$.

Theorem 3 *The matrix $J(\boldsymbol{q}^{(\gamma)})$ has at least one eigenvalue 1 and at least one eigenvalue $(\eta - 1)/\eta$. It also has $2(N - m)$ eigenvalues 0. The corresponding eigenvector to the eigenvalue 1 is*

$$\boldsymbol{p} = ((\{i\boldsymbol{q}^{(\gamma)}\}^R)^T, (\{i\boldsymbol{q}^{(\gamma)}\}^I)^T)^T \tag{14}$$

and that to the eigenvalue $(\eta - 1)/\eta$ is

$$\boldsymbol{r} = ((\boldsymbol{q}^{(\gamma)R})^T, (\boldsymbol{q}^{(\gamma)I})^T)^T. \tag{15}$$

The proofs of Theorems 2 and 3 will be given in Appendix. Note that $0 < (\eta - 1)/\eta < 1$ because $\eta > 1$. It is important to note that Theorems 1, 2 and 3 hold for all $\Phi^{(\gamma)}$, $\gamma = 1, 2, \cdots, m$. Then the associative memory (3), (4) and (6) is homogeneous in this sense.

It is known that qualitative behavior of solutions near an equilibrium point of a nonlinear dynamical system is determined by its linearized model at the point if it is hyperbolic (in a discrete time system, the coefficient matrix of the linearized model has no eigenvalues of unit modulus) (Sastry 1999). From Theorem 3, however, all the equilibrium points $\boldsymbol{q}^{(\gamma)}$, $\gamma = 1, 2, \cdots, m$ are not hyperbolic. We introduce the center manifold theory in order to investigate qualitative behavior of solutions near the equilibrium points $\boldsymbol{q}^{(\gamma)}$, $\gamma = 1, 2, \cdots, m$.

Center Manifold Theory (Sastry 1999) *Let g be a C^r map with fixed point at $\bar{\boldsymbol{x}}$. That is, $\bar{\boldsymbol{x}} = g(\bar{\boldsymbol{x}})$ and define $Dg(\bar{\boldsymbol{x}}) := \partial g(\bar{\boldsymbol{x}})/\partial \boldsymbol{x}$. Let σ^s, σ^u, σ^c be the disjoint partition of the spectrum of $Dg(\bar{\boldsymbol{x}})$ with associated generalized eigenspaces E^s, E^u, E^c, corresponding to eigenvalues inside the open unit circle, outside the open unit circle, and on the unit circle. Then there exist C^r stable and unstable invariant manifolds W^s, W^u tangent to E^s, E^u at $\bar{\boldsymbol{x}}$ and a C^{r-1} center manifold W^c tangent to E^c at $\bar{\boldsymbol{x}}$.*

It follows from the center manifold theory that, from Theorems 2, each point $\boldsymbol{q}^{(\gamma)}$ on each memory pattern $\Phi^{(\gamma)}$ has no unstable invariant manifold W^u, and from Theorem 3, it has stable and center manifolds; W^s and W^c. Let c be the number of eigenvalues 1 which $J(\boldsymbol{q}^{(\gamma)})$ has. Then there exist $2N - c$ dimensional stable invariant manifold W^s and c dimensional center manifold W^c near the equilibrium point $\boldsymbol{q}^{(\gamma)}$. If $c = 1$, that is, $J(\boldsymbol{q}^{(\gamma)})$ has only one eigenvalue 1, the following theorems hold.

Theorem 4 *Consider the linearized model (10). If $J(\boldsymbol{q}^{(\gamma)})$ has only one eigenvalue on the unit circle, the corresponding eigenspace denoted by $E^{c(\gamma)}$ is given by*

$$E^{c(\gamma)} = \{a\boldsymbol{p} : \forall a (\in \boldsymbol{R})\} \subset \boldsymbol{R}^{2N} \tag{16}$$

and $E^{c(\gamma)}$ is stable, that is, every solution $\boldsymbol{y}[t]$ of the linearized model (10) approaches $E^{c(\gamma)}$ as $t \to \infty$.

Proof. From Theorem 3, $\boldsymbol{p}$ is the eigenvector associated with the eigenvalue 1. The stability of $E^{c(\gamma)}$ comes from the fact that the magnitude of all the other eigenvalues are less than one (Theorem 2). □

Consider the neural network (3), (4) and (6) a $2N$ dimensional real dynamical system by separating it into its real and imaginary parts. Denote the expressions of the equilibrium point $q^{(\gamma)}$ and the memory pattern $\Phi^{(\gamma)}$ in the $2N$ dimensional real space by $q^{(\gamma)2N}$ and $\Phi^{(\gamma)2N}$, respectively. They are given by

$$q^{(\gamma)2N} = ((q^{(\gamma)R})^T, (q^{(\gamma)I})^T)^T$$

and

$$\Phi^{(\gamma)2N} = \{T(\beta)\begin{pmatrix} q^{(\gamma)R} \\ q^{(\gamma)I} \end{pmatrix} : \forall\beta \in \boldsymbol{R}\} \subset \boldsymbol{R}^{2N} \tag{17}$$

where $T(\beta)$ is defined in (12).

Theorem 5 *If $J(q^{(\gamma)})$ has only one eigenvalue on the unit circle, the equilibrium point $q^{(\gamma)2N}$ has one dimensional center manifold, denoted by $W^{c(\gamma)}$, and $2N-1$ dimensional stable invariant manifold, denoted by $W^{s(\gamma)}$, and $W^{c(\gamma)} = \Phi^{(\gamma)2N}$.*

Proof. The first half of the theorem is obvious. We will show $W^{c(\gamma)} = \Phi^{(\gamma)2N}$. Note that $\Phi^{(\gamma)2N}$ is invariant for the dynamical system (3), (4) and (6) because it is its equilibrium set. Calculating the gradient of $\Phi^{(\gamma)2N}$ with respect to β at $\beta = 0$, we obtain

$$\left.\frac{d\Phi^{(\gamma)2N}}{d\beta}\right|_{\beta=0} = \begin{pmatrix} -q^{(\gamma)I} \\ q^{(\gamma)R} \end{pmatrix} = \boldsymbol{p}. \tag{18}$$

Then $\Phi^{(\gamma)2N}$ is tangent to $E^{c(\gamma)}$ in Theorem 4 at $\beta = 0$. This completes the proof. □

Theorem 5 implies that if $J(q^{(\gamma)})$ has only one eigenvalue on the unit circle, the memory pattern $\Phi^{(\gamma)2N}$ itself is the center manifold of the equilibrium point $q^{(\gamma)2N} \in \Phi^{(\gamma)2N}$ (Figure 3). Note that this theorem holds for all points in the memory pattern $\Phi^{(\gamma)2N}$ if it holds for any one point $q^{(\gamma)2N}$ in $\Phi^{(\gamma)2N}$ since all the eigenvalues of $J(q^{(\gamma)})$ are the same for all the points $q^{(\gamma)} \in \Phi^{(\gamma)}$ (from Theorem 1).

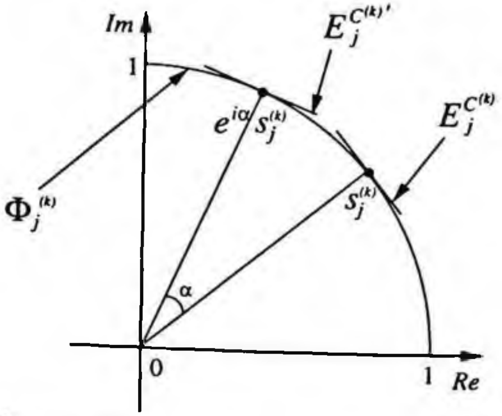


Figure 3. Memory pattern and center manifold

4 Discussions

4.1 Recalling Condition for Memory Patterns

We now discuss the recalling condition of each memory pattern of the model of complex valued associative memories (3), (4) and (6). In the previous section we have shown that, if $J(q^{(\gamma)})$ has only one eigenvalue on the unit circle for a point $q^{(\gamma)}$ in a memory pattern $\Phi^{(\gamma)}$, the structure of solution orbits near each point in $\Phi^{(\gamma)2N}$ consist of $2N-1$ dimensional stable invariant manifold and one dimensional center manifold. Furthermore the memory pattern $\Phi^{(\gamma)2N}$ itself is the center manifold which is tangent to $E^{c(\gamma)}$ given by (16) in Theorem 4. It can be concluded, therefore, that '$J(q^{(\gamma)})$ has only one eigenvalue on the unit circle' is a condition for each memory pattern $\Phi^{(\gamma)}$ to be correctly recalled, that is, every solution starting in the neighborhood of $\Phi^{(\gamma)}$ approaches $\Phi^{(\gamma)}$ as $t \to \infty$. Then we can determine if each memory pattern is correctly recalled or not by evaluating the eigenvalues of $J(q^{(\gamma)})$. In order to verify this, we have carried out numerical experiments. Note that for each memory pattern $\Phi^{(\gamma)}$ it is enough to evaluate the eigenvalues of $J(q^{(\gamma)})$ at any one point in $\Phi^{(\gamma)}$.

Table 1. Orthogonal vectors $s^{(\gamma)}$, $\gamma = 1, 2, \cdots, 6$.

	$s^{(1)}$	$s^{(2)}$	$s^{(3)}$
$s_1^{(\gamma)}$	$1.000 + i0.000$	$1.000 + i0.000$	$1.000 + i0.000$
$s_2^{(\gamma)}$	$-0.021 - i0.999$	$-0.959 - i0.282$	$-0.294 + i0.955$
$s_3^{(\gamma)}$	$-0.659 - i0.752$	$0.937 + i0.347$	$0.600 - i0.799$
$s_4^{(\gamma)}$	$0.127 - i0.991$	$-0.497 + i0.867$	$-0.164 + i0.986$
$s_5^{(\gamma)}$	$-0.844 + i0.535$	$-0.766 - i0.642$	$0.525 + i0.850$
$s_6^{(\gamma)}$	$-0.541 - i0.840$	$0.822 + i0.569$	$-0.800 - i0.599$
$s_7^{(\gamma)}$	$0.440 + i0.897$	$0.440 + i0.897$	$-0.944 - i0.328$
$s_8^{(\gamma)}$	$0.820 - i0.570$	$-0.993 + i0.110$	$-0.313 - i0.949$
$s_9^{(\gamma)}$	$0.444 - i0.895$	$0.444 - i0.895$	$0.971 + i0.238$

	$s^{(4)}$	$s^{(5)}$	$s^{(6)}$
$s_1^{(\gamma)}$	$1.000 + i0.000$	$1.000 + i0.000$	$1.000 + i0.000$
$s_2^{(\gamma)}$	$0.997 - i0.071$	$0.086 + i0.996$	$0.961 - i0.274$
$s_3^{(\gamma)}$	$-0.513 + i0.858$	$-0.886 + i0.461$	$0.059 - i0.998$
$s_4^{(\gamma)}$	$-0.014 + i0.999$	$-0.731 - i0.682$	$-0.796 - i0.605$
$s_5^{(\gamma)}$	$-0.453 + i0.891$	$0.807 + i0.590$	$0.425 - i0.904$
$s_6^{(\gamma)}$	$0.549 + i0.835$	$0.941 - i0.335$	$0.903 - i0.428$
$s_7^{(\gamma)}$	$-0.983 - i0.182$	$0.347 + i0.937$	$-0.026 - i0.999$
$s_8^{(\gamma)}$	$0.998 - i0.050$	$-0.128 + i0.991$	$-0.785 - i0.619$
$s_9^{(\gamma)}$	$-0.825 - i0.564$	$0.500 + i0.865$	$-0.986 - i0.164$

4.1.1 Numerical Experiment 1

Let the dimension of the neural network be $N = 9$ and let $\eta = 2$ in the nonlinear function (6). Six orthogonal vectors $s^{(1)}, s^{(2)}, \cdots, s^{(6)}$ shown in Table 1 are specified. First we let $m = 5$ and choose the vectors $s^{(1)}, s^{(2)}, \cdots, s^{(5)}$ and construct the network (3), (4) and (6). We evaluate the eigenvalues of $J(q^{(\gamma)})$ of a point in each memory pattern $\Phi^{(\gamma)}$, $\gamma = 1, 2, \cdots, 5$. The result is shown in Table 2.

It turns out that each $J(q^{(\gamma)})$ has one eigenvalue 1, one eigenvalue

Table 2. Eigenvalues of $J(q^{(\gamma)})$, $\gamma = 1, 2, \cdots, 5$.

$\lambda_j(J(q^{(\gamma)}))$	1	0.5	0	inside the unit disk
the number of eigenvalues	1	1	8	8

0.5, eight eigenvalues 0 and eight eigenvalues inside the open unit circle. Since each $J(q^{(\gamma)})$ has only one eigenvalue 1, it can be determined that all the 5 memory patterns are correctly recalled. To verify this, the recalling test has been performed. We ran the network (3) starting from the various initial conditions $x[0]$ in the neighborhood of each memory pattern. Figures 4 and 5 show examples of the obtained recalling process. In Fig. 4, the directional cosine $c[t]$ between $\Phi^{(\gamma)}$ and the solution $x[t]$ of the network (3) versus time t is plotted for $\gamma = 1$. The directional cosine $c[t]$ is defined by

$$c[t] = \frac{||\Phi^{(\gamma)*}x[t]||}{(||\Phi^{(\gamma)}||\ ||x[t]||)}. \tag{19}$$

In Fig 5, the quasi-distance $d[t]$ between them, defined by

$$d[t] = \left|\left|\frac{x[t]}{x_1[t]} - s^{(\gamma)}\right|\right|, \tag{20}$$

versus time t is plotted for $\gamma = 1$. $c[t] \to 1$ and $d[t] \to 0$ imply that the memory pattern is correctly recalled. It is seen from these figures that, if we choose the initial condition $x[0]$ close enough to the memory pattern $\Phi^{(1)}$, the solution $x[t]$ approaches $\Phi^{(1)}$, that is, the memory pattern $\Phi^{(1)}$ is correctly recalled. In the similar manner, we have checked that all the other memory patterns are correctly recalled.

4.1.2 Numerical Experiment 2

Next, we increase the number of the vectors to be stored to $m = 6$ and choose $s^{(1)}, s^{(2)}, \cdots, s^{(6)}$ in Table 1, and construct the network (3), (4) and (6). The eigenvalues of $J(q^{(\gamma)})$ of a point in each memory pattern $\Phi^{(\gamma)}$, $\gamma = 1, 2, \cdots, 6$ are evaluated as shown in Table 3.

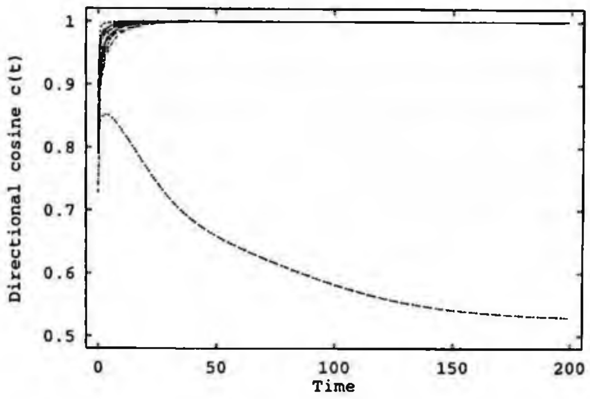


Figure 4. Time evolution of directional cosine $c[t]$ between $\Phi^{(\gamma)}$ and $x[t]$ $(N = 9, m = 5)$.

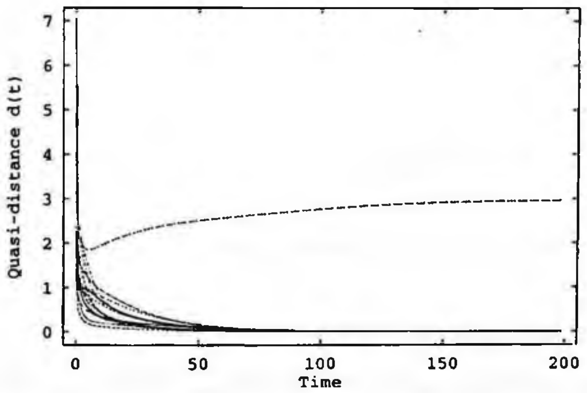


Figure 5. Time evolution of quasi-distance $d[t]$ between $\Phi^{(\gamma)}$ and $x[t]$ $(N = 9, m = 5)$.

Table 3. Eigenvalues of $J(q^{(\gamma)})$, $\gamma = 1, 2, \cdots, 6$.

$\lambda_j(J(q^{(\gamma)}))$	1	0.5	0	inside the unit disk
the number of eigenvalues	3	1	6	8

In this case all $J(q^{(\gamma)})$, $\gamma = 1, 2, \cdots, 6$ have three eigenvalues 1. Therefore we cannot determine if each memory pattern is correctly recalled. We have performed the recalling test and examples of the results for the memory pattern $\Phi^{(1)}$ are shown in Figs. 6 and 7. It is seen from these figures that, no matter how close we choose the initial condition $x[0]$ to the memory pattern $\Phi^{(1)}$, the solution $x[t]$ does not approach the memory pattern $\Phi^{(1)}$, that is, the memory pattern is not recalled. These experimental results support the recalling condition.

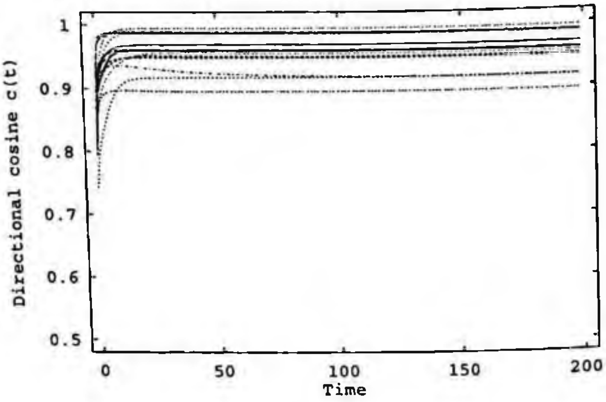


Figure 6. Time evolution of directional cosine $c[t]$ between $\Phi^{(\gamma)}$ and $x[t]$ $(N = 9, m = 6)$.

4.2 Comments on Real-Valued Version of the Model

Here we give some comments on the characteristic of the real-valued version of the proposed model of complex-valued associative memories. As stated in Section 2, a set of the vectors to be stored $s^{(\gamma)} \in C^N$, $\gamma = 1, 2, \cdots, m$ are specified so as to satisfy the condition (1). and (2). There is a difference between the complex domain and the real domain in choosing such a set of vectors. We will consider the relation between the maximum number m^* of the vectors

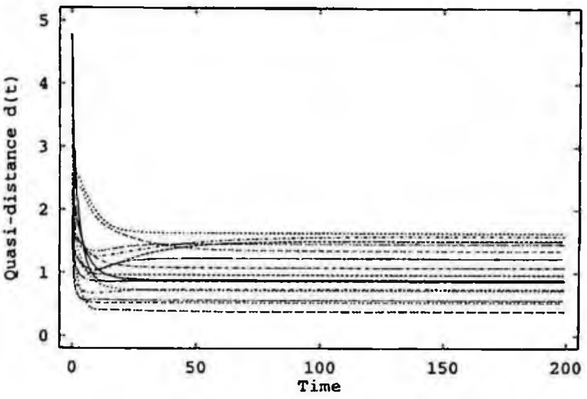


Figure 7. Time evolution of quasi-distance $d[t]$ between $\Phi^{(\gamma)}$ and $x[t]$ ($N = 9, m = 6$).

satisfying (1) and (2) and their dimension N. It can be shown that, in the complex domain there always exist N vectors which satisfy (1) and (2) in any N dimensional space, that is, $m^* = N$. Hence in the complex domain we can always choose N vectors satisfying (1) and (2) in the space of any dimension N. On the other hand, in the real domain such N vectors do not always exist, that is, $m^* \leq N$, and $m^* = N$ holds only for $N = 2^n$ where n is a natural number.

We consider the real-valued version of the model of complex-valued associative memories (3), (4) and (6). Suppose that all the memory vectors satisfying (1) and (2) are real, $s^{(\gamma)} \in \boldsymbol{R}^N$, $\gamma = 1, 2, \cdots, m$, and so is the weight matrix $W = \{w_{jk}\} \in \boldsymbol{R}^{N \times N}$ in (4). Suppose also that, in the dynamics of the neural network (3), $x_i[t]$ is real and $f(\cdot)$ defined by (6) is a real function ($f : \boldsymbol{R} \to \boldsymbol{R}$). Note that, in this real version of the associative memory, each memory vector $s^{(\gamma)}$ is an isolated equilibrium point of the network. Linearizing the real version of the associative memory (3) at each equilibrium point $s^{(\gamma)}$, we have

$$\Delta x[t+1] = \frac{\eta - 1}{\eta} W \Delta x[t] \tag{21}$$

where $\Delta\boldsymbol{x}[t] = \boldsymbol{x}[t] - \boldsymbol{s}^{(\gamma)}$. Note that the coefficient matrix $((\eta - 1)/\eta)W$ is independent of $\boldsymbol{s}^{(\gamma)}$ and the linearized models are the same for all the memory vectors $\boldsymbol{s}^{(\gamma)}$, $\gamma = 1, 2, \cdots, m$. It can be easily seen from (4) that the coefficient matrix $((\eta - 1)/\eta)W$ has m eigenvalues $(\eta - 1)/\eta$ and $m^* - m$ eigenvalues 0. If the dimension of the network is $N = 2^n$, all the eigenvalues of the coefficient matrix $((\eta - 1)/(\eta))W$ can be found; it has m eigenvalues $(\eta - 1)/\eta$ and $N - m$ eigenvalues 0. Note that $0 < (\eta - 1)/\eta < 1$ because $\eta > 1$. Therefore, in the real version of the associative memory, if its dimension is $N = 2^n$, it can store all the m memory vectors as stable equilibrium points and the maximum number of vectors which the network can store is N, that is to say, its storage capacity is N.

4.3 Continuous Complex-Valued Associative Memories

Consider a model of associative memories using a complex-valued continuous-time neural network described by differential equations of the form:

$$\begin{cases} \tau \dfrac{du_j(t)}{dt} = -u_j(t) + \displaystyle\sum_{k=1}^{N} w_{jk} x_k(t) \\ x_j(t) = f(u_j(t)) \end{cases}, \quad j = 1, \cdots, N \qquad (22)$$

where weight matrix $W = \{w_{jk}\} \in \boldsymbol{C}^{N\times N}$ is given by (4) and the activation function $f(\cdot)$ is given by (6) and $\tau \in \boldsymbol{R}$ is the time constant ($\tau > 0$). For the model of associative memories (22) (4) and (6), which we call continuous complex-valued associative memory, similar analysis can be done. In the model of associative memories (22) (4) and (6), each memory vector $\boldsymbol{s}^{(\gamma)}$ to be stored also is not an isolated equilibrium point but a point in the equilibrium set defined by (8). We can investigate the structures and asymptotic behavior of solution orbits near each memory pattern and obtain the results which are parallel to the ones of the discrete-time model (3) (4) and

(6). For example, in the continuous complex-valued (22) (4) and (6), the eigenvalues of the coefficient matrix of the linearized model are all real and all less than or equal to zero. The coefficient matrix of the linearized model has at least one eigenvalue 0, at least one eigenvalue $-1/(\tau\eta)$ and $2(N-m)$ eigenvalues $-1/\tau$. If the coefficient matrix of the linearized model has only one eigenvalue 0 for a point $q^{(\gamma)}$ in a memory pattern $\Phi^{(\gamma)}$, then the structure of solution orbits near each point in $\Phi^{(\gamma)_{2N}}$ consists of $2N-1$ dimensional stable invariant manifold and one dimensional center manifold. Furthermore the memory pattern $\Phi^{(\gamma)_{2N}}$ itself is the center manifold. For details, see the literature (Kuroe, Hashimoto and Mori 2001b).

5 Conclusion

In this chapter we presented a model of associative memories using complex-valued neural networks and studied its qualitative behavior theoretically. The model is a direct extension of the conventional real-valued associative memories of self-correlation type. In the model, each memory pattern is not an isolated equilibrium point but an equilibrium set of the network, which is a closed curve in the complex state space. We investigated the eigenstructures and asymptotic behavior of solution orbits near each memory pattern by linearizing the proposed model at a point of each memory pattern. A recalling condition of each memory pattern, that is, a condition assures that each memory pattern is correctly recalled was also discussed. Some comments on the characteristic of the real-valued version of the proposed model were also given.

The contribution of this chapter is based on the studies (Hashimoto, Kuroe and Mori 2000, Kuroe, Hashimoto and Mori 2001a, Kuroe, Hashimoto and Mori 2001b) The author would like to express his gratitude to Prof. T. Mori and Dr. N. Hashimoto for their valuable and continuous discussions.

References

Georgiou, G.M. and Koutsougeras, C. (1992), " Complex domain backpropagation," *IEEE Trans. on Circuits and Systems*, vol. 39, no. 5, pp. 330–334.

Hashimoto, N., Kuroe, Y. and Mori, T. (2000), " Theoretical study of qualitative behavior of self-correlation type associative memory on complex-valued neural networks," *Trans. of IEICE*, vol.J83-A, no.6, pp.750-760 (in Japanese).

Hirose, A. (1992), " Dynamics of fully complex-valued neural networks," *Electronics Letters*, vol.28, no.16, pp.1492-1494.

Kuroe, Y., Hashimoto, N. and Mori, T. (2001a), " Qualitative analysis of a self-correlation type complex-valued associative memories," *Nonlinear Analysis*, vol.47, pp.5795-5806.

Kuroe, Y., Hashimoto, N. and Mori, T. (2001b), " Qualitative Analysis of Continuous Complex-Valued Associative Memories," *Artificial Neural Networks - ICANN 2001, George Dorffner et. al. (Eds.), Lecture Notes in Computer Science*, 2130, pp.843 -850, Springer.

Nemoto, I. and Kono, T. (1991), " Complex-valued neural network," *The Trans. of IEICE*, vol. J74-D-II, no. 9, pp. 1282-1288 (in Japanese).

Noest, A. J. (1988), " Phaser neural network," *Neural Informaion Processing Systems*, D.Z.Anderson,ed., pp. 584–591, AIP, New York.

Sastry, S. (1999), *Nonlinear Systems Analysis, Stability, and Control*, Springer-Verlag, New York.

A Proofs of Theorems 2 and 3

A.1 Proof of Theorem 2

To prove this theorem the following lemmas are needed.

Lemma 1 *Let A and $B \in C^{n\times n}$ be Hermitian matrices. If one of the two matrix is positive or negative definite, all the eigenvalues of AB are real.*

Lemma 2 *Let A and $B \in C^{n\times n}$ be Hermitian matrices. If $|\lambda_j(A)| \leq 1$ and $|\lambda_j(B)| \leq 1,\ \forall j = 1, 2, \cdots, n$, then $|\lambda_j(AB)| \leq 1,\ \forall j = 1, 2, \cdots, n$.*

Lemma 3 *Let $A = A^R + iA^I \in C^{n\times n}$ ($A^R \in R^{n\times n}, A^I \in R^{n\times n}$) be a Hermitian matrix and define*

$$B = \begin{bmatrix} A^R & -A^I \\ A^I & A^R \end{bmatrix} \in R^{2n\times 2n}. \tag{23}$$

If λ is an eigenvalue of A, then B has two eigenvalues λ.

The proofs of these lemmas are omitted.

It is obvious that $F(q^{(\gamma)})$ is a Hermitian (symmetric) matrix. Y is also a Hermitian (symmetric) matrix because $W^* = W$ ($\{W^R\}^T = W^R, \{W^I\}^T = -W^I$). First we show that $F(q^{(\gamma)})$ is positive definite and $|\lambda_j(F(q^{(\gamma)}))| \leq 1,\ \forall j = 1, 2, \cdots, 2N$. $\det(\lambda I - F(q^{(\gamma)}))$ can be calculated as follows.

$$\begin{aligned}
\det(\lambda I &- F(q^{(\gamma)})) \\
&= \det(\lambda I - F_{RR}) \times \\
&\quad \det((\lambda I - F_{II}) - (F_{IR})(\lambda I - F_{RR})^{-1}(F_{RI})) \\
&= \left.\left\{\left(\lambda - \frac{\partial f^R}{\partial z^R}\right)\left(\lambda - \frac{\partial f^I}{\partial z^I}\right) - \frac{\partial f^I}{\partial z^R}\frac{\partial f^R}{\partial z^I}\right\}\right|_{z=q_1^{(\gamma)}} \\
&\quad \cdots \left.\left\{\left(\lambda - \frac{\partial f^R}{\partial z^R}\right)\left(\lambda - \frac{\partial f^I}{\partial z^I}\right) - \frac{\partial f^I}{\partial z^R}\frac{\partial f^R}{\partial z^I}\right\}\right|_{z=q_N^{(\gamma)}}
\end{aligned}$$

$$
\begin{aligned}
&= \left(\lambda^2 - \frac{2\eta - 1}{\eta}\lambda + \frac{\eta - 1}{\eta}\right)^N \\
&= (\lambda - 1)^N \left(\lambda - \frac{\eta - 1}{\eta}\right)^N \qquad (24)
\end{aligned}
$$

Therefore $F(\boldsymbol{q}^{(\gamma)})$ has N eigenvalues 1 and N eigenvalues $(\eta - 1)/\eta$, which implies $F(\boldsymbol{q}^{(\gamma)}) > 0$ and $|\lambda_j(F(\boldsymbol{q}^{(\gamma)}))| \leq 1$ because $0 < (\eta - 1)/\eta < 1$.

Next, we show $|\lambda_j(Y)| \leq 1$. From (4), $W\boldsymbol{s}^{(\gamma)} = \boldsymbol{s}^{(\gamma)}$, $\gamma = 1, 2, \cdots, m$. This implies that W has m eigenvalues 1. It can be shown that, in the N dimensional complex space, there always exist $N - m$ vectors $\boldsymbol{e}^{(l)} = (e_1^{(l)}, e_1^{(l)}, \cdots, e_N^{(l)})^T \in C^N$, $l = 1, 2, \cdots, N - m$ which satisfy

$$
\boldsymbol{s}^{(\gamma)*}\boldsymbol{e}^{(l)} = 0 \text{ and } |e_j^{(l)}| = 1 \ \ (\gamma, l = 1, 2, \cdots, m, \ j = 1, 2, \cdots, N). \qquad (25)
$$

From (4), $W\boldsymbol{e}^{(l)} = 0$, $l = 1, 2, \cdots, N - m$. This implies that W has $N - m$ eigenvalues 0. From this and Lemma 3, Y has $2m$ eigenvalues 1 and $2(N - m)$ eigenvalues 0, which gives $|\lambda_j(Y)| \leq 1$. Then $F(\boldsymbol{q}^{(\gamma)})$ and Y satisfy the conditions of Lemma 1 and 2, which completes the proof. □

A.2 Proof of Theorem 3

Straight calculation of $J(\boldsymbol{q}^{(\gamma)})\boldsymbol{p}$ gives

$$
J(\boldsymbol{q}^{(\gamma)})\boldsymbol{p} = F(\boldsymbol{q}^{(\gamma)})Y\boldsymbol{p} = F(\boldsymbol{q}^{(\gamma)})\boldsymbol{p} = \boldsymbol{p}.
$$

Thus $J(\boldsymbol{q}^{(\gamma)})$ has an eigenvalue 1 and its corresponding eigenvector is $\boldsymbol{p}$. Similarly we can obtain

$$
J(\boldsymbol{q}^{(\gamma)})\boldsymbol{r} = \frac{\eta - 1}{\eta}\boldsymbol{r}.
$$

Define the vectors $\boldsymbol{g}^{(l)}$ and $\boldsymbol{h}^{(l)}$ by

$$\boldsymbol{g}^{(l)} = ((\boldsymbol{e}^{(l)R})^T, (\boldsymbol{e}^{(l)I})^T)^T \in \boldsymbol{R}^{2N}$$

and

$$\boldsymbol{h}^{(l)} = ((\{i\boldsymbol{e}^{(l)}\}^R)^T, (\{i\boldsymbol{e}^{(l)}\}^I)^T)^T \in \boldsymbol{R}^{2N}$$

where $\boldsymbol{e}^{(l)R}$ and $\boldsymbol{e}^{(l)I}$ are the real and imaginary parts of $\boldsymbol{e}^{(l)}$ in (25), respectively, and $\{i\boldsymbol{e}^{(l)}\}^R$ and $\{i\boldsymbol{e}^{(l)}\}^I$ are the real and imaginary parts of $i\boldsymbol{e}^{(l)}$, respectively. From (25), $J(\boldsymbol{q}^{(\gamma)})\boldsymbol{g}^{(l)} = \boldsymbol{o}$ and $J(\boldsymbol{q}^{(\gamma)})\boldsymbol{h}^{(l)} = \boldsymbol{o}$ for $l = 1, 2, \cdots, N - m$. This implies that $J(\boldsymbol{q}^{(\gamma)})$ has $2(N - m)$ eigenvalues 0. □

Author's address

Yasuaki Kuroe: Department of Electronics and Information Science, Kyoto Institute of Technology, Matsugasaki, Sakyo-ku, Kyoto 606-8585 Japan.

Chapter 5

Clifford Networks

Justin Pearson

This chapter introduces a class of feed-forward neural networks which have Clifford valued weight and activation values. Clifford Algebras generalise the Complex and Quaternion algebras to higher-dimensions, thus Clifford networks are natural generalisations of complex valued networks. In this chapter the back-propagation algorithm is derived for Clifford valued networks and an approximation theorem is proved. Because Clifford Algebras are a generalisation of the Complex and Quaternion algebras the approximation results also shows that Complex networks are universal functional approximators.

1 Introduction

The complex numbers can be motivated either algebraically as providing a field where the equation $x^2 = -1$ can be solved or geometrically as providing an algebra of two-dimensional space. The geometric interpretation is used in engineering an signal processing to provide an algebraic framework that allows reasoning about frequency and phase information.

Clifford Algebras are born out of geometry. A Clifford Algebra includes the normal vector algebra but provides an algebraic framework where symmetries and geometric transformation s can be reasoned about within an algebraic framework.

In this chapter Clifford networks are presented which are a natural extension of Complex valued networks. An approximation theorem is proved for Clifford networks which shows that they are universal approximators in the sense of (Hornick, Stinchcombe & White 1989). This proof specialises and gives a proof of the approximating power of Complex valued feed-forward networks.

2 Clifford Algebras

In this chapter we will only consider Clifford Algebras defined over real valued vector spaces. Clifford Algebras arise from geometric motivations which are sketched below, algebraically they are algebras compatible with quadratic forms. Quadratic forms allow inner products and norms to be studied in non-euclidian spaces. The relationship is studied below.

In this section: first a direct construction of Clifford Algebras over real vector spaces is given; then this is generalised to Clifford algebras generated from Quadratic forms over vector spaces; this is then related back to the geometrical motivation given below; finally a classification is given of all Clifford Algebras over real vectors spaces showing that many matrix algebras over the Complex numbers, the Real Numbers and the Quaternions are in fact Clifford Algebras.

2.1 Geometric Motivation for Clifford Algebras

A Clifford Algebra is an extension of a vector algebra. It has two operations: addition which corresponds to normal vector addition and Clifford Multiplication which generalises the dot product and the cross product of vectors. As an example the construction of the Clifford Algebra Corresponding to 2-dimensional Euclidian Space is considered. A vector, $\vec{v}$, can be be represented as the sum of two unit length bases elements, $\vec{e_1}$ and $\vec{e_2}$ thus $\vec{v} = x\vec{e_1} + y\vec{e_2}$ then the scaler product of a vector $\vec{v}$ with itself, $\vec{v} \cdot \vec{v}$, is equal to the square of the length that is $x^2 + y^2$. If the vector is formally multiplied by its self then $(x\vec{e_1} + y\vec{e_2})(x\vec{e_1} + y\vec{e_2})$ becomes:

$$x^2\vec{e_1}^2 + y^2\vec{e_2}^2 + xy(\vec{e_1}\vec{e_2} + \vec{e_2}\vec{e_1})$$

In Euclidian space the length of a unit length basis element $\vec{e_i}$ is equal to 1, this gives the first equation true in a Clifford Algebra of a euclidian space, that is $\vec{e_i}^2 = 1$. The elements $\vec{e_1}\vec{e_2}$ and $\vec{e_2}\vec{e_1}$ can be thought of as directed areas and are referred as bivectors. For an arbitrary pair of vectors, $\vec{v_1}$ and $\vec{v_2}$ the bivectors $\vec{v_1}\vec{v_2}$ and $\vec{v_2}\vec{v_1}$ represent directed areas. Then the second main equation of in a Clifford Algebra (true in all Clifford Algebras) is:

$$\vec{e_1}\vec{e_2} = -\vec{e_2}\vec{e_1}$$

hence:

$$(x\vec{e_1} + y\vec{e_2})(x\vec{e_1} + y\vec{e_2}) = x^2 + y^2$$

giving the the normal scaler product of two vectors $\vec{v_1}$ and $\vec{v_2}$. The symmetric inner product is defined as:

$$\vec{v_1} \cdot \vec{v_2} = \frac{1}{2}\left(\vec{v_1}\vec{v_2} + \vec{v_2}\vec{v_1}\right)$$

by cancelling out the terms $\vec{e_1}\vec{e_2} + \vec{e_2}\vec{e_1} = 0$ the expression reduces to:

$$\frac{1}{2}(2x_1x_2 + y_1y_2) = x_1x_2 + y_1y_2$$

which is the normal scaler product of a vector.

The non-symmetrical version of the scaler product, the outer product is defined as:

$$\vec{v_1} \wedge \vec{v_2} = \frac{1}{2}(\vec{v_1}\vec{v_2} - \vec{v_2}\vec{v_1})$$

corresponds to the cross product of two vectors in three dimensions, but instead of giving a new vector gives a sum of directed areas, thus giving a dimension independent definition of the cross product which is intrinsic to the vector space, for example:

$$(x_1\vec{e_1} + y_1\vec{e_2} + z_1\vec{e_3}) \wedge (x_2\vec{e_1} + y_2\vec{e_2} + z_2\vec{e_3})$$

which is equal to:

$$(x_1y_2 - y_1x_2)\vec{e_1}\vec{e_2} + (z_1x_2 - x_1z_2)\vec{e_3}\vec{e_1} + (y_1z_2 - z_1y_2)\vec{e_2}\vec{e_3}$$

Using the above two rules, the product of any two vectors $\vec{v_1}$ and $\vec{v_2}$ can be written as:

$$\vec{v_1}\vec{v_2} = \vec{v_1} \cdot \vec{v_2} + \vec{v_1} \wedge \vec{v_2}$$

Clifford Algebras have applications in non-Euclidian spaces, that is vector spaces where $e_i^2 = -1$ for some elements e_i. The Lorentzian inner product used in special relativity where:

$$(x_1\vec{e_1} + y_1\vec{e_2} + z_1\vec{e_3} + t_1\vec{e_4}) \cdot (x_1\vec{e_1} + y_1\vec{e_2} + z_1\vec{e_3} + t_1\vec{e_4})$$

is defined to be:

$$x_1x_2 + y_1y_2 + z_1z_2 - t_1t_2$$

has an associated Clifford Algebra where $e_1^2 + e_2^2 + e_3^2 = 1$ and $e_4^2 = -1$.

Clifford Algebras have been used extensively in physics to aid calculations (Chisholm & Common 1986, Hestenes 1986)

2.2 A Direct Construction

A $p+q$ dimensional real vector space will be denoted as $\mathcal{R}^{p+q}$ (the reason for the notation $p+q$ will become clear later on). A Clifford Algebra $\mathcal{R}_{p,q}$ is two things, a 2^{p+q} vector space constructed from $\mathcal{R}^{p+q}$ and a set of algebraic rules defining multiplication and addition of vectors (when constructing Clifford Algebras using quadratic form theory, these rules come out naturally).

The vector space $\mathcal{R}^{p+q}$ has a basis of the form:

$$e_1, e_2, e_3, \ldots e_{p+q}$$

From this construct a $2^{n=p+q}$ dimensional vector space with basis elements:

$$\{e_A = e_{(h_1 \ldots h_r)} | A = (h_1, \ldots, h_r) \in \mathcal{P}(\mathcal{N}), 1 \leq h_1 < \ldots < h_r \leq n\}.$$

(where $\mathcal{P}(\mathcal{N})$ represents the set of subsets of the set $\{1, \ldots, n\}$). For example the vector space over $\mathcal{R}^{1,2}$ would have the basis,

$$e_{\emptyset},\ e_{(1)},\ e_{(2)}\ e_{(1,2)}, e_{(1,3)}, e_{(2,3)}, e_{(1,2,3)}$$

For notational convenience when no confusion can arise, $e_{(h_1, \ldots h_r)}$ will be denoted as $e_{h_1 h_2 \ldots h_r}$ and $e_{\emptyset} = e_0$ or since e_0 acts as the unit of the algebra it is often dropped when writing out elements a Clifford Algebra.

An element of the Clifford Algebra is written as a formal sum:

$$x = \sum_A x_A e_A$$

with each $x_A \in \mathcal{R}$. In what follows a summation with a capital letter near the beginning of the alphabet denotes a sum over the basis elements of a Clifford Algebra.

Addition of two elements of the algebra is defined as for vectors:

$$x + y = \sum_A (x_A + y_A) e_A$$

Multiplication is slightly more complicated. It is done formally element by element as in expanding brackets subject to the following algebraic rules:

$$e_i^2 = 1 \quad , \quad i = 1, \ldots, p \tag{1}$$
$$e_i^2 = -1 \quad , \quad i = p+1, \ldots, p+q \tag{2}$$
$$e_i e_j = -e_j e_i \quad , \quad i \neq j \tag{3}$$

with $1 \leq h_1 < \ldots h_r \leq n$, $e_{h_1} \cdot e_{h_2} \cdots e_{h_r} = e_{h_1 \ldots h_r}$. (the geometric motivation of these rules are as in section 2.1). This can be expressed more compactly in the following way,

$$e_A e_B = \kappa_{A,B} e_{A \Delta B},$$

where:

$$\kappa_{A,B} = (-1)^{\#((A \cap B) \setminus P)} (-1)^{p(A,B)} \tag{4}$$

P stands for the set $1, \ldots p$, and $\#X$ represents the number of elements in the set X,

$$p(A,B) = \sum_{j \in B} p'(A,j), \quad p'(A,j) = \#\{i \in A | i > j\}, \tag{5}$$

and the sets A, B and $A \Delta B$(the set difference of A and B) are ordered in the prescribed way.

For example in $\mathcal{R}_{1,1}$ given $x = 3 + 4e_1 + e_2$ and $y = e_2 + 2e_{12}$ then

$$\begin{aligned} xy &= (3 + 4e_1 + e_2)(e_2 + 2e_{12}) \\ &= 3e_2 + 6e_{12} + 4e_1 e_2 + 8e_1 e_{12} + e_2^2 + 2e_2 e_{12} \end{aligned}$$

By using the reduction rules $e_1 e_{12} = e_1 e_1 e_2 = e_1^2 e_2 = e_2$ and $e_2 e_{12} = -e_2 e_{21} = -e_2^2 e_1 = e_1$. So the product xy is equal to $xy = -1 + 2e_1 + 11e_2 + 10e_{12}$ In general a Clifford Algebra is associative but non-commutative.

2.3 Quadratic Forms

This section is intended to show that Clifford Algebras arise naturally as mathematical structures and in particular how the rules (1 - 3) arise.

An orthogonal space is a real linear vector space X together with a symmetric inner product $(\cdot|\cdot)$ from X^2 to $\mathcal{R}$ which is linear in each component. That is, the following equations are satisfied for all vectors x,y and z and all reals α and β, first is symmetric in x and y that is $(x, y) = (y, x)$ and further:

$$(\alpha x + \beta y, z) = \alpha(x, z) + \beta(y, z)$$
$$(x, \alpha y + \beta z) = \alpha(x, y) + \beta(x, z)$$

These equations abstract the standard inner product on euclidian vector spaces. A Quadratic form Q on X, can be constructed from an inner product by defining:

$$Q(x) = (x|x)$$

The inner product is recoverable from the Quadratic form by the following equation:

$$(x|y) = \frac{Q(x) + Q(y) - Q(x - y)}{2} \tag{6}$$

Thus the Quadratic form uniquely determines the inner product and vice versa.

A Clifford Algebra for a vector space X with respect to a Quadratic form Q is the universal real associative algebra A which has the vector space X embedded in, such that for each element x of X, the following is true (see (Porteous 1995, Blaine Lawson Jr. & Michelsohn 1989) for details of the universal construction):

$$x^2 = -Q(x) \tag{7}$$

(where x^2 is carried out in the algebra A).

There is a theorem in Quadratic form theory due to Sylvester (for a proof see (Lam 1973) or (Bromwich 1986, O'Meara, T.O 1999)) that given a real vector space and a Quadratic form, it is possible by change of basis to represent the Quadratic form as:

$$Q(x) = -x_1^2 - x_2^2 - \cdots x_p^2 + x_{p+1}^2 + \cdots + x_{p+q}^2$$

where p, q is called the signature of the Quadratic form, and $p+q$ is equal to the dimension of X.

Thus to get the equations (1 - 3), we apply the equation (7) to the basis elements of X:

$$e_i^2 = -Q(e_i) = +1 \quad \text{For } 0 < i \leq q \tag{8}$$
$$e_i^2 = -Q(e_i) = -1 \quad \text{For } q < i < p+q \tag{9}$$
$$\tag{10}$$

The anti-commutative law (equation (3)) can be derived from the following proposition.

Proposition 1 *For all $x, y \in X$ then in A,*

$$(x|y) = \frac{-(xy+yx)}{2} \tag{11}$$

Proof: From the formula (6):

$$2(x|y) = Q(x)+Q(y)+Q(x-y) = -x^2-y^2+(x-y)^2 = -xy-yx$$

In particular for distinct basis elements e_i and e_j we have $(e_i|e_j) = 0$ because e_i and e_j are orthogonal and

$$-2(e_i|e_j) = e_ie_j + e_je_i$$

from the equation (11) which implies that $e_ie_j + e_je_i = 0$.

2.4 Some Familiar Clifford Algebras

We now show that the Clifford Algebras arising from vector spaces over the reals include the Complex numbers the Quaternions and various matrix algebras; a complete classification can be found in (Porteous 1981, Blaine Lawson Jr. & Michelsohn 1989, Porteous 1995). It is easy to show that the complex numbers are simply the Clifford Algebra $\mathcal{R}_{0,1}$.

The quaternion algebra is generated from the basis elements $1, i, j, k$ with the relations $i^2 = j^2 = k^2 = -1$ and

$$ij = k = -ji \quad , \quad jk - i = -kj \quad , ki = j = -ik$$

The quaternions are isomorphic to $\mathcal{R}_{0,2}$ with the following isomorphisms,

$$e_0 \cong 1 \, , \, e_1 \cong i \, , \, e_2 \cong j \, , \, e_{12} \cong k$$

Some perhaps less familiar examples include the Dirac algebra $\mathcal{R}_{4,1}$ and the Pauli algebra $\mathcal{R}_{3,3}$.

The algebra ${}^2\mathcal{R}$ (often denoted as $\mathcal{R} \oplus \mathcal{R}$) is defined over ordered pairs (x_1, x_2) with addition and multiplication defined as:

$$(x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y_2)$$
$$(x_1, x_2) * (y_1, y_2) = (x_1 * y_1, x_2 * y_2)$$

This algebra is isomorphic to the algebra $\mathcal{R}_{1,0}$ under the isomorphism ϕ given by:

$$\phi(\alpha + \beta e_1) \mapsto (\alpha + \beta, \alpha - \beta)$$

with

$$\phi^{-1}(a, b) \mapsto \frac{a+b}{2} + \frac{a-b}{2} e_1$$

The algebra $\mathcal{R}_{1,0}$ is called the algebra of hyperbolic complex numbers and can be used in relativity calculations in physics.

$R_{1,1}$ is isomorphic to the set of 2 by 2 real valued matrices. This can seen by the following identification,

$$e_1 \cong \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad e_2 \cong \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$
$$e_{12} \cong \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad e_0 \cong \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

This is a basis for $\mathcal{R}(2)$ and defines the isomorphism of algebras. This can be generalised to show that $\mathcal{R}_{n,n} \cong \mathcal{R}(2^n) \cong \text{End}(\mathcal{R}^n)$.

2.5 A Partial Classification of Clifford Algebras

Every Clifford Algebra is either isomorphic to a matrix algebra of $\mathcal{R}, \mathcal{C}, \mathcal{H}$ or a direct product of such matrix algebras. Not all Clifford Algebras are distinct, and there is the so called periodicity theorem which relates higher dimensional Clifford Algebras to low dimensional algebras. This section is a short guide to the relationships between Clifford Algebras.

All proofs in this section are omitted and can be found in Chapter 13 of (Porteous 1981). So far the following relationships have been demonstrated:

$$\mathcal{R}_{0,0} \cong \mathcal{R}\ ,\ \mathcal{R}_{0,1} \cong \mathrm{C}\ ,\ \mathcal{R}_{0,2} \cong \mathcal{H}\ ,\ \mathcal{R}_{n,n} \cong \mathcal{R}(2^n)\ ,\ \mathcal{R}_{1,1} \cong {}^2\mathcal{R}$$

In fact all Clifford Algebras defined over the real numbers can be constructed in some way from $\mathcal{R}$,$\mathcal{C}$ or $\mathcal{H}$. Table 1 extends this information. The reader interested in the explicit construction of the table should again consult (Porteous 1981) or (Blaine Lawson Jr. & Michelsohn 1989). The most useful fact (again for a proof see (Porteous 1981)) is perhaps that $\mathcal{R}_{p+1,q} \cong \mathcal{R}_{q+1,p}$.

To complete the table to arbitrary algebras it is enough to know that

$$\mathcal{R}_{p,q+8} \cong \mathcal{R}_{p,q} \otimes_{\mathcal{R}} \mathcal{R}(16) \cong \mathcal{R}_{p,q}(16).$$

where $\otimes_{\mathcal{R}}$ denotes the real tensor product of two algebras. This is the so called periodicity theorem a proof can be found in (Porteous 1981) or (Blaine Lawson Jr. & Michelsohn 1989)

3 Clifford Back-Propagation

Now the material on Clifford Algebras has been setup, we derive the back-propagation equations for a feed-forward Clifford network with Clifford valued weights and synapses. The derivation of a Clifford valued feed-forward network is similar to the derivation of

Table 1. A table of Clifford Algebras up to dimension 256, p goes horizontally and q vertically

$\mathcal{R}_{p,q}$	0	1	2	3	4	5	6	7	8
0	$\mathcal{R}$	$\mathcal{C}$	$\mathcal{H}$	${}^2\mathcal{H}$	$\mathcal{H}(2)$	$\mathcal{C}(4)$	$\mathcal{R}(8)$	${}^2\mathcal{R}(8)$	$\mathcal{R}(16)$
1	${}^2\mathcal{R}$	$\mathcal{R}(2)$	$\mathcal{C}(2)$	$\mathcal{H}(2)$	${}^2\mathcal{H}(2)$	$\mathcal{H}(4)$	$\mathcal{C}(8)$	$\mathcal{R}(16)$	${}^2\mathcal{R}(16)$
2	$\mathcal{R}(2)$	${}^2\mathcal{R}(2)$	$\mathcal{R}(4)$	$\mathcal{C}(4)$	$\mathcal{H}(4)$	${}^2\mathcal{H}(4)$	$\mathcal{H}(8)$	$\mathcal{C}(16)$	$\mathcal{R}(32)$
3	$\mathcal{C}(2)$	$\mathcal{R}(4)$	${}^2\mathcal{R}(4)$	$\mathcal{R}(8)$	$\mathcal{C}(8)$	$\mathcal{H}(8)$	${}^2\mathcal{H}(8)$	$\mathcal{H}(16)$	$\mathcal{C}(32)$
4	$\mathcal{H}(2)$	$\mathcal{C}(4)$	$\mathcal{R}(8)$	${}^2\mathcal{R}(8)$	$\mathcal{R}(16)$	$\mathcal{C}(16)$	$\mathcal{H}(16)$	${}^2\mathcal{H}(16)$	$\mathcal{H}(32)$
5	${}^2\mathcal{H}(2)$	$\mathcal{H}(4)$	$\mathcal{C}(8)$	$\mathcal{R}(16)$	${}^2\mathcal{R}(16)$	$\mathcal{R}(32)$	$\mathcal{C}(32)$	$\mathcal{H}(32)$	${}^2\mathcal{H}(32)$
6	$\mathcal{H}(4)$	${}^2\mathcal{H}(4)$	$\mathcal{H}(8)$	$\mathcal{C}(16)$	$\mathcal{R}(32)$	${}^2\mathcal{R}(32)$	$\mathcal{R}(64)$	$\mathcal{C}(64)$	$\mathcal{H}(64)$
7	$\mathcal{C}(8)$	$\mathcal{H}(8)$	${}^2\mathcal{H}(8)$	$\mathcal{H}(16)$	$\mathcal{C}(32)$	$\mathcal{R}(64)$	${}^2\mathcal{R}(64)$	$\mathcal{R}(128)$	$\mathcal{C}(128)$
8	$\mathcal{R}(16)$	$\mathcal{C}(16)$	$\mathcal{H}(16)$	${}^2\mathcal{H}(16)$	$\mathcal{H}(32)$	$\mathcal{C}(64)$	$\mathcal{R}(128)$	${}^2\mathcal{R}(128)$	$\mathcal{R}(256)$

the learning rule for a complex valued network as in (Georgiou & Koutsougeras 1992).

In what follows the norm on an arbitrary Clifford number, $\|\cdot\|$ will be used, where,

$$\|x\| = \left(\sum_A [x]_A^2\right)^{\frac{1}{2}} \tag{12}$$

where $[x]_A$ represents the A'th part of the Clifford number x.

A feed-forward Clifford network with n inputs and m outputs will have a transfer function,

$$\Psi : (\mathcal{R}_{p,q})^n \rightarrow (\mathcal{R}_{p,q})^m$$

Where $(\mathcal{R}_{p,q})^n$ is the n-dimensional left module over the Clifford Algebra $\mathcal{R}_{p,q}$.

To implement Clifford back-propagation an error measure E is defined which measures how well the network models a data set X. The basic form is the same,

$$E = \frac{1}{2} \sum_{\mathbf{x} \in X} \|\Psi - \Phi\|^2$$

where X is a set of training vectors.

It is convenient from the point of view of the derivation to define $\|\cdot\|$ as:

$$\|\mathbf{x}\|^2 = \sum_{i=1}^{k} |(x)_i|^2$$

where $(x)_i$ is a Clifford number representing the i'th part of $\mathbf{x}$ in the m-dimensional Clifford module over $\mathcal{R}_{p,q}$.

Assume that each node in the network has the same Clifford valued activation function $f : \mathcal{R}_{p,q} \to \mathcal{R}_{p,q}$. The output o_j of the j'th neuron can be written as,

$$o_j = f(net_j) = \sum_A u_A^j e_A$$

With u_A^j a function from $\mathcal{R}_{p,q}$ to $\mathcal{R}$ and

$$net_j = \sum_l \omega_{lj} o_l$$

where l sums over all the inputs to neuron j.

It is important to notice since $\mathcal{R}_{p,q}$ is in general non-commutative the order of multiplication in the above equation is important, although it will be shown later (in Section 4) that networks with left weight multiplication are equivalent in expressive power to networks with right multiplication.

In the real case E depends on the number of weights in the network. In the Clifford case E depends not only on all the weights but on the components of each of the weights. Again define $\lambda = \|\Psi - \Phi\|^2$. Then:

$$\frac{\partial E}{\partial[\omega_{ij}]_A} = \frac{1}{2} \sum_{\mathbf{x}\in X} \frac{\partial \lambda}{\partial[\omega_{ij}]_A}$$

and using the chain rule,

$$\frac{\partial \lambda}{\partial[\omega_{ij}]_A} = \sum_B \left(\frac{\partial \lambda}{\partial u_B^j} \left(\sum_C \frac{\partial u_B^j}{\partial[net_j]_C} \frac{\partial[net_j]_C}{\partial[\omega_{ij}]_A} \right)\right)$$

The partial derivative

$$\frac{\partial[net_j]_C}{\partial[\omega_{ij}]_A}$$

needs a bit of care. Using equation 3:

$$\frac{\partial[net_j]_C}{\partial[\omega_{ij}]_A} = \sum_l \frac{\partial[\omega_{lj}x_l]_C}{\partial[\omega_{ij}]_A} = \frac{\partial[\omega_{ij}o_i]_C}{\partial[\omega_{ij}]_A}$$

Then using the fact that $\omega_{ij}o_i = \sum_{D,E}[\omega_{ij}]_D[o_i]_E e_D e_E$ and

$$\frac{\partial[\omega_{ij}o_i]_C}{\partial[\omega_{ij}]_A} = \frac{\partial\left(\sum_{D,E}[\omega_{ij}]_D[o_i]_E\kappa_{A,E}\right)}{\partial[\omega_{ij}]_A}$$

with κ defined as in (4) and D, E summing over all the elements such that $e_D e_E = \pm e_C$. Since the denominator of the partial derivative only refers to $[\omega_{ij}]_A$ the partial derivative will equal:

$$\frac{\partial[\omega_{ij}o_i]_C}{\partial[\omega_{ij}]_A} = \frac{\partial[\omega_{ij}]_A[o_i]_E\kappa_{A,E}}{\partial[\omega_{ij}]_A} = \kappa_{A,E}[o_i]_E \text{ with } e_A e_E = \pm e_C \tag{13}$$

For example in the algebra $\mathcal{R}_{2,0}$ the table of derivatives would look like,

$\frac{\partial[x]_B}{\partial[\omega_{jl}]_A}$	$B = 0$	1	2	12
$A = 0$	$[x_{jl}]_0$	$[x_{jl}]_1$	$[x_{jl}]_2$	$[x_{jl}]_{12}$
1	$[x_{jl}]_1$	$[x_{jl}]_0$	$[x_{jl}]_{12}$	$[x_{jl}]_2$
2	$[x_{jl}]_2$	$-[x_{jl}]_{12}$	$[x_{jl}]_0$	$-[x_{jl}]_1$
12	$-[x_{jl}]_{12}$	$[x_{jl}]_2$	$-[x_{jl}]_1$	$[x_{jl}]_0$

The error derivative is now quite easy to calculate. If j is an output neuron then,

$$\frac{\partial\lambda}{\partial u_A^j} = \frac{\partial}{\partial u_A^j}\|\Psi - \Phi\|$$

$$\frac{\partial}{\partial u_A^j}|o_j - \Phi_j^2|^2 = 2[o_j - \Phi_j]_A$$

If j is not an output unit then the chain rule has to be used again.

$$\frac{\partial \lambda}{\partial u_A^j} = \sum_k \frac{\partial \lambda}{\partial u_A^k} \left(\sum_{B,C} \frac{\partial u_B^k}{\partial [net_k]^C} \frac{\partial [net_k]^C}{\partial u_A^j} \right)$$

with k running over the neurons that receive input from neuron j.

The term

$$\frac{\partial [net_k]_C}{\partial [u_j]_A}$$

is calculated in a similar manner to (13),

$$\frac{\partial [net_k]_C}{\partial u_A^j} = \kappa_{A,E} [\omega_{jk}]_D$$

where $\kappa_{A,E} e_A e_E = e_C$. The derivatives:

$$\frac{\partial u_B^k}{\partial [x_k]_C}$$

play the same rôle as $f'(net_j)$ does in the real-valued case and depends on the activation function used; this will be discussed in the next section.

Bringing this all together we have,

$$\frac{\partial E}{\partial [\omega_{ij}]_A} = \frac{1}{2} \sum_{x \in X} \sum_B \lambda_j^B \left(\sum_C \frac{\partial u_B^j}{\partial [net_j]_C} \kappa_{A,E} [o_k]_E \right)$$

with $e_A e_E = \pm e_C$ and

$$\lambda_j^B = \frac{\partial ||\Psi - \Phi||^2}{\partial u_B^j} = 2[o_j - \Phi_j]_B$$

if j is an output neuron. If j is not an output unit then the chain rule has to be used again:

$$\lambda_j^B = \sum_k \lambda_k^B \left(\sum_{B,C} \frac{\partial u_B^k}{\partial [net_k]^C} \kappa_{A,D} [\omega_{jk}]_D \right)$$

with k running over the neurons that receive input from neuron j and $e_A e_D = e_C$.

The choice of activation function for a Clifford network as with a complex network requires some care. The normal sigmoid function $(1 + e^{-x})^{-1}$ can not be used. In (Georgiou & Koutsougeras 1992) a simple complex activation is proposed:

$$f(z) = \frac{z}{c + \frac{1}{r}|z|}$$

This function extended to the Clifford Domain is a suitable activation function and has been used successfully in experiments (Pearson & Bisset 1992, Pearson & Bisset 1994, Rahman, Howells & Fairhurst 2001, Pearson 1995) and applications.

4 Approximation Results

Now we have derived the back-propagation rules for Clifford networks this proves that Clifford networks are universal approximators: that is they any compact continuous function can be approximated arbitrarily close with a feed-forward Clifford network, these results generalise the results in (Hornick et al. 1989, Cybenko 1989) from real valued networks to Clifford valued networks.

There are essentially two ways of analysing feed-forward networks. The first views a feed-forward network as a pattern classifier and uses statistical techniques to assess the performance of a network; see (MacKay 1992). The second treats a feed-forward network essentially as a function approximator, that is, given a network with n inputs and m outputs and a set of weight values ω_{ij} the network can be seen to be computing a function:

$$\Phi_\omega : \mathcal{R}^n \rightarrow \mathcal{R}^m$$

In the Clifford case the real numbers $\mathcal{R}$ are replaced by an arbitrary Clifford Algebra $\mathcal{R}_{p,q}$. The sort of question then asked is how well can a given class of networks approximate classes of functions? Various theorems have been proved (Cybenko 1989, Hornick et al. 1989, Ito 1991, Kůrková 1991) which show that feed-forward networks with one hidden layer are sufficient to approximate continuous functions. Further results by Sontag (Sontag 1992) show that in certain problems two hidden layers are required; this is because the function trying to be approximated is too discontinuous to be approximated by a single hidden layer network.

This section first extends Cybenko's (Cybenko 1989) proof, that real valued networks with a single hidden layer can approximate any bounded continuous function with compact support, to networks over an arbitrary Euclidean Clifford Algebra (that is algebras with signature $0, q$), these include the Complex numbers $\mathcal{R}_{0,1}$ and the Quaternions $\mathcal{R}_{0,2}$. Before the proof can be made some background material has to be provided in Clifford modules (section 4.1) and Clifford Analysis.

4.1 Clifford Modules and Clifford Analysis

This section deals with a generalisation of vector spaces, the theory of Modules over rings: specifically Clifford modules. This generalisation is necessary in order to state the relevant approximation theorems. Various theorems are stated which are generalisations of traditional theorems such as the Hahn-Banach theorem and the Riesz representation theorem (Rudin 1966, Rudin 1973); all proofs are omitted, but these can be found in (Brackx, Delenghe & Sommen 1982).

From now on the convention adopted in (Brackx et al. 1982) is used, where a Euclidean Clifford Algebra is referred as an $\mathcal{A}$ algebra. A module is a generalisation of a vector space, where the set of coefficients come from some ring instead of a field, thus modules have a different geometrical structure from vector spaces.

Definition 1 *A unitary left* $\mathcal{A}$-module $X_{(l)}$ *is an Abelian group* $X_{(l)}, +$ *and an operation* $(\lambda, f) \rightarrow \lambda f$ *from* $\mathcal{A} \times X_{(l)}$ *into* $X_{(l)}$ *s.t.*

for all $\lambda, \mu \in \mathcal{A}$ *and* $f, g \in X_{(l)}$ *the following hold:*

$$(\lambda + \mu)f = \lambda f + \mu f$$
$$(\lambda\mu)f = \lambda(\mu f)$$
$$\lambda(f + g) = \lambda f + \lambda g$$
$$e_0 f = f$$

We have already met an example of a Clifford Module in Section 3, the space $\mathcal{R}^n_{p,q}$.

Definition 2 *Let* $X_{(l)}$ *be a unitary left* $\mathcal{A}$*-module, then a function* $p : X_{(l)} \to \mathcal{R}$ *is said to be a* proper semi-norm *if there exists a constant* $C_0 \geq 0$ *s.t. for all* $\lambda \in \mathcal{A}$ *and* $f, g \in X_{(l)}$ *the following conditions are satisfied:*

$$p(f + g) \leq p(f) + p(g)$$
$$p(\lambda f) \leq C_0 |\lambda| p(f)$$
$$p(\lambda f) = |\lambda| p(f) \;\; \textit{if}\, \lambda \in \mathcal{R}$$
$$\textit{If}\, p(f) = 0 \textit{ then } \; f = 0$$

Definition 3 *Given a module* $X_{(l)}$ *the* algebraic dual $X^{*alg}_{(l)}$ *is defined to be the set of left* $\mathcal{A}$*-linear functionals from* $X_{(l)}$ *into* $\mathcal{A}$*. That is the set of functionals* $T : X_{(l)} \to \mathcal{A}$ *s.t.*

$$T(\lambda f + g) = \lambda T(f) + T(g)$$

$f, g \in X_{(l)}$ *and* $\lambda \in \mathcal{A}$.

Definition 4 *The set of bounded T functionals with respect to a semi-norm p is denoted* $X^*_{(l)} \subset X^{*alg}_{(l)}$*. Explicitly for all functionals T and for all* $f \in X_{(l)}$*:*

$$|T(f)| \leq C p(f)$$

for some real constant C.

The following theorem is a a corollary to a Hahn-Banach type theorem for Clifford modules for details and proof see sections 2.10-2.11 in (Brackx et al. 1982).

Theorem 1 *Let $X_{(l)}$ be a unitary left $\mathcal{A}$-module provided with a semi norm p and let $Y_{(l)}$ be a submodule of $X_{(l)}$. Then $Y_{(l)}$ is dense in $X_{(l)}$ if and only if for each $T \in X^*_{(l)}$ such that $T|Y_{(l)} = 0$*[1] *we have $T = 0$ on $X_{(l)}$.*

Now a useful class of function spaces is introduced.

Definition 5 *The space $C^0(\mathcal{K}; \mathcal{A})$. Let $\mathcal{K}$ be a compact subset of $\mathcal{R}^r$ ($r \geq 1$). Then $C^0(\mathcal{K}; \mathcal{A})$ stands for the unitary bi-$\mathcal{A}$-module of $\mathcal{A}$-valued continuous functions on $\mathcal{K}$.*

This can be thought of as a product of classical real valued functions i.e.:

$$C^0(\mathcal{K}; \mathcal{A}) = \Pi_A C^0(\mathcal{A}; \mathcal{R}) e_A \tag{14}$$

where A runs over all the basis elements in the Clifford Algebra in question. A norm can be defined for each $f \in C^0(\mathcal{K}; \mathcal{A})$:

$$||f|| = \sup_{x \in \mathcal{K}} |f(x)|$$

This norm is equivalent to the product norm taken from (14).

Definition 6 *Given an open set $\Omega \subset \mathcal{R}^n$ and a sequence $(\mu_B)_B$ of real valued measures on Ω. Then for any open set in Ω an $\mathcal{A}$ valued measure can be defined:*

$$\mu(I) = \sum_B \mu_B(I) e_B$$

Definition 7 *An A-valued function:*

$$f = \sum_A f_a e_A$$

is said to be μ-integrable in Ω if for all A and B ranging over the basis elements of $\mathcal{A}$ each f_A is μ_B integrable.

[1] T restricted to $Y_{(l)}$ equal to zero

Definition 8 *For any μ-integrable function f define:*

$$\int_\Omega f(x)d\mu = \sum_{A,B} e_A e_B \int_\Omega f_A(x)d\mu_B$$

A Riesz representation type theorem can be obtained.
Theorem 2 *Let T be a bounded $\mathcal{A}$ valued function in $C^0_{(l)}(\mathcal{K};\mathcal{A})$. Then there exists a unique $\mathcal{A}$ valued measure μ with support contained in $\mathcal{K}$ such that for all $f \in C^0_{(l)}(\mathcal{K};\mathcal{A})$:*

$$T(f) = \int_{\mathcal{K}} f(x)d\mu$$

For a proof again see (Brackx et al. 1982).

4.2 The Approximation Result

Now all the machinery has been set up and the approximation result can be proved. A feed-forward network with one output neuron and N inputs units and K hidden units computes a function:

$$\Phi(\mathbf{x}) = \sum_{j=1}^{K} \alpha_j f(\sum_{i=1}^{N} y_{ij}x_i + \theta_j)$$

with f the activation function x_i the i'th input, y_{ij} weight values for the connection between the input layer and the hidden layer and α_j the weights from the hidden layer to the output node.

The function $\Phi(\mathbf{x})$ can be seen as a function from $\mathcal{R}^{N2^n}$ (where 2^n is the dimension of $\mathcal{A}$) to $\mathcal{A}$ and hence a member of $C^0_{(l)}(\mathcal{R}^{N2^n};\mathcal{A})$.
Definition 9 *An activation function f (considered as a function from $\mathcal{R}^{N2^n}$ to $\mathcal{A}$) is said to be* discriminating *if for any given Clifford valued measure μ with support I^{N2^n} if:*

$$\int_{I^{N2^n}} f(\sum_{i=1}^{N} y_i x_i + \theta)d\mu(x) = 0$$

for all $y_i, \theta \in \mathcal{R}_{0,n}$ implies $\mu(x) = 0$.

It will be shown that when networks have activation functions that are discriminating are universal approximators.

The following theorem is the heart of the approximation result.
Theorem 3 *Let f be any continuous discriminating functions. Then finite sums of the form:*

$$\Phi(x) = \sum_{j=1}^{K} \alpha_j f(\sum_{i=1}^{N} y_{ij} x_i + \theta_j) \tag{15}$$

are dense in $C^0_{(l)}(I^{N2^n}; \mathcal{A})$

Proof: The proof is essentially a modification of Cybenko's Theorem 1 in (Cybenko 1989) using the theory of Clifford modules in the last section.

Let S be the function space generated by sums of the form (15). Assume that the closure of S is not all of $C^0_{(l)}(I^{N2^n}; \mathcal{A})$; denote the closure of S by R. By the Hahn-Banach type theorem 1 there is a bounded linear functional T on $C^0_{(l)}(I^{N2^n}; \mathcal{A})$, with $T \neq 0$ but $T(R) = T(S) = 0$. By Theorem 2 this bounded linear functional is of the form:

$$T(h) = \int_{I^N 2^n} h(x)\mu(x)$$

for some measure μ and $h \in C^0_{(l)}(I^{k2^n}, \mathcal{A})$. In particular since $f \in C^0_{(l)}(I^{k2^n}, \mathcal{A})$ is in R, for any y_i:

$$T(f) = \int_{k2^n} f(\sum_{i=1}^{N} y_i x_i + \theta) d\mu(x) = 0$$

Since f is discriminating this implies $\mu = 0$ contradicting our assumption hence S must be dense in $C^0_{(l)}(I^{N2^n}; \mathcal{A})$.

So to prove that the class of feed-forward networks considered in Chapter 3 are universal approximators, we have to show that functions of the form:

$$f(x) = \frac{x}{1+|x|}$$

are discriminating.

Theorem 4

$$f(x) = \frac{x}{1+|x|}$$

is discriminatory.

Proof: A function $f(x)$ is discriminatory if:

$$\int_{N2^n} f(\sum_{i=1}^{N} y_i x_i + \theta) d\mu(x) = 0$$

for all y_i implies that $\mu(x) = 0$. This is equivalent to saying that:

$$\int_{N2^n} f(\sum_{i=1}^{N} y_i x_i + \theta) d\mu(x) = \sum_{A,B} e_A e_B \int_{N2^n} f_A(\sum_{i=1}^{N} y_i x_i + \theta) d\mu_B(x) = 0$$

for all y_i.

Define $\gamma_A(x) : I^{N2^n} \rightarrow \mathcal{R}$ to be the limit of:

$$\gamma_A(x) = \lim_{\lambda \rightarrow \infty} f_A(\lambda x)$$

(where λx is a Clifford multiplication, with λ a real number). So

$$f_A(\lambda x) = \frac{[\lambda z]_A}{1+|\lambda z|} = \frac{\lambda [z]_A}{1+\lambda |z|}$$

So

$$\gamma_A(z) = \begin{cases} 1 & \text{if } [z]_A > 0 \\ 0 & \text{if } [z]_A = 0 \\ -1 & \text{if } [z]_A < 0 \end{cases}$$

In our case:

$$\gamma_A(\sum_{i=1}^{N} y_i x_i + \theta) = \begin{cases} 1 & \text{if } [\sum_{i=1}^N y_i x_i + \theta]_A > 0 \\ 0 & \text{if } [\sum_{i=1}^N y_i x_i + \theta]_A = 0 \\ -1 & \text{if } [\sum_{i=1}^N y_i x_i + \theta]_A < 0 \end{cases}$$

The sets defined by $[\sum_{i=1}^N y_i x_i + \theta]_A = 0$ are hyper-planes, since $[\sum_{i=1}^N y_i x_i + \theta]_A$ is just a set of linear equations in the components of x_i.

The rest of the proof is almost verbatim from Lemma 1 of Cybenko (Cybenko 1989). So let $\Pi^A_{y,\theta} \subset I^{2^n}$ be the hyper-plane defined by:

$$\left\{ x \middle| \left[\sum_{i=1}^{N} y_i x_i + \theta \right]_A = 0 \right\}$$

and let $H^A_{y,\theta}$ be the half space defined by:

$$\{x | [\sum_{i=1}^{N} y_i x_i + \theta]_A > 0\}$$

Then by the Lebesgue bounded convergence theorem we have:

$$0 = \int_{I^{N2^n}} f_A(\lambda x) d\mu_B(x) = \int_{I^{N2^n}} \gamma_A(x) d\mu_B(x) = \mu(H^A_{y,\theta})$$

Now if μ_B were always a positive measure the result would be trivial, but since μ_B is an arbitrary measure the result is harder (since positive bits of μ might cancel out negative bits of μ_B).

Fix the y_i's and define:

$$F_A(h) = \int_{I^{N2^n}} h([\sum_{i=1}^{K} y_i x_i]_A)$$

for some bounded μ_B measurable function $h : \mathcal{R} \to \mathcal{R}$. F_A is a bounded functional on $L^\infty(\mathcal{R})$.

Let h be the indicator function on the interval $[\theta_A, \infty)$, then:

$$F(h) = \int_{k2^n} h([\sum_{i=1}^{K} y_i x_i]_A) = \mu_B(\Pi^A_{y,\theta}) + \mu(H^A_{y,\theta})$$

Similarly $F(h) = 0$. If h is the indicator of any open interval, by linearity $F(h) = 0$ and hence for any simple function. Since the simple functions are dense in $L^\infty(\mathcal{R})$, $F = 0$.

In particular given the two functions $s(x) = \sin(x)$, $c(x) = \cos(x)$:

$$\begin{aligned} F_A(s(x) + ic(x)) &= \int_{I^{k2^n}} s([\sum_{k=1}^{K} y_k x_k]_A) + ic([\sum_{k=1}^{K} y_k x_k]_A) d\mu_B \\ &= \int_{I^{k2^n}} \exp(i[\sum_{k=1}^{K} y_k x_k]_A) d\mu_B \\ &= 0 \end{aligned} \tag{16}$$

for all y_k. Therefore the Fourier transform of μ_B is zero, hence μ_B must be zero and hence f is discriminatory.

One important thing to point out with this proof is that the order of weight multiplication is irrelevant; the whole proof could be repeated with networks where multiplication was done on the right. Thus it does not matter theoretically which sort of nets (left or right weight multiplication) is used for a particular problem. Practically not much is known, but in all the examples the author has tried, the performance of the net does not seem to be affected by the order of weight multiplication.

5 Conclusion and Related Work

This chapter has been largely theoretical, but it has been shown that it is possible to derive a back-propagation algorithm for Clifford valued feed-forward networks. Such networks are a natural extension of Complex valued networks by virtue of Clifford Algebras being the natural geometric extension of the Complex numbers. Due to space limitations no experimental results have been presented, applications of Clifford networks can be found in (Pearson 1995, Rahman et al. 2001). Other work on Clifford valued neural networks has been done with self-organising networks with Clifford Algebras applied to motion modelling systems (Bayro-Corrochano, Buchholz & Sommer 1996*b*, Bayro-Corrochano, Buchholz & Sommer 1996*a*, Bayro-Corrochano 1996).

References

Bayro-Corrochano, E. (1996), Clifford selforganizing neural network, clifford wavelet network., *in* 'Proc. 14th IASTED Int. Conf. Applied Informatics. Innsbruck, Austria,', pp. 271–274.

Bayro-Corrochano, E., Buchholz, S. & Sommer, G. (1996*a*), A new selforganizing neural network using geometric algebra, *in* 'Proc. Int. Neural Network Society 1996 Annual Meeting: World Congress on Neural Networks, WCNN'96, CA, San Diego, USA', pp. 245–249.

Bayro-Corrochano, E., Buchholz, S. & Sommer, G. (1996*b*), Selforganizing clifford networks, *in* 'Proceedings of ICNN', Vol. 1, pp. 120–125.

Blaine Lawson Jr., H. & Michelsohn, M.-L. (1989), *Spin Geometry*, Princeton University Press, Princeton, New Jersey.

Brackx, F., Delenghe, R. & Sommen, F. (1982), *Clifford Analysis*, Research notes in mathematics; 76, Pitman Advanced Publishing Program.

Bromwich, T. I. (1986), *Quadratic Forms and Their Classification by Means of Invariant-Factors*, Vol. 3 of *Cambridge Tracts in Mathematics and Mathematical Physics*, Cambridge University Press.

Chisholm, J. & Common, A., eds (1986), *Clifford Algebras and their applications in mathematical physics*, Vol. 183 of *NATO ASI series, C:Mathematical and physical sciences*.

Cybenko, G. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control Signals and Systems* pp. 303–314.

Georgiou, G. M. & Koutsougeras, C. (1992), 'Complex domain backpropagation', *IEEE Transactions on Circuits and Systems* pp. 330–334.

Hestenes, D. (1986), *New foundations for Classical Mechanics*, Reidel.

Hornick, K., Stinchcombe & White (1989), 'Multilayer feedforward networks are universal approximators', *Neural Networks* **2**, 359–366.

O'Meara, T.O., '*Introduction to Quadratic Forms*' (1999), Springer Verlag, Classics in Mathematics.

Ito, Y. (1991), 'Representation of functions by superpositions of step or sigmoid functions and their applications to neural network theory', *Neural Networks* **4**, 385–394.

Kůrková, V. (1991), 'Kolmogorov's theorem is relevant', *Neural Computation* **3**(4), 617–622.

Lam, T. (1973), *The algebraic theory of quadratic forms*, Mathematics lecture note series, W.A. Benjamin, Reading, Mass.

MacKay, D. J. (1992), Bayesian Methods for Adaptive Models, PhD thesis, California Institute of Technolgy, Pasadena, California.

Pearson, J. & Bisset, D. (1992), Back Propagation in a Clifford Algebra, *in* 'ICANN Brighton'.

Pearson, J. & Bisset, D. (1994), Neural Networks in the Clifford Domian, *in* 'IEEE94 symposium on Neural Networks Orlando'.

Pearson, J. K. (1995), Clifford Networks, PhD thesis, University of Kent (Electronic engineering).

Porteous, I. (1981), *Topological Geometry*, Cambridge University Press.

Porteous, I. (1995), *Clifford Algebras and the Classical Groups*, Cambridge Studies in Advanced Mathematics, Cambridge University Press.

Rahman, A., Howells, W. & Fairhurst, M. (2001), 'A multi-expert framework for character recognition: A novel application of clifford networks', *IEEE Neural Networks* **12**(1).

Rudin, W. (1966), *Real and complex analysis*, McGraw-Hill series in higher mathematics, McGraw-Hill.

Rudin, W. (1973), *Functional analysis*, McGraw-Hill series in higher mathematics, McGraw-Hill.

Sontag, E. (1992), 'Feedback stablisation using two hidden layer nets', *IEEE Transactions on Neural Networks* **3** pp. 981–990.

Author's address

Justin Pearson: Uppsala University Department of Information Technology Box 337 SE-751 05 Uppsala Sweden.

Chapter 6

Complex Associative Memory and Complex Single Neuron Model

Iku Nemoto

In this chapter, we present two different applications of complex neuron models. One is a complex associative memory and the other is a complex version of the Nagumo-Sato model of a single neuron. Although these two applications are on the opposite extremes of scale, the one with many (desirably infinite in the limit) units and the other with a single or at most a few neurons, the motivation for the use of complex numbers is the same; we want to treat the timing or the phase of impulse trains with as simple a method as possible. Therefore, the properties of the model so far found should be attributed to the fact that the phase of impulse trains are being taken into consideration. It is our future work to use this model to explain (to some degree of accuracy) those phenonema in the brain in which timing of impulses or chaotic behaviros are important.

1 Phase Information and Complex Model

It is an elementary notion to analyze the response of an electric circuit to a sinusoidal input by expressing the circuit variables and constants by complex numbers. For example, in the complex voltage representation $E = Ae^{i\theta}$, the argument of E, θ represents the phase of the sinusoidal signal of amplitude A. Complex value representation of electric circuits greatly simplifies its analysis. In neural networks, we do not deal with sinusoidal waves but with impulse trains. An impulse train is somewhat similar to a sinusoidal wave in that it is a periodic signal with amplitude, frequency and phase. The absolute phase is irrelevant and we are mostly concerned with relative phase between impulse trains. When several impulse trains of no or little phase differences are added at the dendrite, they result in a strong membrane potential whereas they produce a much weaker signal when they have random phases. Although this situation cannot be accurately modeled by merely representing the signals by complex numbers as in the sinusoidal case, complex-number representation would at least be a better approximation to the real neuron than the conventional real-valued discrete-time model. There is no question that a better model can be obtained by a differential equation model. However, the latter poses a computational problem when many neurons are involved. Our complex neuron is a discrete-time model and thus the computational load is of the same order as the real-valued discrete-time model. Therefore, we propose that complex valued models be used in situations where timing in impulse trains is of considerable importance but the computational load of the differential equation models is forbidding.

Needless to say, the whole algorithm can be written in a real-valued formulation, and we do not use those concepts characteristic of complex-valued functions such as homeomorphism. Therefore, the point in using complex values lies in the fact that it makes the model look more natural and moreover, more accurate than the conventional discrete time real-valued model and yet the computational cost is not significantly greater.

2 Complex Perceptron

The results shown below are basically from our previous work (Nemoto and Kono, 1988, Nemoto and Kubono 1992). Since then we have seen that the performance of the complex associative memory is better than shown in these works. Some of the new results are included in thie section.

Let $w_j = \alpha_j + i\beta_j$, $j = 1,\ldots,n$ be the weights where $i^2 = -1$ and $\alpha_j, \beta_j \in \mathbf{R}$. Let $\mathbf{x} = \{-1,1\}^n$ denote the input pattern vector and $\mathbf{w} = (w_j)$ be the weight vector. Their inner product is denoted $\mathbf{w} \cdot \mathbf{x}$. The output of the cell is given by

$$o = \operatorname{sgn}(g(\mathbf{w}, \mathbf{x}, h)) \tag{1}$$

where

$$g(\mathbf{w}, \mathbf{x}, h) = | \mathbf{w} \cdot \mathbf{x} | - h \tag{2}$$

and

$$\operatorname{sgn}[u] = \begin{cases} 1, & u > 0 \\ -1, & u \le 0 \end{cases} \tag{3}$$

The learning rule for the coefficients is

$$\alpha_j(t+1) = \alpha_j(t) + \Delta\alpha_j(t+1), j = 1,\ldots,n \tag{4}$$

where

$$\Delta\alpha_j(t+1) = \varepsilon(y(t) - \operatorname{sgn}(g(\mathbf{w}(t), \mathbf{x}(t), h)))\frac{\partial g}{\partial \alpha_j} + \lambda\Delta\alpha_j(t)$$

in which $\varepsilon > 0$ and $\lambda > 0$ are constants and $y(t)$ is the desired output (1 or -1) for the pattern $\mathbf{x}(t)$ presented at time t. The last term represents the inertia. The learning rule for β_j is obtained by replacing α_j with β_j. The threshold h has the same form of learning rule as well.

The network used for simulation consisted of 50 input cells (thus 50 weights) and 1 output cells. One of the input cells always gives 1 and the real-valued weight connecting this cell to the output cell is considered as the threshold. Each element of an input pattern was assigned 1 or -1, each with probability 1/2. For k patterns, there are a total of 2^k dichotomies (the ways in which k patterns are divided into two classes), which are too many to be tested. Here we chose 50-70 dichotomies for testing which were quite sufficient for our purpose. Each dichotomy corresponds to an assignment of desired outputs for the given input patterns. Separability is the fraction of the dichotomies that were successfully separated. It is established (Cover, 1965) that there are $C(m,n)$ homogeneously linearly separable dichotomies of m points in general position in Euclidean n-space where

$$C(m,n) = 2\sum_{k=0}^{n-1}\binom{m-1}{k}. \tag{5}$$

Therefore, the probability that a randomly chosen dichotomy is separable is

$$P(m,n) = \frac{1}{2^{m-1}}\sum_{k=0}^{n-1}\binom{m-1}{k} \quad . \tag{6}$$

Figure 1 shows simulation result and the two curves for $n = 50$ and 100. (This was recently obtained and better than the result shown in Nemoto and Kubono 1992.) The curve $n = 50$ corresponds to the theoretical performance of the real-valued perceptron having 50 weights and the one for $n = 100$ is a hypothetical curve for the CP having twice the number of freedom in the weights compared to the real-valued perceptron. The simulation results show that the actual performance far exceeds the second hypothetical curve indicating the advantage of the use of complex values for the weights.

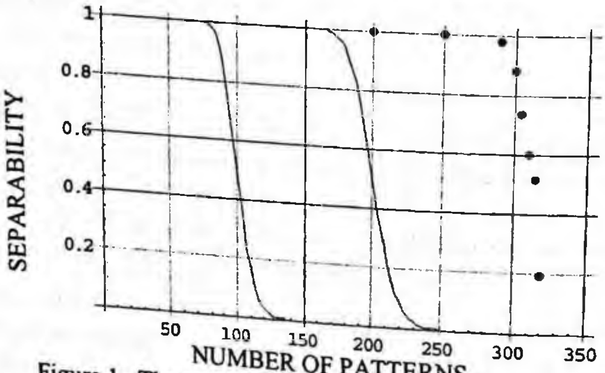


Figure 1. The separation power of the complex perceptron.

3 Complex Associative Memory

In the following model of complex associative memory (CAM), weights and membrane potential take complex values whereas the outputs take 1 and -1. Its biological implication would be that the synaptic transmission is accompanied by a delay that can be modified by learning whereas the output impulse trains are all synchronours. This might seem biologically unnatural. However, the main purpose of the present model is to investigate the complex associative memory as a direct extension of the complex perceptron. Note that if one is interested only in the fixed points realized by the CAM and not in the dynamics, phase differences among the outputs will have no significance.

The associative memory can be considered as a set of perceptrons producing ± 1 outputs forming the prescribed pattern. Thus, the learning rule of the perceptron is directly extended to the associative memory. The dynamics of the network is described by

$$x_j(t+1) = \mathrm{sgn}(u_j(t)), \ j = 1, \ldots, n \tag{7}$$

where

$$u_j(t) = g(\mathbf{w}_j, \mathbf{x}(t), h_j) = |\mathbf{w}_j \cdot \mathbf{x}(t)| - h_j$$

and $\mathbf{w}_j = (w_{j1}, \ldots, w_{jn})$ is the weight vector consisting of the complex weights to the j-th cell. At time t, one of the m patterns to be stored (called learning patterns or stored patterns depending on the context) is randomly chosen. Denote it by $\mathbf{s}(t)$. The learning rule for the real part α_{jk} of the weight w_{jk} is

$$\alpha_{jk}(t+1) = \alpha_{jk}(t) + \Delta\alpha_{jk}(t+1), j, k = 1, \ldots, n \tag{8}$$

where

$$\Delta\alpha_{jk}(t+1) = \lambda\Delta\alpha_{jk}(t) - \varepsilon(s_j(t) - f(u_j(t)))\left(\frac{df}{dt} + \eta\right)\frac{\partial u}{\partial \alpha_{jk}} \tag{9}$$

where $\varepsilon > 0$ is the learning rate and η a constant to avoid local minima (Tanaka and Komura (1992)). f is the sigmoid function given by

$$f(u) = \frac{2}{1+\exp(-cu)} - 1, \ c: \text{constant}.$$

The rule for the imaginary part is the same. The threshold $h_j(t)$ changes according to the same rule as well except that $\partial u / \partial h_j$ is set to 1.

It should be noted that the performance in terms of the number of fixed points (patterns) is already given by that for the CP because CAM is simply considered as n perceptrons in parallel, so we do not repeat it here.

Figure 2 shows the average attractivity (defined below) of the target patterns plotted against the number of stored patterns. Fifty distinct test patterns for each target pattern were made by reversing one bit of the target. The attractivity of the target is the fraction of its 50 test patterns that induced the network to output the target exactly. Average attractivity is the average of such attractivities taken over all the stored patterns used as targets. The curve designated as 'real, $\neq 0$' shows the result for the real-valued associative memory (RAM) with all the elements of **w** treated equally throughout the learning process. The one with 'real, $\equiv 0$' was obtained for the RAM with all the diagonal elements w_{ii} fixed to 0 from the beginning. The results for CAM are shown similarly. Here again, it is seen that the performance of CAM is better than expected from the mere increase (twice) in the number of parameters.

Figure 3 shows typical time courses of the direction cosine between the target pattern and the network output when at time 0 a test pattern with the direction cosine $dc(0)$ is given. The direction cosine between patterns $\mathbf{y}$ and $\mathbf{z}$ is defined to be $\mathbf{y} \cdot \mathbf{z} / n$. The storage ratio $r = m/n$ is 0.5 for all the examples shown. The dc curves for unsuccessful recollection by the CAM show somewhat chaotic

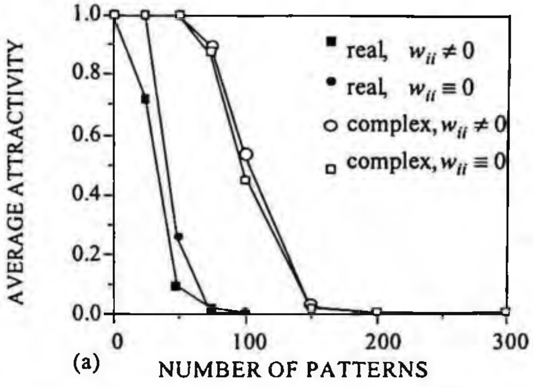


Figure 2. The average attractivity of the associative memory.

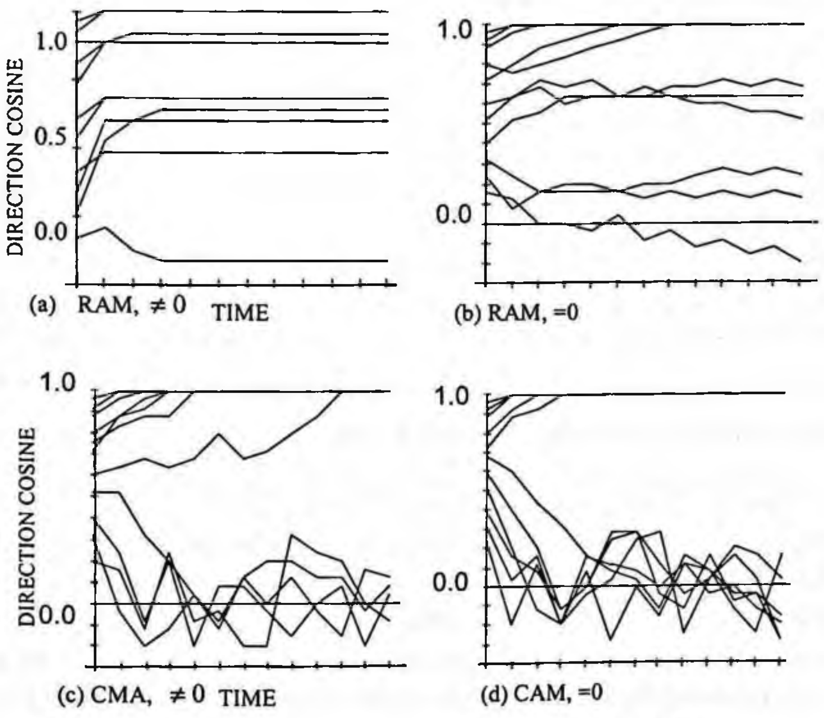


Figure 3. Time courses of direction cosine. =0 means $w_{ii} \equiv 0$, etc.

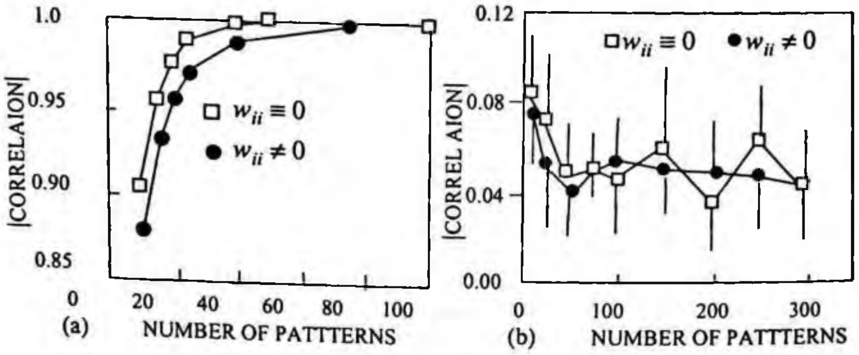


Figure 4. Correlation of the weights in RAM (left) and CAM (right).

behavior for both $w_{ii} \neq 0$ and $w_{ii} \equiv 0$. It was found that a typical (though not every) state transition for the CAM either converged to a fixed point after many steps of transition (sometimes more than 500 steps) or fell into a cycle of a very long period. This behavior is advantageous compared to the behavior such as shown in (a) because one would know when recollection is unsuccessful.

Figure 4 gives a rough idea on how definitely the weight matrices are determined by the stored patterns. It shows the average of the absolute value of the correlation coefficient ρ between two weight matrices (w_{ij}) and (w'_{ij}) obtained by learning the same set of patterns starting from uncorrelated initials states. ρ is defined by $\rho = \sum_{i,j}^{n} w_{ij}\overline{w}'_{ij} / \sqrt{\sum_{i,j}|w_{ij}|^2 \sum_{i,j}|\overline{w}'_{ij}|^2}$ where $^{-}$ stands for complex conjugate. Ten randomly made pairs of initial value sets were given and 10 pairs of weight matrices were obtained. Both schemes regarding w_{ii} were used. The correlation coefficients were calculated for the 10 pairs of weight matrices and their absolute values were averaged. The graph shows the averages and the standard deviations plotted against the number of the stored patterns. The RAM yielded very similar weight matrices for the same target patterns even when the storage ratio was low, and the matrices became identical when the storage ratio tended to 1. In the CAM, the wegith matrices showed almost no correlation when learning started from independnt initial values, even when the storage ratio was quite large.

4 Summary for CP and CAM

The separating power of the CP was more than predicted from the increase of the number of parameters of the network. This is due to the nonlinear dependence among the coefficients in the decision function (2). Indeed, if we put $y_{ij} = x_i x_j$, $w_{ij} = w_i \overline{w}_j$ then using eqn (2) as a decision function is equivalent to using $g'(\mathbf{w},\mathbf{y},h) = \sum_{ij} w_{ij} y_{ij}$. If w_{ij} were linearly related, Cover's theory shows that the separability would be determined by the degree of freedom enjoyed by w_{ij}. We do not even know whether the storage capacity is given by a linear relashionship $m = a\, n$. It is possible that the storage capacity is a nonlinear function of the number of weights.

We found several qualitative differences between the behavior of RAM and CAM. The time evolution of the direction cosine of the RAM looks quite flat both for $w_{ii} \equiv 0$ and $w_{ii} \neq 0$ and there is no clear-cut separation of time courses at the threshold $dc(0)$ value, which is the lowest initial direction cosine that leads the state to reach the target pattern (or very close to it). In the autocorrelation associative memory with 0 diagonal elements the threshold effect is clearly seen (Amari and Maginu 1988). The CAM had somewhat similar tendency and showed clearly divided time courses depending on the initial *dc* values. Time courses corresponding to unsuccessful recalls showed very different behavior from that of the RAM, with either a very long path to a fixed point or a very long limit cycle. This feature has some resemblance to chaotic associative memory (Adachi and Aihara, 1997), although the present model has a finite number of possible states and thus the period of orbit is bounded. The differences in the behavior of the weights was remarkable, too. These features may be more attractive than the increase of the storage capacity.

5 A Complex Version of the N-S Model

The model considered in this section is obtained by allowing the variables and constants in the Nagumo-Sato model of a single neuron to take complex values. This neuron receives an external input and its past output through a complex-valued weight and fires when the absolute value of the membrane potential exceeds a threshold. In the following, we present the basic features of the model, such as associated with fixed points and period-two orbits. We also show some interesting orbits of the model including chaotic behavior. The complex-valued output of the neuron at time t is given by

$$\xi^{(t)} = \Theta(\eta^{(t-1)}),\ \eta^{(t)} = A - \frac{1}{\alpha}\sum_{l=0}^{t}\beta^{l}\xi^{(t-l)} \tag{10}$$

where

$$\Theta(\eta) = \begin{cases} 0, & (|\eta| < 1) \\ \eta/|\eta| = e^{i\arg\eta}, & (|\eta| \geq 1) \end{cases}$$

and $\alpha > 0$. β and A are complex constants. If we convert the variables by $z^{(t)} = 1 + \alpha\beta A - \sum_{l=0}^{t}\beta^{l}\xi^{(t-l)}$ and let $c = 1 - \alpha A(1-\beta)$, then the original model is transformed into

$$z^{(t)} = f(z^{(t-1)}),$$
$$f(z) = \begin{cases} 1 + \beta(z-c), & (|z-c| < \alpha) \\ 1 + \beta(z-c) - \dfrac{z-c}{|z-c|}, & (|z-c| \geq \alpha) \end{cases} \tag{11}$$

We take c, α and β to be independent parameters. Derivation of the following properties is shown in Nemoto and Saito, 2002.

Simple calculation shows that:
Property 1. For any $\varepsilon>0$, any orbit eventually enters and then never exits
$C_{R_{b,c}+\varepsilon}=\{z:|z-c|<R_{b,c}+\varepsilon\}$ where $R_{b,c}=(1+|1-c|)/(1-b)$.

The following property makes the model somewhat simpler:
Property 2. The model dynamics has topological conjugacy expressed as:

$$f(\bar{z}\,|\,\bar{c},\bar{\beta})=\overline{f(z\,|\,c,\beta)} \text{ and } f(T_\tau z\,|\,T_\tau c,\beta)=T_\tau f(z\,|\,c,\beta)$$

where $^{-}$ stands for complex conjugate and T_τ is the operater which causes the rotation by angle τ around the point 1. Hence, we can take as our parameter space

$$\mathbf{S}=\{(\alpha,\beta,c):\ \alpha>0,\ c\le 1,\ |\,\beta\,|<1, 0\le \arg\beta\le\pi\}.$$

For $r,R>0$, let $C_r=\{z:|z-c|<r\}$, $\Gamma_r=\{z:\ |z-1|<R\}$ and $\partial C_r, \partial\Gamma_R$ be their boundaries. Then, the following property makes the intuitive comprehension of the mapping f easier:
Property 3.

(i) A circle C_r centered at c is mapped by f to the circle $\Gamma_{R(r)}$ centered around 1 where

$$R(r)=\begin{cases} rb, & r<\alpha \\ \sqrt{1+r^2b^2-2r\beta_R}, & r\ge\alpha \end{cases}$$

(ii) A line in a radial direction of C_r is mapped to a line.

From the above property, it is seen that the circle C_α plays an important role. We call it the *critical circle*. A point z satisfying $f(z)=z$ is called a fixed point. Fixed points and their stability are

quite easily obtained in closed form. We have the following regarding fixed points:
Property 4. For a point in the parameter space S, there is at most one (inner or outer) fixed point. Inner fixed points are stable. If there is an outer fixed point, then $c \leq 1$. The dynamics of this model involves periodic behavior. A period-2 point is defined to satisfy $f^2(z) = z$. It is easily seen that both of the two points forming a period-2 orbit cannot lie within the critical circle and we have:
Property 5. There are two types of period-two orbits, in-out and out-out ('in (out)' meaning in(out)side-the-critical-circle) type. Their traces can be calculated as functions of θ for given α, c, b. 'out-out' orbits may be born from outer fixed points.

Also, we often encounter closed-curve orbits in the dynamics. If we set $c = 1$, then we have a circular orbit centered at 1. Indeed, we can prove:
Property 6. When $c = 1$, a stable periodic circular orbit of arbitrary period can be realized on the circle centered at 1 with radius r_c. Further, on the same circle, quasiperiodic orbit with arbitrary constant angular increment Δ can be constructed.

Figure 5 shows a transition of the dynamics for $\alpha = 0.1, c = -1$, $b = 0.5$ as $\theta = \arg \beta$ increases from 0 to π. For $0 \leq \theta \leq 120^\circ$, the dynamics has a fixed point, and then it experiences a bifurcation into period-two mode at around $\theta = 102.5^\circ$. It eventually enters a closed-curve orbit and finally gets back to the period-two mode. Figure 6 (left) shows an example of chaotic orbit (20000 orbit points starting from $z(0) = 0.5$) with the parameter values $\alpha = 0.1, c = 0.3, \beta = 0.98 e^{i\pi/18}$. The larger one of the two Lyapunov exponents is 0.105 and high sensitivity of the dynamics to the initial condition was experimentally shown and thus the orbit is very likely of chaos. In the right is shown one of its two period-two saddles (p=0.641065+0.4434381i) and the stable (Ms) and

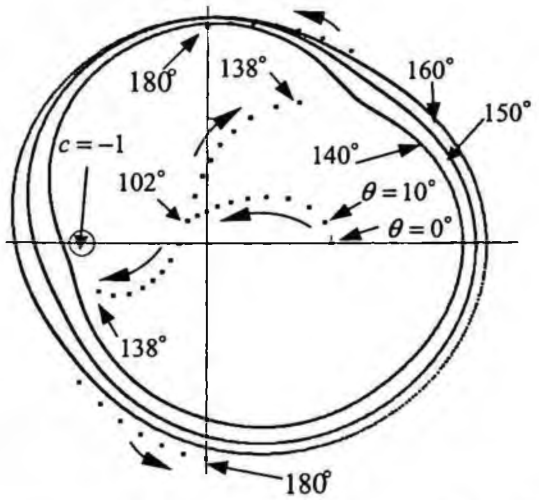


Figure 5. Transition of the dynamics for $0 \leq \arg \beta \leq \pi$.

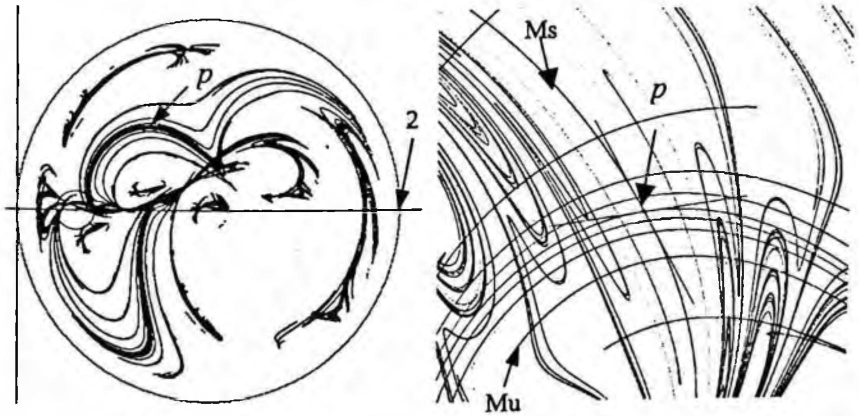


Figure 6. Chaotic dynamics (left) and the stable and unstable manifolds at its period-two point p (right).

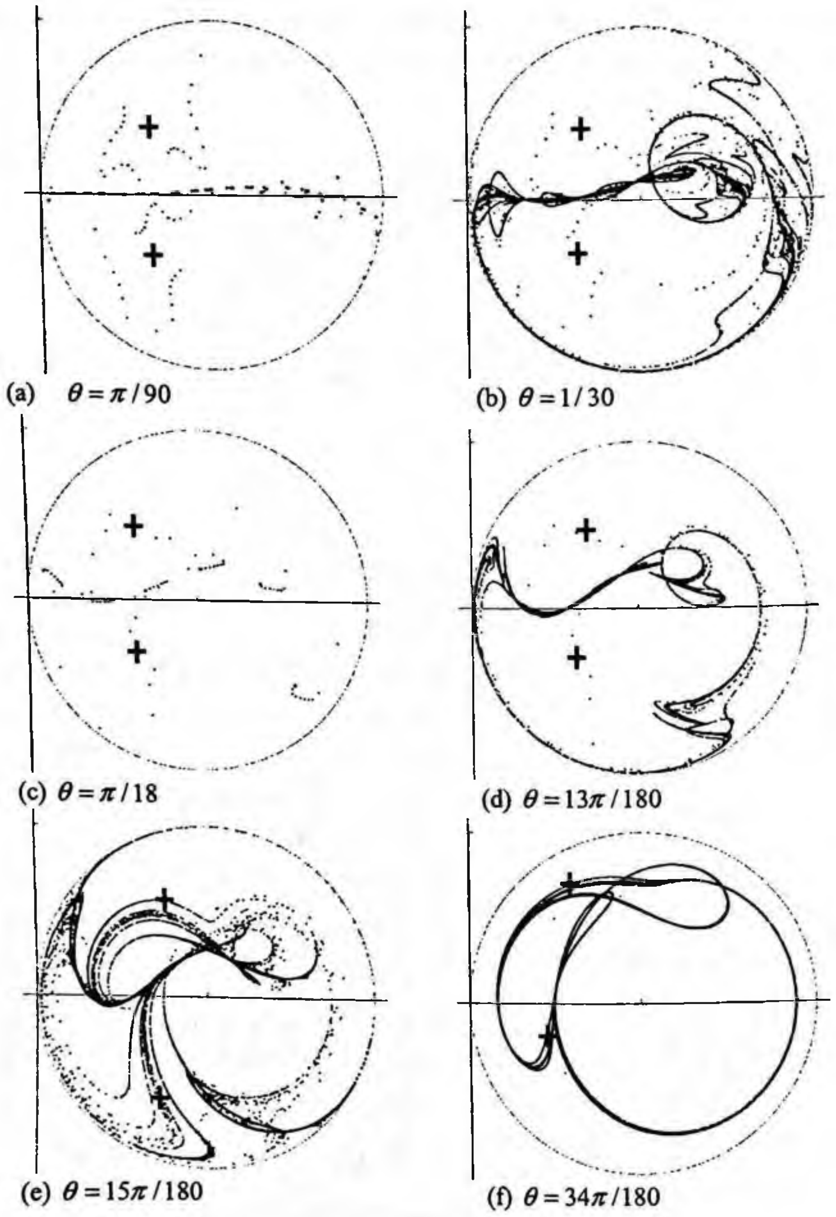


Figure 7. Orbits of the complex NS model for various values of θ.

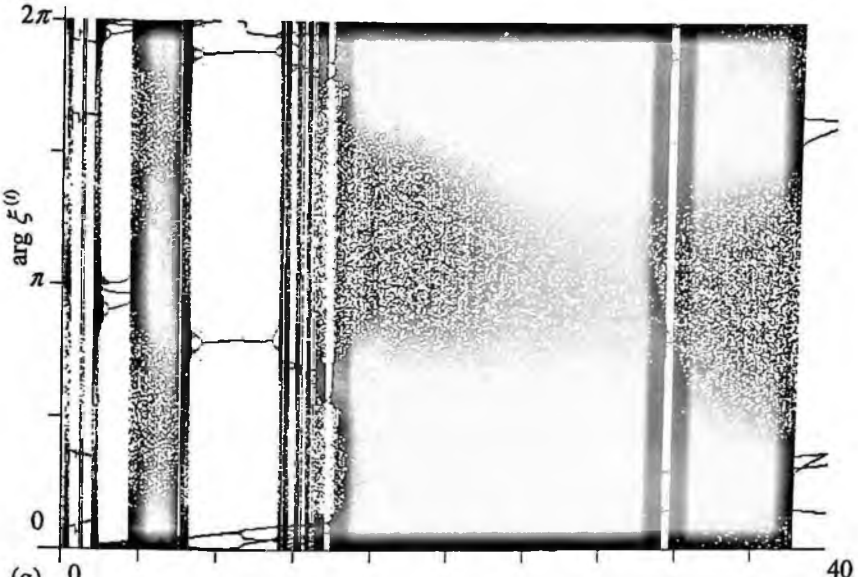


Figure 7. Continued.

unstable (Mu) manifolds associated with it. It is seen that homoclinic points (intersection points of the two manifolds) seem to accumulate to the saddle. Therefore, in this case it is likely that the chaotic behavior develops through period-two saddles, although there were many situations where neither unstable fixed points nor unstable period-two points caused chaotic behavior. The cause of chaotic behavior of the model has not been theoretically elucidated.

Figure 7 shows orbits starting from $0.5+0.5i$ and consisting of 50,000 points for six values of θ where $\alpha = 0, |\beta| = 0.98$. The initial point for the orbit did not make essential difference to the result in this figure. All the orbits are shown with the period-two points of the map. In (a) they are stable periodic points and have a basin of attraction. Those in (b)-(f) are all period-two saddles. Those in (e) and (f) look closely associated with the apparently chaotic orbits. Therefore, the seemingly chaotic behavior in (e) and (f) are probably related to the period-two saddles. (g) shows the corresponding

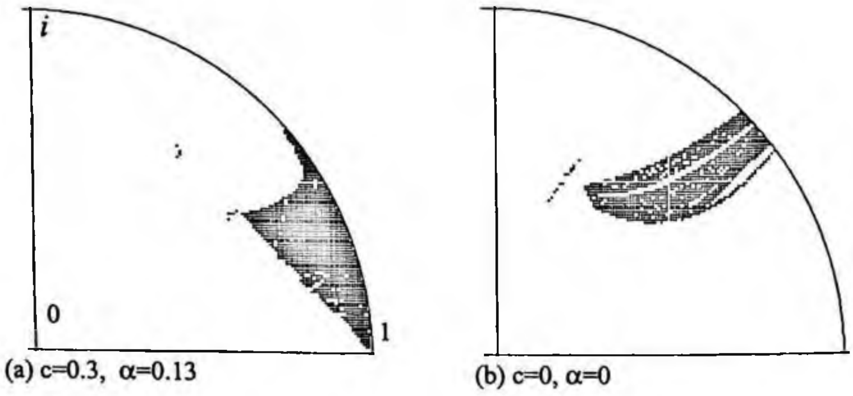


Figure 8. The values of β shown in the complex plane such that the orbit is chaotic (positive Lyapunov exponent).

bifurcation diagram where the horizontal line represents the value of θ and the vertical axis shows $\arg \xi^{(t)}, t = 5000,...,10000$.

The seemingly chaotic behavior certainly is an interesting aspect of the complex N-S model which should further be studied more theoretically. However, it is not a prevailing aspect of the dynamics. Figure 8 shows the values of β in the complex plane for which the larger of the Lyapunov exponents is larger than 0.01 and the orbit looks chaotic.

6 A Two-Neuron Network

Next we show some behavior of a network consisting of 2 complex-valued Nagumo-Sato neurons. In particular, we show the orbits when the two neurons are excited by two impulse trains having a phase difference between. The results we so far obtained show that the relationship between the phase difference of the membrane potentials of the two neurons and that in the input stimuli is often quite complicated. However, its possible indication in the corresponding brain activity is not clear and not discussed here.

The dynamics of the network is described by

$$\eta_i(n) = A_i - \frac{1}{\alpha_i}\sum_{l=0}^{n} \beta_i{}^l \xi_i(n-l) + \sum_{j=1}^{N} w_{ij} \sum_{l=0}^{n} \gamma_i{}^l \xi_j(n-l) \tag{12}$$

$$\xi_i(n+1) = \Theta(\eta_i(n)) = \begin{cases} 0 & (|\eta_i(n)| < 1) \\ \dfrac{\eta_i(n)}{|\eta_i(n)|} & (|\eta_i(n)| \geq 1) \end{cases} \quad i = 1,2,\ldots,N \tag{13}$$

where $\beta_i, \gamma_i, c_i, A_i \in \mathbf{C}, \alpha_i > 0, N = 2$ and $w_{ij} \in \mathbf{R}$. Let

$$\zeta_i(n) = 1 + \alpha_i \beta_i A_i - \sum_{l=0}^{n-1} \beta_i \xi_i(n-l)$$
$$\delta_i(n) = \sum_{j=1}^{N} w_{ij} \sum_{l=0}^{n} \gamma_i{}^l \xi_j(n-l)$$
$$z_i(n) = \zeta_i(n) + \alpha_i \delta_i(n)$$

Then, we get the dynamical system:

$$\zeta_i(n) - 1 = \begin{cases} \beta_i(\zeta_i(n-1) - c_i), & |z_i(n-1) - c_i| < \alpha_i \\ \beta_i(\zeta_i(n-1) - c_i) \dfrac{z_i(n-1) - c_i}{|z_i(n-1) - c_i|}, & |z_i(n-1) - c_i| \geq \alpha_i \end{cases} \tag{14}$$

$$\delta_i(n) = \gamma_i \delta_i(n-1) + \sum_{j=1}^{N} w_{ij} \Theta_j \tag{15}$$

$$\Theta_j = \begin{cases} 0, & |z_i(n-1) - c_i| < \alpha_i \\ \dfrac{z_i(n-1) - c_i}{|z_i(n-1) - c_i|}, & |z_i(n-1) - c_i| \geq \alpha_i \end{cases}$$

$$z_i(n) = \zeta_i(n) + \alpha_i \delta_i(n) \tag{16}$$

When $|z_i(n)-c_i| \ge \alpha_i$, the i-th neuron is considered to fire with the phase of the impulse train being $\arg(z_i(n)-1)$. If $|z_i(n)-c_i| < \alpha_i$, then it is in the resting state.

$$A_k = |A_k| \exp\{i\rho_k\}, \; k=1,\dots, N \tag{17}$$

is the input impulse train to the k-th neuron where $|A_k|$ is the amplitude (or alternatively the frequency) and ρ_k the phase of the impulse train. The model we study here consists of 2 neurons and satisfies:

$$N=2, \; \beta_k=\beta, \; \gamma_k=\gamma, \; k=1,2 \tag{18}$$

Assume that $A_1 = A, \; A_2 = Ae^{i\Phi}, A \in \mathbf{R}$, i.e., the two neurons receive input of the same amplitude but with phase difference Φ. Figure 9 shows an example response of the circuit to such stimulus. The parameter values are:

$$\alpha=0.2, A=4, \Phi=2\pi/9, \beta=0.8e^{i2\pi/9}, \;\; \gamma=0.8e^{2\pi i/90},$$
$$w_{ii}=0.5, \; w_{ij}=-1, i,j=1,2.$$

Figure 10 shows the behavior of the network when Φ (horizontal axis) is varied. The vertical axis shows

$$\Delta(n)=\arg z_1(n)-\arg z_2(n), \;\; n=K-1000,\dots K$$

where K takes a sufficiently large number such that the behavior is well depicted for a particular set of parameter values. In (a) Δ is almost linearly related to Φ for all its values. The orbit in this case is of period-2 and z_1 and z_2 take the two periodic points in turn. In (b), the dynamics seems to have a pseudo-periodic orbits when Φ is between 0.8 π and 1.4 π. For the rest of the values of Φ, the dynamics again is of periodic orbits of very large periods. The orbits corresponding to (c) were chaotic for most of the values of Φ. The relationship between Δ and Φ looks quite random but still

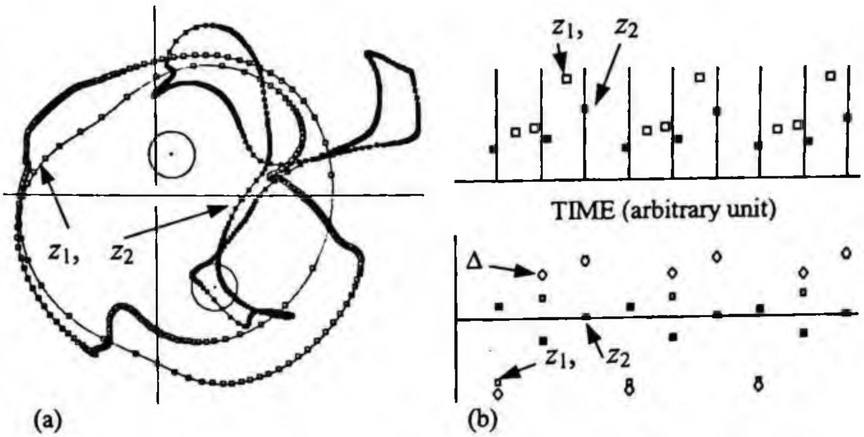


Figure 9. Response of the network to 2 inputs of the same amplitude with a phase difference. The lower panel of (b) shows the phases of z_1, z_2 and the difference $\Delta = z_1 - z_2$ between indicated by the ellipses.

seems to preserve some linear tendency seen in (a). The parameter values used are:

(a): $\alpha = 0.1$, $A = 10$, $\beta = 0.4e^{\pi i}$, $\gamma = 0.5$, $w_{ij} = 0.5, i, j = 1,2$;

(b): $\alpha = 0.1$, $A = 10$, $\beta = 0.4e^{\pi i/4}$, $\gamma = 0.5$, $w_{ij} = 0.5, i, j = 1,2$;

(c) $\alpha = 0.1$, $A = 79$, $\beta = 0.98e^{i\pi/72}$, $\gamma = 0.5e^{i\pi/18}$,
$w_{11} = w_{22} = w_{12} = 0.5$, $w_{21} = 0.5$.

The examples in Figures 9 and 10 show that the behavior of this very simple network consisting of only two elements can be very complicated. Although such behavior has not been associated with any brain process yet, it shows that combinations of complex-valued neuron models may have a much stronger power to represent states of neural networks in the brain than the real-valued models.

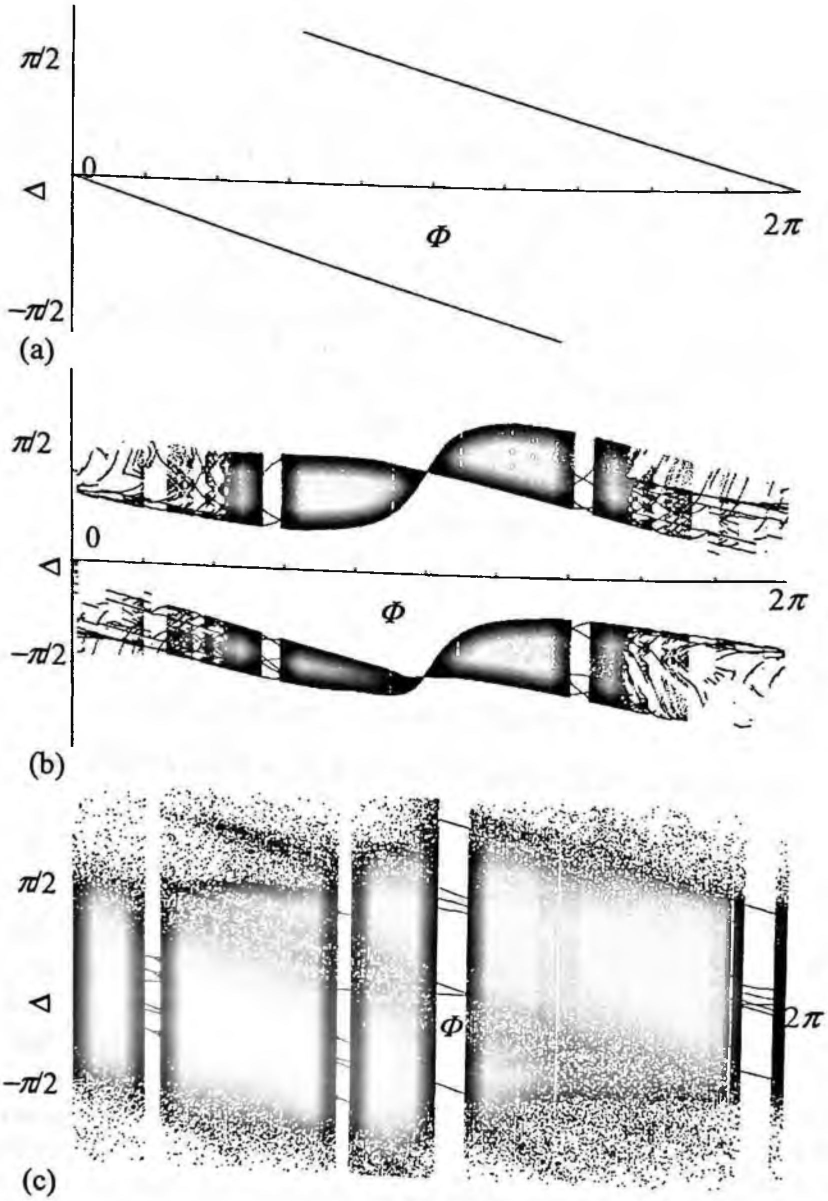


Figure 10. The relationship between input and output phase differences.

7 Concluding Remarks

In this chapter we looked at two aspects of the use of complex numbers in neural networks. One is the complex associative memory (CAM) and the other is the complex-valued Nagumo-Sato model (CNS) of a single neuron. Both models use complex numbers to represent impulse trains, treating as if they were sinusoidal waves. The modulus of the number is the amplitude (or it can be the frequency) and the argument angle is the phase of an impulse train. The CAM shows some definite advantage over RAM in the storage capacity and the dynamics of recalling stored patterns. An interesting problem would be to derive storage capacity of the CAM theoretically. On the other hand, the CNS and its extension to two-element model have been investigated of their behaviors without specific reference to their utility. Their apparently chaotic behaviors have been emphasized in this chapter although other properties have to be more investigated as well, because chaotic behaviors attract our attention and also because chaotic behaviors in the brain have been discussed and considered important by many researchers recently.

Aihara *et al.* 1990 introduced the logistic function instead of the original step function into the Nagumo-Sato model, which made the chaotic behavior intrinsically involved in the original model explicitly observable. Our use of complex values in the Nagumo-Sato model brought about a different kind of chaos which is not intrinsic in the original model. Freeman (1995) and others proposed that in many aspects of the brain function, chaos can play important roles, such as in keeping a sensory system at high sensitivity, providing itinerant state transactions during perception. They emphasize that autonomy and creative power of the biological intelligence may be ascribed to chaos. Aihara and others have continuously worked on the role of chaos in artificial neural networks for associative memory.

Putting aside chaotic behavior of neurons, temporal coding in impulse trains has been attracting many researchers. Among others Eckhorn *et al.* 1988 found the importance of synchronization of impulse trains in visual perception in the brain. More recently, phase relationship of the firings of the so-called place cells in the hippocampus with respect to EEG theta wave have been shown to play an important role in locating the position of the animal itself when it is moving in an area. The use of complex values in neural networks will probably be able to make simple models for these neural phenomena in which timing in impulse train is important.

References

Adachi, M. and Aihara, K. (1997), "Associative dynamics in a chaotic neural network," *Neural Networks,* vol. 10, pp. 83-98.

Amari, S. and Maginu, K. (1988), "Statistical neurodynamics of associative memory," *Neural Networks,* vol. 1, pp. 63-73.

Nemoto, I. and Kono, T. (1992), "Complex neural networks," *Systems and Computers in Japan,* vol. 23, pp.5-84.

Nemoto, I. and Kubono, K. (2001), "Complex-valued Nagumo-Sato model of a single neuron," *Neural Networks,* vol. 9, pp. 253-261.

Nemoto, I. and Saito, K. (2002), "A complex-valued version of Nagumo-Sato model of a sigle nheuron and its behavior," *Neural Networks,* vol. 15, pp. 833-853.

O'Keefe J. and Recce, M.L. (1993), "Phase relationship between hippocampal place units and the EEG theta rhythm," *Hyppocampus,* vol. 3, pp. 317-330.

Tsuda, I. (1992), "Dynamic link of memory – Chaotic memory map in nonequilibrium neural networks," *Neural Networks*, vol. 5, pp. 313-326.

Author's address
Iku Nemoto: Dept. Information Environment Design and Integration, Tokyo Denki University, Muzai-gakuendai, 2-1200, Inzai, Chiba, 270-1382, Japan.

Chapter 7

Data-Reusing Algorithm for Complex-Valued Adaptive Filters

Danilo P. Mandic, Su Lee Goh, and Andrew Hanna

A class of data-reusing (DR) learning algorithms for complex-valued linear and nonlinear adaptive filters is analyzed. This class of algorithms has an improved convergence over the standard algorithms by virtue of re-using of the external input data while performing iterations on weight adaptation. The class of algorithms are introduced starting from the case of linear adaptive filters trained with the complex-valued least mean square (CLMS) algorithm, through to the case of feedforward and recurrent neural networks employed as nonlinear adaptive filters trained with a complex-valued gradient descent (CGD) learning algorithm and a complex-valued real time recurrent (CRTRL) learning algorithm, respectively. Both the error bounds and convergence conditions are provided for the case of contractive and expansive complex activation functions. The improved local performance of the complex-valued data-reusing algorithm over the standard algorithms is verified by simulations on the prediction of linear and nonlinear complex-valued signals.

1 Introduction

There has been recent interest in the research of complex-valued adaptive filters due mainly to the increasing trend of processing complex-valued signals in modern disciplines (e.g. satellite communications) (Mandic and Chambers 2001), (Kim and Adali 2001). To this cause, a considerable research effort has been directed towards extending real-valued adaptive filters to the complex plane, $\mathbb{C}$. The complex least mean square (CLMS) algorithm (Widrow *et al.* 1975), gave rise to applications of complex-valued linear filters. This in turn led to the development of complex-valued nonlinear algorithms, such as the complex-valued nonlinear gradient descent (CNGD) algorithm (Hanna and Mandic 2002), complex backpropagation algorithm (CBP), (Hirose 1990), (Benvenuto and Piazza 1992), (Georgiou and Koutsougeras 1992), and the complex real time recurrent learning (CRTRL) algorithm (Kechriotis and Manolakos 1994). This family of complex-valued algorithms suffer from the same well known problems of slow convergence and a tendency to converge to local minima of the error performance surface as their real-valued counterparts. In the field of real-valued nonlinear adaptive filtering, the family of data-reusing (DR) algorithms is employed to help speed up convergence. Here, following that concept, we extend the data-reusing real-valued adaptation filters to the complex domain.

Increasing the speed of convergence of adaptive filters often implies a corresponding increase in computational complexity of the adaptation algorithm. For many applications, the class of gradient-type algorithms is not fast enough for a satisfactory performance, and Newton-type algorithms are too computationally complex. Recently, there has been significant interest in the so-called data-reusing (DR) approach which aims to provide a compromise solution to improve the speed of convergence of a gradient-type algorithm while keeping the extra computation to a minimum. The DR algorithm proposed in (Schnaufer and Jenkins 1993) is a modification of the well known LMS algorithm. At every discrete time instant, k, the DRLMS algorithm reuses the current desired response and the current input vector to update the filter coefficients (weights) several times every iteration. This results in the commonly named *a posteriori* updates. For linear filters, this class of algorithms offers an increased rate of convergence when compared to the standard LMS algorithm (Treichler *et al.* 1987), (Roy and Shynk 1989), (Schnaufer and Jenkins 1993). However, for the case of real and complex-valued nonlinear adaptive filters, the increase in local and global performance is dependent on the choice of the activation function.

2 Complex Linear Adaptive Filters

The LMS adaptive filtering algorithm is one of the most common approaches to linear adaptive filtering problems. The algorithm is computationally inexpensive and simple to implement. However, although robust, this algorithm is relatively slow at converging to the global solution.

2.1 Complex-Valued LMS Algorithm

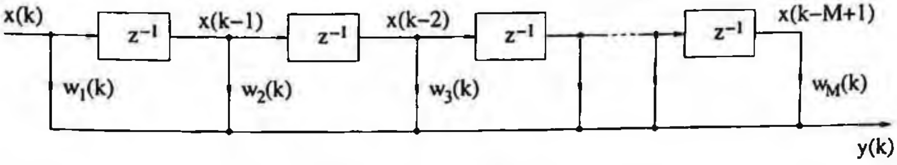


Figure 1. Linear adaptive finite impulse response filter.

Figure 1 shows the layout of the finite impulse response (FIR) filter, for which the output of the filter is given by

$$y(k) = \mathbf{w}^T(k)\mathbf{x}(k), \tag{1}$$

where $\mathbf{x}(k) \triangleq [x_1(k), \ldots, x_M(k)]^T$ denotes the complex input signal vector, $\mathbf{w}(k) \triangleq [w_1(k), \ldots, w_M(k)]^T$ the complex weight vector, M the number of tap inputs, and $(\cdot)^T$ denotes the vector transpose operator. The error signal $e(k)$ required for adaptation is obtained as the difference between the desired response $d(k)$ and the output of the filter $y(k)$, given as

$$e(k) = d(k) - y(k) = e^r(k) + je^i(k), \tag{2}$$

where $j = \sqrt{-1}$ and the superscripts $(\cdot)^r$ and $(\cdot)^i$ denote the real and imaginary parts respectively. The weight adaptation in the complex gradient descent algorithm is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \nabla_{\mathbf{w}} J(k)_{|\mathbf{w}=\mathbf{w}(k)}, \tag{3}$$

where μ is the rate of adaptation and $J(k)$ is the cost function of the network defined by

$$J(k) = \frac{1}{2}|e(k)|^2 = \frac{1}{2}\left[e(k)e^*(k)\right], \tag{4}$$

and the operator $(\cdot)^*$ denotes the complex conjugate. The gradient of the cost function with respect to the complex-valued weight can be computed as (Widrow *et al.* 1975)

$$\nabla_{\mathbf{w}} J(k) = \frac{\partial J(k)}{\partial \mathbf{w}^r(k)} + j\frac{\partial J(k)}{\partial \mathbf{w}^i(k)}. \tag{5}$$

Calculating the gradient of the objective function with respect to the real part of the complex weight vector gives,

$$\begin{aligned} \nabla_{\mathbf{w}^r} J(k) &= \frac{\partial J(k)}{\partial \mathbf{w}^r(k)} = \frac{1}{2}\frac{\partial\left[e(k)e^*(k)\right]}{\partial \mathbf{w}^r(k)} \\ &= -\frac{1}{2}e(k)\mathbf{x}^*(k) - \frac{1}{2}e^*(k)\mathbf{x}(k). \end{aligned} \tag{6}$$

Similarly, the derivation of the objective function with respect to the imaginary part of the weight vector yields,

$$\nabla_{\mathbf{w}^i} J(k) = \frac{j}{2}e(k)\mathbf{x}^*(k) - \frac{j}{2}e^*(k)\mathbf{x}(k) \tag{7}$$

therefore, substituting equations (6) and (7) into (5) gives,

$$\begin{aligned} \nabla_{\mathbf{w}} J(k) &= \nabla_{\mathbf{w}^r} J(k) + j\nabla_{\mathbf{w}^i} J(k) \\ &= -e(k)\mathbf{x}^*(k). \end{aligned}$$

Finally the weight update can be computed as (Widrow *et al.* 1975),

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) - \mu \nabla_{\mathbf{w}} J(k) \\ &= \mathbf{w}(k) + \mu e(k)\mathbf{x}^*(k), \end{aligned} \tag{8}$$

which gives the weight update term for the complex-valued gradient descent based least mean square (CLMS) algorithm. We now show that this algorithm can be extended to conform to the analysis of the class of data-reusing (DR) algorithms in the complex domain.

2.2 The DRCLMS Algorithm

The DRLMS algorithm was first introduced in (Schnaufer and Jenkins 1993), and in some recent work, it was shown that the normalised LMS (NLMS) algorithm was closely related to the DRLMS algorithm, and could be interpreted as a limiting case of the latter (Roy and Shynk 1989). The NLMS algorithm offers low-computational complexity but due to the division in the learning rate of the NLMS algorithm, small values of the instantaneous input vector can cause the learning rate to grow very large and hence lead to instability and problems of dynamic range.

Following the approach from (Roy and Shynk 1989), it is clear that the weight update in the data-reusing complex-valued LMS (CLMS) algorithm can be written as

$$\mathbf{w}_{t+1}(k) = \mathbf{w}_t(k) + \mu e_t(k)\mathbf{x}^*(k) \tag{9}$$

$$e_t(k) = d(k) - \mathbf{w}_t^T(k)\mathbf{x}(k), \quad t = 1, \dots, L \tag{10}$$

where $\mathbf{w}_1(k) = \mathbf{w}(k)$, $\mathbf{w}_{L+1}(k) = \mathbf{w}(k+1)$ and t represents the current data-reuse iteration. Letting $L = 1$, reduces equations (9) and (10) to the standard CLMS algorithm given in the previous section. Conforming with the analysis in (Roy and Shynk 1989), we can now analyse the weight update as

$$\begin{aligned}\mathbf{w}(k+1)=\mathbf{w}_{L+1}(k) &= \mathbf{w}_L(k)+\mu e_L(k)\mathbf{x}^*(k)\\ &= \mathbf{w}_{L-1}(k)+\mu(e_{L-1}(k)+e_L(k))\mathbf{x}^*(k)\\ &= \mathbf{w}(k)+\mu\left[\sum_{t=1}^{L}e_t(k)\right]\mathbf{x}^*(k).\end{aligned}\tag{11}$$

To evaluate the error term, for $t=2$ we have,

$$\begin{aligned}e_2(k) &= d(k)-\mathbf{w}_2^T(k)\mathbf{x}(k)\\ &= d(k)-\left[\mathbf{w}_1^T(k)+\mu e_1(k)\mathbf{x}^H(k)\right]\mathbf{x}(k)\\ &= e_1(k)-\mu\left[\mathbf{x}^H(k)\mathbf{x}(k)\right]e_1(k)\\ &= e_1(k)\left[1-\mu\left(\mathbf{x}^H(k)\mathbf{x}(k)\right)\right],\end{aligned}\tag{12}$$

where $(\cdot)^H$ represents the Hermitian transpose operator. Therefore, the t-th data-reusing error can be presented as

$$e_t(k)=e(k)\left[1-\mu\left(\mathbf{x}^H(k)\mathbf{x}(k)\right)\right]^{t-1},\ t=1,\ldots,L \tag{13}$$

From (13), the total error $\sum_{t=1}^{L}e_t(k)$ from (11) can be expressed as

$$\begin{aligned}\sum_{t=1}^{L}e_t(k) &= \sum_{t=1}^{L}e(k)\left[1-\mu\left(\mathbf{x}^H(k)\mathbf{x}(k)\right)\right]^{t-1}\\ &= \frac{e(k)\left[1-\left(1-\mu\left(\mathbf{x}^H(k)\mathbf{x}(k)\right)\right)^L\right]}{\mu\left(\mathbf{x}^H(k)\mathbf{x}(k)\right)}.\end{aligned}\tag{14}$$

Finally, rephrasing equation (11) we can write the DR weight update, for a complex LMS algorithm as (Roy and Shynk 1989),

$$\mathbf{w}(k+1)=\mathbf{w}(k)+\frac{1-\left(1-\mu\left(\mathbf{x}^H(k)\mathbf{x}(k)\right)\right)^L}{\left(\mathbf{x}^H(k)\mathbf{x}(k)\right)}e(k)\mathbf{x}^*(k). \tag{15}$$

Figure 2. gives the geometric representation of the complex-valued data-reusing gradient algorithm for online training of adaptive filters.

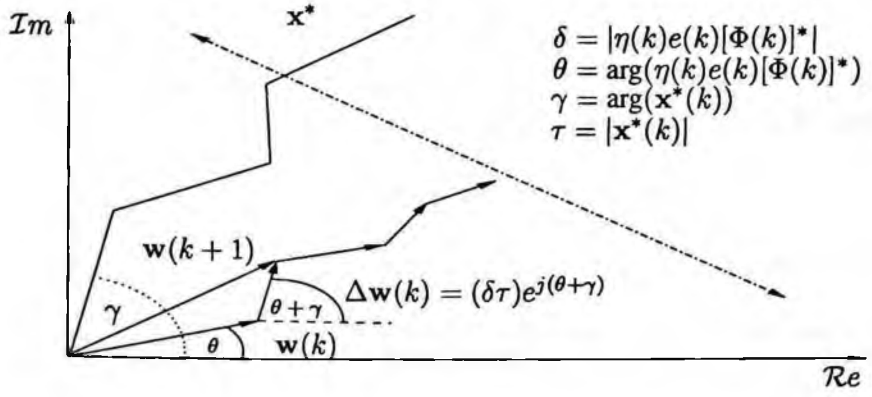

$\mathcal{I}m$
$\mathbf{x}^*$
$\delta = |\eta(k)e(k)[\Phi(k)]^*|$
$\theta = \arg(\eta(k)e(k)[\Phi(k)]^*)$
$\gamma = \arg(\mathbf{x}^*(k))$
$\tau = |\mathbf{x}^*(k)|$
$\mathbf{w}(k+1)$
γ
$\theta+\gamma$
$\Delta\mathbf{w}(k) = (\delta\tau)e^{j(\theta+\gamma)}$
θ
$\mathbf{w}(k)$
$\mathcal{R}e$

Figure 2. Geometric representation of the complex-valued weight update.

2.3 The Convergence Analysis of Linear Adaptive Filters DRCLMS

It is expected that the instantaneous *a posteriori* (data-reusing) error, $e_t(k)$, $t = 2, \ldots, L$, be smaller in magnitude than the corresponding *a priori* error $e_1(k)$ thus resulting in improved convergence and accuracy. In order to ensure proper behavior of the algorithm in the data-reusing phase, we aim to preserve the error to be uniformly decaying, that is

$$|e_t(k)| \leq \gamma\,|e_{t-1}(k)|, \quad 0 < \gamma < 1, \quad t = 2, \ldots, L \tag{16}$$

at each iteration of the data-reusing algorithm. The stability of a linear adaptive filtering algorithm can be determined from the relationship between the *a priori* and *a posteriori* error. It is essential to determine algorithms that guarantee stability for a large range of choices of the learning rate $\mu(k)$. Such algorithms employ the *a posteriori* error directly within the coefficient updates. The *a posteriori* adaptation for the DRCLMS algorithm is given as (Douglas and Rupp 1997),

$$\mathbf{w}_{t+1}(k) = \mathbf{w}_t(k) + \mu F(e_t(k))\mathbf{x}^*(k) \tag{17}$$

$$e_t(k) = d(k) - \mathbf{w}_t^T(k)\mathbf{x}(k), \quad t = 1, \ldots, L \tag{18}$$

where the *a posteriori* data-reusing error $e_t(k)$ is as defined in equation (18) and $F(e_t(k))$ is any function satisfying

$$sgn(F(e)) = sgn(e), \quad \text{and} \quad |F(e)| \geq \delta\,|e| \tag{19}$$

for some $\delta > 0$[1].

We can now state that the *a posteriori* estimation error obtained by algorithm (17) is

$$e_{L+1}(k) \geq \left[1 - \mu(k)\mathbf{x}^T(k)\mathbf{x}^*(k)\right]^L e_L(k) \tag{20}$$

To show this, we pre-multiply both sides of equation (17) by $\mathbf{x}^T(k)$ and subtracting $d(k)$ from both sides, we arrive at

$$e_{t+1}(k) = e_t(k) - \mu(k)F(e_t(k))\mathbf{x}^T(k)\mathbf{x}^*(k) \tag{21}$$

Since $sgn(F(e)) = sgn(e)$, we can deduce that $sgn(e_{t+1}) = sgn(e_t)$ when $\mu(k) > 0$, thus[2]

$$|e_{t+1}(k)| \geq |e_t(k)| - \mu(k)\,|F(e_t(k))|\,\left|\mathbf{x}^T(k)\mathbf{x}^*(k)\right|, \tag{22}$$

and using equation (19) we obtain

$$e_{t+1}(k) \geq \left[1 - \mu(k)\mathbf{x}^T(k)\mathbf{x}^*(k)\right] e_t(k). \tag{23}$$

Here, we assume that $F(e_t)$ does not have a significant change in direction or magnitude at each iteration, $t = 1, \ldots, L$. Therefore, we can state that,

$$e_{L+1}(k) \geq \left[1 - \mu(k)\mathbf{x}^T(k)\mathbf{x}^*(k)\right]^L e_L(k), \tag{24}$$

[1]Let $z \in \mathbb{C}$ and $F(\cdot)$ be some function. Then $sgn(F(z)) = sgn(z)$ iff $sgn(F^r(z)) = sgn(z^r)$ and $sgn(F^i(z)) = sgn(z^i)$.

[2]Let $\alpha, \beta \in \mathbb{R}^2$, then $|\alpha - \beta| \geq |\alpha| - |\beta|$, and $|\alpha\beta| = |\alpha||\beta|$

which is the bound for the error adaption for the DRCLMS algorithm.□

In order to maintain the error to be decreasing as shown in equation (16), the term $\left[1 - \mu(k)\mathbf{x}^T(k)\mathbf{x}^*(k)\right]$ must have the $\mathcal{L}_2$ norm less than unity. This way, we could set the range of the learning rate which guarantees stability of the algorithm. This range is given as

$$0 < \mu(k) < \frac{1}{\mathbf{x}^T(k)\mathbf{x}^*(k)}, \tag{25}$$

where $\mu(k) \in \mathbb{R}$ since $\mathbf{x}^T(k)\mathbf{x}^*(k) \in \mathbb{R}$.

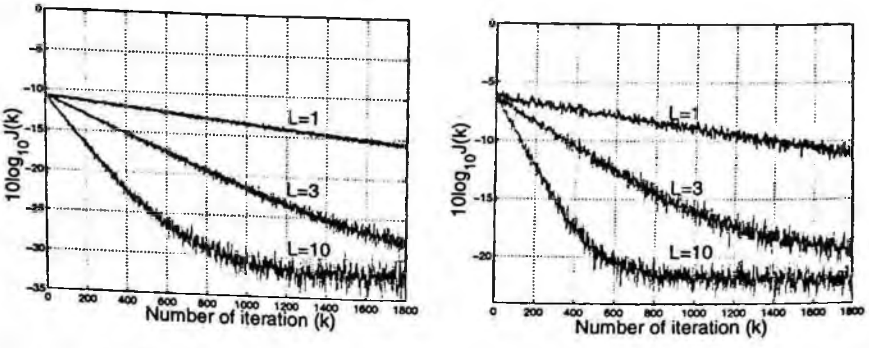
For the experiments on linear FIR adaptive filters, the learning rate is chosen to be $\mu = 0.0001$, and the input signal was a complex $AR(4)$ process given by

$$\begin{aligned} r(k) &= 1.79r(k-1) - 1.85r(k-2) \\ &\quad +1.27r(k-3) - 0.41r(k-4) + n(k) \end{aligned} \tag{26}$$

with complex white Gaussian noise (WGN) $n(k) \sim \mathcal{N}(0,1)$ as the driving input. The complex WGN can be expressed as $n(k) = n^r(k) + jn^i(k)$. The real and imaginary components of the WGN noise are mutually independent sequences having the same variances so that $\sigma_n^2 = \sigma_{n^r}^2 + \sigma_{n^i}^2$. The complex nonlinear input signal was calculated as (Narendra and Parthasarathy 1990),

$$z(k) = \frac{z(k-1)}{1 + z^2(k-1)} + r^3(k). \tag{27}$$

Simulations were performed with 100 iterations of independent trials averaged on the prediction of the complex-valued nonlinear input signals. Figure 3 shows the ensemble performance curves for the CLMS algorithm with $L = 1$, $L = 3$, and $L = 10$. The DRCLMS outperformed the CLMS algorithm ($L = 1$). As the number of data reuse iterations L increases, the speed of convergence improved approaching to the NCLMS algorithm in the limit.



(a) Nonlinear Input

(b) Coloured input

Figure 3. Performance of CLMS and data-reusing CLMS with L=1,3 and 10 for prediction of a nonlinear input (27) and coloured input (26)

3 Complex Nonlinear Adaptive Filters

We now extend the analysis of DR algorithm to common choices of complex nonlinear adaptive filters realised as a feedforward perceptron and recurrent perceptron from the class of recurrent neural networks (RNNs).

3.1 The Choice of Activation Functions In Complex-Valued Neural Networks

It is stated in Liouville's theorem, *a bounded entire function must be a constant in* $\mathbb{C}$ (Kim and Adali 2001). To this cause, meromorphic functions are employed as the nonlinear activation function in complex neural network. This class of activation function is analytic everywhere except in a discrete subset of $\mathbb{C}$ and at their singularities where they must tend to infinity thus removing the possibility of encountering essential singularities. The Cauchy-Riemann equations provide the necessary conditions for a complex function to be analytic at the point $z \in \mathbb{C}$. There are five desirable properties of a fully complex activation function $\Phi(k)$ given as (Kim and Adali 2001):

1. $\Phi(net(k))$ is nonlinear and $net(k) = \sigma(k) + j\tau(k)$.
2. $\Phi(net(k))$ is bounded.
3. Partial derivatives $u_\sigma, u_\tau, v_\sigma, v_\tau$ exist and are bounded, where $u_\sigma = \frac{\partial u}{\partial \sigma}$, $u_\tau = \frac{\partial u}{\partial \tau}$, $v_\sigma = \frac{\partial v}{\partial \sigma}$, $v_\tau = \frac{\partial v}{\partial \tau}$.
4. $\Phi(net(k))$ is not entire.
5. $u_\sigma v_\tau \neq v_\sigma u_\tau$.

To arrive at the Cauchy-Riemann equations, note that the partial derivatives of $\Phi(k)$ along the real and imaginary axes should be $\Phi'(k) = u_\sigma + jv_\sigma = v_\tau - ju_\tau$, where $u_\sigma = v_\tau$, $v_\sigma = -u_\tau$.

3.2 The CNGD Algorithm

As stated in the previous section, complex-valued nonlinear algorithms must employ a suitable activation function that exhibits the five properties given in Section 3.1. To this cause we employ functions that are analytic and bounded almost everywhere in $\mathbb{C}$.

Initially we derive the complex nonlinear gradient descent (CNGD)

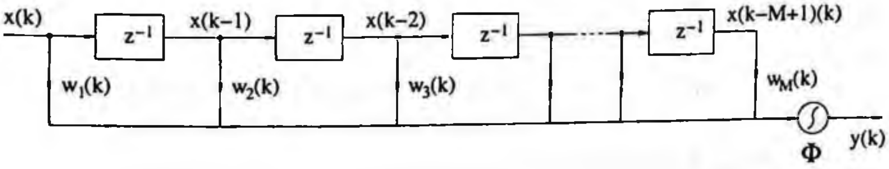


Figure 4. A complex-valued dynamical perceptron (nonlinear FIR filter).

algorithm starting from a complex dynamical perceptron as shown in Figure 4, for which the output is given by $y(k) = \Phi(\mathbf{x}^T(k)\mathbf{w}(k))$. For simplicity we state that,

$$\Phi(\mathbf{x}^T(k)\mathbf{w}(k)) = \Phi(k) = u(k) + jv(k) \tag{28}$$

where $\Phi(k)$ is some complex nonlinear analytic function that is bounded almost everywhere in the complex domain. The vectors $\mathbf{x}(k)$ and $\mathbf{w}(k)$ denote the complex input and complex weight vectors respectively.

By using the chain rule, and defining the following derivatives as $\frac{\partial\sigma}{\partial\mathbf{w}^r} = \mathbf{x}^r$, $\frac{\partial\sigma}{\partial\mathbf{w}^i} = -\mathbf{x}^i$, $\frac{\partial\tau}{\partial\mathbf{w}^r} = \mathbf{x}^i$, $\frac{\partial\tau}{\partial\mathbf{w}^i} = \mathbf{x}^r$, the gradient of the cost function for both the real and imaginary component of the cost gradient can be written as

$$\frac{\partial J(k)}{\partial\mathbf{w}^r(k)} = -e^r(k)\left[u_\sigma\mathbf{x}^r + u_\tau\mathbf{x}^i\right] - e^i(k)\left[v_\sigma\mathbf{x}^r + v_\tau\mathbf{x}^i\right] \tag{29}$$

$$\frac{\partial J(k)}{\partial\mathbf{w}^i(k)} = -e^r(k)\left[-u_\sigma\mathbf{x}^i + u_\tau\mathbf{x}^r\right] - e^i(k)\left[-v_\sigma\mathbf{x}^i + v_\tau\mathbf{x}^r\right] \tag{30}$$

Combining (29) and (30), and employing the Cauchy-Riemann equations, the gradient of the error function can be simplified as

$$\begin{aligned}\nabla_{\mathbf{w}}J(k) &= -\mathbf{x}^*(k)\left[e^r(k)\left(u_\sigma - jv_\sigma\right) + e^i(k)\left(v_\sigma + ju_\sigma\right)\right]\\ &= -\mathbf{x}^*(k)\left[\{\Phi'_\sigma(k)\}^* e^r(k) + j\{\Phi'_\sigma(k)\}^* e^i(k)\right]\\ &= -\mathbf{x}^*(k)\{\Phi'(k)\}^* e(k)\end{aligned} \tag{31}$$

Finally we obtain the weight update to be

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\{\Phi'(k)\}^*\mathbf{x}^*(k). \tag{32}$$

3.3 The DRCNGD Algorithm

The technique of data-reusing relies on the updated weight vector $\mathbf{w}(k+1)$ being available before the next input vector $\mathbf{x}(k+1)$. The equations for the DR complex nonlinear gradient descent (CNGD) algorithm are given by

$$e_t(k) = d(k) - \Phi\left(\mathbf{x}^T(k)\mathbf{w}_t(k)\right) \tag{33}$$

$$\mathbf{w}_{t+1}(k) = \mathbf{w}_t(k) + \mu\left\{\Phi'(\mathbf{x}^T(k)\mathbf{w}_t(k))\right\}^* e_t(k)\mathbf{x}^*(k) \tag{34}$$

It can be clearly seen that

$$\mathbf{w}_1(k) = \mathbf{w}(k), \;\; \mathbf{w}_{L+1}(k) = \mathbf{w}(k+1) \tag{35}$$

where L denotes the number of iterations in the data-reusing (DR) algorithm and if $L = 1$ the DRCNGD algorithm reduces to the standard complex nonlinear gradient descent (CNGD) algorithm. From the analysis for the linear filter it follows that (Hanna and Mandic 2002)

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}_{L+1}(k) \\ &= \mathbf{w}_L(k) + \mu\left\{\Phi'(\mathbf{x}^T(k)\mathbf{w}_L(k))\right\}^* e_L(k)\mathbf{x}^*(k) \\ &= \mathbf{w}(k) + \mu\sum_{t=1}^{L} e_t(k)\left\{\Phi'(\mathbf{x}^T(k)\mathbf{w}_t(k))\right\}^* \mathbf{x}^*(k), \quad (36) \end{aligned}$$

which is the total weight update along the data-reusing iterations. The instantaneous data-reusing output error can be further expressed as (Hanna and Mandic 2002)

$$\begin{aligned} e_t(k) &= d(k) - \Phi\left(\mathbf{x}^T(k)\mathbf{w}_t(k)\right) \\ &= e_{t-1}(k) - \left[\Phi\left(\mathbf{x}^T(k)\mathbf{w}_t(k)\right) - \Phi\left(\mathbf{x}^T(k)\mathbf{w}_{t-1}(k)\right)\right]. \quad (37) \end{aligned}$$

3.4 The Convergence Analysis of Complex-Valued Feedforward Neural Networks

The performance of the data-reusing approach, working as a nonlinear adaptive filter depends also on the characteristics of the complex nonlinear activation function of a neuron, *i.e.* whether it is a contraction or an expansion (Mandic and Chambers 2001). We address the relationships between the *a priori* and *a posteriori* error nonlinear adaptive predictor realised as feedforward, and derive bounds which determine *a posteriori* nonlinear prediction. As illustrated earlier, we used an example of a complex gradient descent based algorithm (CNGD) for a perceptron where the *a posteriori* adaptation is given in equations (33) and (34). Pre-multiplying equation (34) with $\mathbf{x}^T(k)$ and applying the nonlinear activation function Φ on either side, we obtain (Mandic and Chambers 2000)

$$\begin{aligned}\Phi(\mathbf{x}^T(k)\mathbf{w}_{t+1}(k)) &= \Phi[\mathbf{x}^T(k)\mathbf{w}_t(k) + \\ &\qquad \mu e_t(k)\left\{\Phi'[\mathbf{x}^T(k)\mathbf{w}_t(k)]\right\}^* \mathbf{x}^T(k)\mathbf{x}^*(k)] \end{aligned} \tag{38}$$

Further analysis into the convergence of such an algorithm will depend on the characteristic of the complex-valued activation function (see Appendix).

We can now define the lower bound for the *a posteriori* estimation error obtained by algorithm (34), and a contractive nonlinear activation function Φ as

$$e_{L+1}(k) > \left[1 - \mu(k)\left\{\Phi'[\mathbf{x}^T(k)\mathbf{w}(k)]\right\}^* \mathbf{x}^T(k)\mathbf{x}^*(k)\right]^L e_L(k). \tag{39}$$

To show this we apply expression (60) to equation (34), and subtract $d(k)$ from both sides of the resulting equation. Due to the contractivity of Φ, we obtain

$$e_{t+1}(k) > \left[1 - \mu(k)\left\{\Phi'[\mathbf{x}^T(k)\mathbf{w}_t(k)]\right\}^* \mathbf{x}^T(k)\mathbf{x}^*(k)\right] e_t(k) \tag{40}$$

Here, we assume that $\left\{\Phi'[\mathbf{x}^T(k)\mathbf{w}_t(k)]\right\}^*$, $t = 1, 2, \ldots, L$ does not change significantly during successive iterations. Also note that $e_t(k)$ and $e_{t+1}(k)$ have the same sign. By iterating equation (40), we have

$$e_{L+1}(k) > \left[1 - \mu(k)\left\{\Phi'[\mathbf{x}^T(k)\mathbf{w}(k)]\right\}^* \mathbf{x}^T(k)\mathbf{x}^*(k)\right]^L e_L(k) \tag{41}$$

which is the lower bound for the *a posteriori* expectation error for a contractive nonlinear activation function. □

For the algorithm given by the equation (41) to be feasible, the term $\left[1 - \mu(k)\Phi'[\mathbf{x}^T(k)\mathbf{w}(k)]^*\mathbf{x}^T(k)\mathbf{x}^*(k)\right]$ must have the $\mathcal{L}_2$ norm less than unity. In that case the whole procedure is the fixed point iteration . This leads us to the following constraint on the learning rate parameter

$$0 < \mu(k) < \frac{1}{\{\Phi'[\mathbf{x}^T(k)\mathbf{w}(k)]\}^* \, \mathbf{x}^T(k)\mathbf{x}^*(k)} \tag{42}$$

Combining equations (36), (41), and letting

$$\Psi(k) = \mu(k)\left\{\Phi'[\mathbf{x}^T(k)\mathbf{w}(k)]\right\}^* \mathbf{x}^T(k)\mathbf{x}^*(k) \tag{43}$$

yields

$$\mathbf{w}(k+1) < \mathbf{w}(k) + \frac{1 - [1 - \Psi(k)]^L}{\mathbf{x}^T(k)\mathbf{x}^*(k)} e(k)\mathbf{x}^*(k) \tag{44}$$

Thus, for the case of a contractive activation function of a neuron, the upper bound of the weight vector update of the proposed algorithm, after L data-reusing iterations at the time instant k is given by

$$\Delta\mathbf{w}(k) < \frac{1 - [1 - \Psi(k)]^L}{\mathbf{x}^T(k)\mathbf{x}^*(k)} e(k)\mathbf{x}^*(k) \tag{45}$$

If function Φ is an expansion, instead of considering lower bounds, we have upper bounds for *a posteriori* error estimation given as

$$e_{t+1}(k) < \left[1 - \mu(k)\left\{\Phi'[\mathbf{x}^T(k)\mathbf{w}_t(k)]\right\}^* \mathbf{x}^T(k)\mathbf{x}^*(k)\right] e_t(k) \tag{46}$$

and the lower bound of the weight vector update is given as

$$\Delta\mathbf{w}(k) > \frac{1 - [1 - \Psi(k)]^L}{\mathbf{x}^T(k)\mathbf{x}^*(k)} e(k)\mathbf{x}^*(k) \tag{47}$$

This why in practice we try to avoid an expansive activation function of an output neuron.

For the experiments for nonlinear adaptive filters, the nonlinearity at the neuron was chosen to be the logistic sigmoid function,

$$\Phi(\beta, z) = \frac{1}{1+e^{-\beta z}} \tag{48}$$

where $z \in \mathbb{C}$. For a contractive activation function, the slope was chosen as $\beta = 1$. For the case of expansive activation function, the slope was chosen as $\beta = 4$ for the experiment on coloured input (26) and $\beta = 8$ for the experiment on nonlinear input (27). The learning rate was chosen to be a small value of $\mu = 0.001$ in order to have a clear visualization of the performance of the algorithms. Figure

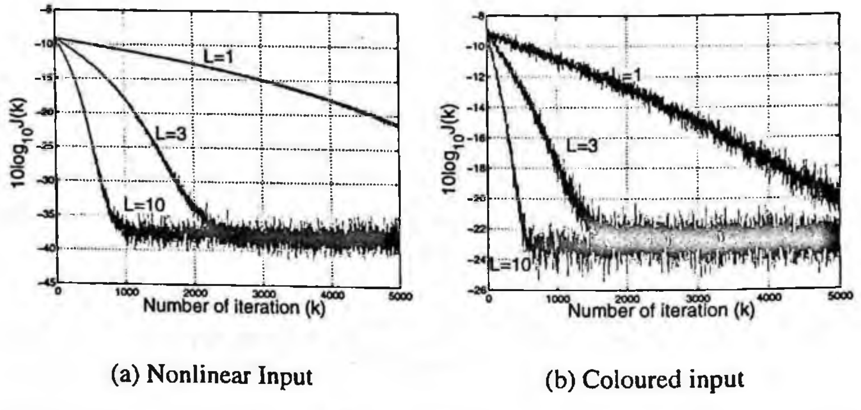


Figure 5. Performance of CNGD and data-reusing CNGD with L=1,3 and 10 for prediction of a nonlinear input (27) and coloured input (26)

5 shows the ensemble performance curves of the DRCNGD algorithm for a perceptron with a contractive activation function for both coloured (26) and nonlinear (27) signals. It is shown that the data-reusing algorithm showed faster convergence than the standard algorithm ($L = 1$) for both types of input signals. The performance of

this algorithm improves with increasing the order of the data-reusing iteration and saturates for large L as proven in the experiment.

3.5 The Complex RTRL Algorithm

For simplicity we start the analysis from the case of a simple recurrent perceptron. The output y of a recurrent perceptron as shown in Figure 6 is given by $y(k) = \Phi(net(k))$ where $\Phi(net(k))$ is some complex nonlinear analytic function that is bounded almost everywhere in the complex domain, and

$$\begin{aligned} net(k) &= \sum_{m=1}^{M} w_m(k)x(k-m) + w_{M+1}(k)(1+j) \\ &\quad + \sum_{n=1}^{N} w_{n+M+1}(k)y(k-n) \end{aligned} \tag{49}$$

where $\mathbf{x}(k) \triangleq [x(k-1), \ldots, x(k-M)]^T$ and $\mathbf{y}(k) \triangleq [y(k-1), \ldots, y(k-N)]^T$ denote the complex input and feedback signals respectively, and the complex weight vector is given by $\mathbf{w}(k) \triangleq [w_1(k), \ldots, w_{M+N+1}(k)]$.

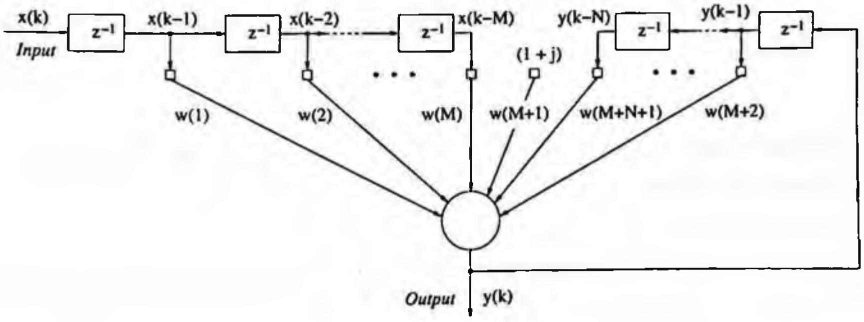


Figure 6. Recurrent perceptron

The complex input signal to the recurrent perceptron consists of the delayed input, delayed output and bias. The complex input vector to the network is therefore given as

$$\mathbf{I}(k) \triangleq [x(k-1), \dots, x(k-M), 1+j, y(k-1), \dots, y(k-N)]^T$$

The weight adaptation in the nonlinear complex gradient descent algorithm is given by equations (3). We want to accumulate the weight changes $\Delta \mathbf{w}(k) = -\mu \nabla_{\mathbf{w}} J(k)$ at each step along the trajectory. Since $J(k)$ is real valued, its gradient can be computed in a similar way to the previous sections. The factor $\frac{\partial y(k)}{\partial \mathbf{w}(k)}$ is a measure of sensitivity of the value of the output at time k to a small increase in the value of $\mathbf{w}(k)$ taking into account the effect of such a change in the weight over the entire trajectory. For simplicity, we represent the sensitivities with the symbols given as $\pi^{rr}(k) = \frac{\partial u(k)}{\partial \mathbf{w}^r(k)}$, $\pi^{ir}(k) = \frac{\partial v(k)}{\partial \mathbf{w}^r(k)}$, $\pi^{ri}(k) = \frac{\partial u(k)}{\partial \mathbf{w}^i(k)}$ and $\pi^{ii}(k) = \frac{\partial v(k)}{\partial \mathbf{w}^i(k)}$. The sensitivity π is a complex-valued vector which can be shown as $\pi(k) = \pi^{rr}(k) + j\pi^{ir}(k) = \pi^{ii}(k) - j\pi^{ri}(k)$. From Cauchy-Riemann conditions, we can write the condition that, $\pi^{rr}(k) = \pi^{ii}(k)$, $\pi^{ri}(k) = -\pi^{ir}(k)$. By employing the Cauchy-Riemann condition, we finally obtain the weight update as

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) - \mu \nabla_{\mathbf{w}} J(k) \\ &= \mathbf{w}(k) + \mu e(k) \left(\pi^{rr}(k) - j\pi^{ir}(k) \right) \\ &= \mathbf{w}(k) + \mu e(k) \pi^*(k) \end{aligned} \tag{50}$$

with the initial conditions $\pi(0) = 0$. The update for the sensitivities is then derived as

$$\pi^*(k) = (\Phi'(net)(k))^* \left[\mathbf{I}^*(k) + \sum_{m=1}^{N} w^*_{m+M+1}(k) \pi^*(k-m) \right]. \tag{51}$$

3.6 The DRCRTRL Algorithm

At each iteration, the adaptive weight vector is being updated by reusing the current tap input vector. The data-reusing weight update algorithm for a complex-valued recurrent nonlinear filter can be written as (Mandic and Chambers 2001)

$$\mathbf{w}_{t+1}(k) = \mathbf{w}_t(k) + \mu e_t(k)\boldsymbol{\pi}_t^*(k) \tag{52}$$

$$e_t(k) = d(k) - \Phi(\mathbf{I}^T(k)\mathbf{w}_t(k)), \quad t = 1, \ldots, L \tag{53}$$

where the cost function $J_t(k)$ is given by (4). The index t denotes the t–th iteration of the equations (52) and (53) and μ is the learning rate. Note that the weight update recursion starts with $\mathbf{w}_1(k) = \mathbf{w}(k)$ and ends with $\mathbf{w}_{L+1}(k) = \mathbf{w}(k+1)$.

Following the approach from (Mandic 2002), starting from the last iteration in equation (52), for $t = L$, we obtain the final data-reusing weight update at $t = L + 1$ as

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}_{L+1}(k) = \mathbf{w}_L(k) + \mu e_L(k)\boldsymbol{\pi}_L^*(k) \\ &= \mathbf{w}_{L-1}(k) + \mu e_{L-1}(k)\boldsymbol{\pi}_{L-1}^*(k) + \mu e_L(k)\boldsymbol{\pi}_L^*(k) \\ &= \mathbf{w}(k) + \sum_{t=1}^{L} \mu e_t(k)\boldsymbol{\pi}_t^*(k) \end{aligned} \tag{54}$$

The instantaneous error at the output neuron can be further expressed as

$$\begin{aligned} e_t(k) &= d(k) - \Phi(\mathbf{I}^T(k)\mathbf{w}_t(k)) \\ &= \left[d(k) - \Phi(\mathbf{I}^T(k)\mathbf{w}_{t-1}(k))\right] \\ &\quad - \left[\Phi(\mathbf{I}^T(k)\mathbf{w}_t(k)) - \Phi(\mathbf{I}^T(k)\mathbf{w}_{t-1}(k))\right] \\ &= e_{t-1}(k) - \left[\Phi(\mathbf{I}^T(k)\mathbf{w}_t(k)) - \Phi(\mathbf{I}^T(k)\mathbf{w}_{t-1}(k))\right] \end{aligned} \tag{55}$$

From the equation above, we can see that the t–th data–reusing error depends on the features of a complex nonlinear activation function of the neuron, Φ. Pre-multiplying equation (52) by $\mathbf{I}^T(k)$, and then applying the nonlinear activation function Φ on either side, we obtain

$$\Phi(\mathbf{I}^T(k)\mathbf{w}_{t+1}(k)) = \Phi\left(\mathbf{I}^T(k)\mathbf{w}_t(k) + \mu\mathbf{I}^T(k)e_t(k)\boldsymbol{\pi}_t^*(k)\right) \tag{56}$$

The characteristics of Φ, that is, whether it is a contraction or expansion will effect the convergence of the data–reusing algorithm.

3.7 The Convergence Analysis of a Complex-Valued Recurrent Neural Filter

The bounds on the data-reusing error for the complex-valued RTRL algorithm conforms to the analysis shown in section 3.4. It is assumed that the value of the learning rate $\mu(k)$ is held fixed during the fixed-point iteration of the data-reusing algorithm. After applying the data-reusing algorithm L times, the cummulative output error becomes

$$\begin{aligned}\sum_{t=1}^{L} e_t(k) &> \sum_{t=1}^{L}\left[1-\mu(k)\mathbf{I}^T(k)\pi^*(k)\right]^{t-1} e(k)\\ &= \frac{1-\left[1-\mu(k)\mathbf{I}^T(k)\pi^*(k)\right]^L}{\mu(k)\mathbf{I}^T(k)\left[\pi^*(k)\right]} e(k) \qquad (57)\end{aligned}$$

where every $e_t(k)$ decreases along the data-reusing iteration due to the contractivity of the term in the square brackets in the nominator of (57). We obtain the amount for which the weights change after L iterations as

$$\Delta\mathbf{w}(k) < \frac{1-\left[1-\mu(k)\mathbf{I}^T(k)\pi^*(k)\right]^L}{\mathbf{I}^T(k)\pi^*(k)} e(k)\pi^*(k) \qquad (58)$$

Figure 7 shows the ensemble performance curves of the DRCRTRL algorithm for a perceptron with a contractive activation function for both nonlinear (27) and coloured input (26) signals. It is shown that the data-reusing algorithm showed faster convergence than the standard algorithm ($L = 1$) for both types of input signals. The performance of this algorithm improves with increasing the order of the data-reusing iteration and saturates for large L as proven in the experiment. Figure 8 shows the performance of a data–reusing CRTRL algorithm for a recurrent perceptron with an expansive activation function. As shown in the figures, the error curve does not converge and grows without bound. The divergence is more emphasized with the order of data-reusing iteration.

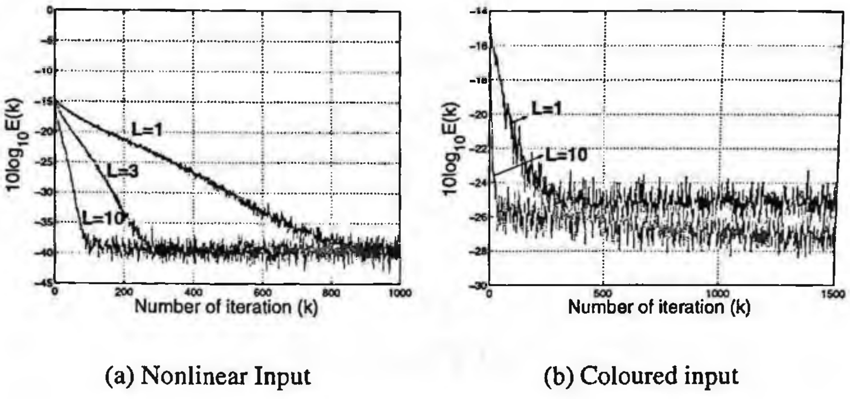


Figure 7. Performance of CRTRL and data-reusing CRTRL for prediction of a nonlinear input (27) and coloured input (26) for a contractive nonlinear activation function

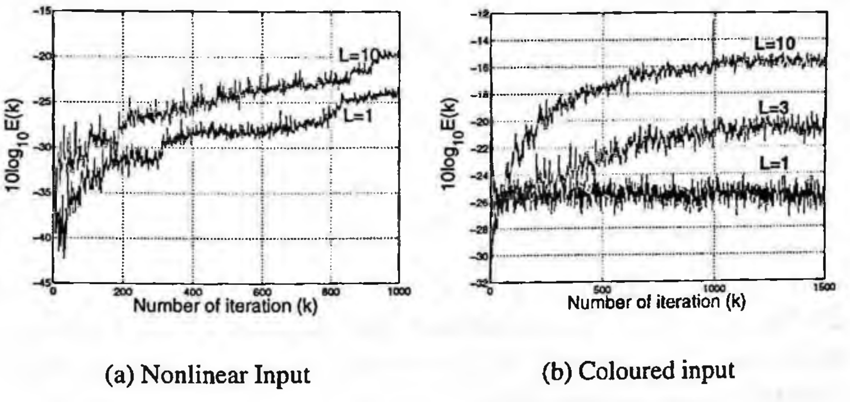


Figure 8. Performance of CRTRL and data-reusing CRTRL for prediction of a nonlinear input (27) and coloured input (26) for an expansive nonlinear activation function

4 Conclusions

The class of data-reusing algorithms for complex-valued linear and nonlinear adaptive filters has been presented and the relationships for the prediction error and learning rate have been provided. A lower bound on the data-reusing output error has been established and the error is proven to have uniformly smaller magnitude along the data-reusing iteration. The data-reusing algorithm offers a significantly improved convergence rate over the standard algorithm for a nonlinear activation function exhibiting contractive behavior.

5 Appendix — The Characteristics of the Activation Function

By the Contraction Mapping Theorem (CMT), function K is a contraction on $[a,b] \in \mathbb{R}$ if (Gill *et al.* 1981):

i) $x \in [a,b] \Rightarrow K(x) \in [a,b]$

ii) $\exists \gamma < 1 \in \mathbb{R}^+ \ \ s.t. \ \ |K(x) - K(y)| \leq \gamma |x-y|, \quad \forall x, y \in [a,b]$

Using the Mean Value Theorem (MVT), for $\forall x, y \in [a,b]$, $\exists \xi \in (a,b)$ such that

$$|K(x) - K(y)| = |K'(\xi)(x-y)| = |K'(\xi)||x-y| \qquad (59)$$

Now, the clause $\gamma < 1$ in **ii)** becomes $\gamma \geq |K'(\xi)|$, $\xi \in (a,b)$. For the example of the logistic nonlinear activation function of a neuron $\Phi(v) = \frac{1}{1+e^{-\beta v}}$, with slope $\beta, \gamma < 1 \ \Leftrightarrow \ \beta < 4$ is the condition for function Φ to be a contraction.

For a meromorphic complex contractive activation function Φ, we can write the contractivity condition as

$$|\Phi(a+b)| \leq |\Phi(a) + \Phi(b)| \qquad (60)$$

For the case of meromorphic complex expansive activation function Φ, we can write the expansive condition as

$$|\Phi(a+b)| \geq |\Phi(a) + \Phi(b)| \qquad (61)$$

References

Benvenuto, N. and Piazza, F. (1992), "On The Complex Backpropagation Algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967-969.

Connor, J.T., Martin, R.D. and Atlas, L.E (1994), "Recurrent neural networks and robust time series prediction," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 240-254.

Douglas, S.C. and Rupp, M. (1997), "A Posteriori Updates For Adaptive Filters," *Signals, Systems and Computers, 1997. Conference Record of the Thirty-First Asilomar Conference*, vol. 2, pp. 1641 -1645.

Georgiou, G. M. and Koutsougeras, C. (1992), "Complex Domain Backpropagation," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Processing*, vol. 35, no. 5, pp. 330-334.

Gill, P. E., Murray, W. and Wright, M. H. (1981), "Practical Optimization," *Academic Press, London.*

Hanna, A.I. and Mandic, D.P. (2002), "A Data-Reusing Nonlinear Gradient Descent Algorithm for a Class of Complex-Valued Neural Adaptive Filters," *Neural Processing Letters*, vol. 17, pp. 1-7.

Hirose, A. (1990), "Continuous Complex-Valued Backpropagation Learning," *Electronics Letters*, vol. 28, no. 20, pp. 1854-1855.

Kechriotis, G. and Manolakos, E. S. (1994), "Training Fully Recurrent Neural Networks with Complex Weights," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Processing*, vol. 41, no. 3, pp. 235-238.

Kim, T. and Adali, T. (2001), "Complex Backpropagation Neural Network Using Elementary Transcendental Activation Functions," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, vol. 6, pp. 1281-1284

Kim, T. and Adali, T. (2001), "Fully Complex Backpropagation for Constant Envelope Signal Processing," *Proceedings of the 2000 IEEE Neural Networks for Signal Processing X, 2000 Society Workshop*, vol. 1, pp. 231 -240.

Mandic, D.P. and Chambers, J.A. (2000), "Relations between the a priori and a posteriori errors in nonlinear adaptive neural filter," *Neural Computation*, vol. 12, no. 6, pp. 1285-1292.

Mandic, D.P. and Chambers, J.A. (2000), "A posteriori real time recurrent learning schemes for a recurrent neural network based non-linear predictor," *IEE Proceedings–Vision, Image and Signal Processing*, vol. 145, no. 6, pp. 365-370.

Mandic, D. P. and Chambers, J. A. (2001), "Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability," *John Wiley & Sons.*

Mandic, D.P. (2002), "Data-reusing recurrent neural adaptive filters," *Neural Computation*, vol. 14, no. 11, pp. 2693-2707 .

Narendra, K. S. and Parthasarathy, K. (1990), "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4-27.

Nerrand, O., Roussel-Ragot, P., Personnaz, L. and Dreyfus, G. (1993), "Neural networks and nonlinear adaptive filtering: Unifying concepts and new algorithms," *Neural Computation*, vol. 5, pp. 165-199.

Roy, S. and Shynk, J.J. (1989), "Analysis of the data–reusing LMS algorithm," *Proceedings of the 32nd Midwest Symposium on Circuits and Systems*, vol. 2, pp. 1127-1130.

Schnaufer, B.A. and Jenkins, W.K. (1993), "New data–reusing LMS algorithms for improved convergence," *Conference Record of the Twenty-Seventh Asilomar Conference on Signals and Systems*, vol. 2, pp. 1584-1588.

Treichler, J. R., Johnson, Jr. C. R. and Larimore, M.G (1987), "Theory and design of adaptive filters," *Wiley-Interscience*.

Widrow, B., McCool, J. and Ball, M. (1975), "The Complex LMS Algorithm," *Proceedings of the IEEE*, vol. 63, pp. 710-720.

Authors' address

Danilo P. Mandic and **Su Lee Goh**: Communications & Signal Processing Group, Department of Electrical and Electronic, Imperial College of Science, Technology and Medicine, London SW7 2AZ, United Kingdom
Andrew I. Hanna: Signal & Imaging Informatics Group, Royal Society Wolfson Bioinformatics Lab, The University of East Anglia, Norwich, NR4 7TJ, United Kingdom

Chapter 8

Instantaneously Trained Neural Networks with Complex Inputs

Pritam Rajagopal and Subhash Kak

Neural network architectures that can handle complex inputs, such as backpropagation networks, perceptrons or generalized Hopfield networks, require a large amount of time and resources for the training process. Here we adapt the time-efficient corner classification approach to train feedforward neural networks to handle complex inputs and present a new algorithm called the 3C algorithm. This algorithm uses prescriptive learning, where the network weights are assigned simply upon examining the inputs. The performance of the algorithm is tested using the pattern classification experiment and the time series prediction experiment with the Mackey-Glass time series. An input encoding called quaternary encoding is used for both experiments since it reduces the network size significantly by cutting down on the number of neurons that are required at the input layer.

1 Introduction

Prior complex neural network models (Kim and Guest 1990, Noest 1988, Sutherland 1990) have generalized the Hopfield model, backpropagation and the perceptron learning rule to handle complex inputs. Noest (1988) formulated the Hopfield model for inputs and outputs falling on the unit circle in the complex plane. Georgiou (1992) described the complex perceptron learning rule. Also, Georgiou and others (Benvenuto and Piazza 1992, Leung and

Haykin 1991, Little *et al.* 1990) described the complex domain backpropagation algorithm. More recently, work presented by Li, Liao and Yu (Li *et al.* 2002) uses digital filter theory to perform the fast training of complex-valued recurrent neural networks.

The training processes used by the different architectures mentioned above are iterative, requiring a large amount of computer resources and time for the training. This may not be desirable in some applications. The corner classification approach (Kak 1993, 1994, 1998 and Tang 1997) (algorithms CC1 to CC4), speeds up the training process of neural networks that handle binary inputs, achieving instantaneous training. A generalization of this approach for mapping non-binary inputs to non-binary outputs is presented by Kak and Tang (Kak 2002, Tang and Kak 2002).

The corner classification approach utilizes prescriptive learning. In this procedure, the network interconnection weights are assigned based entirely on the inputs without any computation. The corner classification algorithms such as CC3 and CC4 are based on two main ideas that enable the learning and generalization of inputs:

1. The training vectors are mapped to the corners of a multidimensional cube. Each corner is isolated and associated with a neuron in the hidden layer of the network. The outputs of these hidden neurons are combined to produce the target output.
2. Generalization using the *radius of generalization* enables the classification of any input vector within a Hamming Distance from a stored vector as belonging to the same class as the stored vector.

Due to its generalization property, the CC4 algorithm can be used efficiently for certain AI problems. The results of pattern recognition and time series prediction experiments using CC4 are presented by Tang (1997). When sample points from a pattern are presented to the network, the CC4 algorithm trains it to store these samples. The network then classifies the other input points based on the radius of generalization, allowing for the network to recognize the pattern with good accuracy. In time-series prediction, some samples from the series are used for training, and then the network can predict future values in the series.

Here the corner classification approach is generalized to handle complex inputs and a modification of the CC4 algorithm is presented, which uses a new procedure of weight assignment. The next section describes a new encoding scheme called the quaternary encoding, which will be used in different experiments to analyze the new algorithm. Section 3 presents the 3C algorithm, and in section 4 the performance of the algorithm is tested using the time series prediction experiment. Finally, the last section provides the conclusions related to the use of complex binary inputs in corner classification and the future of the 3C algorithm.

2 Quaternary Encoding

The quaternary encoding scheme is a simple modification of the unary scheme and accommodates two additional characters i and $1+i$ besides 0 and 1. Due to the additional characters in this scheme, the length of the *codewords* for a range of integers is reduced when compared to unary. For example the integers 1 to 16 is represented using quaternary codewords only five characters, whereas the unary required 16-bit strings for the same range of numbers. Table 1 shows the set of codewords used to represent the integers 1 to 16.

2.1 Length of the Codewords

An important issue is to decide the length of the codewords required to represent a desired range of integers. Let l be the length of the codewords for a range of C integers. Consider the integers in Table 1. For this range $C = 16$ and $l = 5$. We can now examine how 16 codewords can be formed with $l = 5$. The 16 codewords can be classified into three groups. The first group represents integers 1 to 6, where the codewords are constructed without using characters *i* or *1+i*. The codewords in the second group represent integers 7 to 11 and don't use *1+i*, while in the third group the codewords representing integers 12 to 16 use *1+i*.

Table 1. Quaternary codewords for integers 1 to 16

Integer	Quaternary code				
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	1
4	0	0	1	1	1
5	0	1	1	1	1
6	1	1	1	1	1
7	1	1	1	1	i
8	1	1	1	i	i
9	1	1	I	i	i
10	1	i	I	i	i
11	i	i	I	i	i
12	i	i	I	i	1+i
13	i	i	i	1+i	1+i
14	i	i	1+i	1+i	1+i
15	i	1+i	1+i	1+i	1+i
16	1+i	1+i	1+i	1+i	1+i

We see here that the first group has 6 codewords. The other two have 5 each, corresponding to the length of the codewords as the next new character fills up one position after another in each successive codeword. For any *C*, the set of codewords would consist of three such groups where the first group has *l* + 1 codewords, and the second and third have *l* codewords each. This can be summarized as follows:

$$C = (l + 1) + l + l \quad (1)$$
$$C = 3 * l + 1 \quad (2)$$
$$l = (C - 1) / 3 \quad (3)$$

Equation 3 is valid only when $(C - 1)$ is divisible by 3. For cases when this is not true, we obtain:

$$l = \textbf{ceil}\ [(C - 1) / 3] \quad (4)$$

When $(C - 1)$ is not divisible by 3, the number of codewords that can be formed using the l obtained from Equation 4 is more than required. In this case any C consecutive codewords from the complete set of words of length l may be used.

3 The 3C Algorithm

The 3C algorithm (from Complex Corner Classification, CCC) is a generalization of the CC4 and is capable of training 3-layered feedforward networks to map inputs from the alphabet {0, 1, i, 1+i} to the real binary outputs 0 and 1. This algorithm uses a different procedure for the assignment of the input interconnection weights when compared to the CC4 algorithm. Therefore the combination procedure of these weights with the inputs is also different. The features of the algorithm and its network are:

1. The number of input neurons is one more than the number of input elements in a training sample. The extra neuron is the bias neuron which is always set to one.

2. A hidden neuron is created for each training sample; the first hidden neuron corresponds to the first training sample, the second neuron corresponds to the second sample and so on.

3. The output layer is fully connected; each hidden neuron is connected to all the output neurons.

4. The interconnection weights from all the input neurons excluding the bias neuron are complex. Each input of the alphabet {0, 1, i, 1+i} is treated as complex for the weight assignment.

5. If the real part of the input element is 0 then the real part of the corresponding input interconnection weight is assigned as -1. If the real part is of the input element is 1 then the real part of the weight is also set as 1.

6. Similarly if the complex part of the input is 0 then the complex part of the weight is assigned as -1 or if the complex part of the input is 1 then the weight is also assigned as 1.

7. The weight from the bias neuron to a hidden neuron is assigned as $r - s + 1$, where r is the radius of generalization. The value of s is assigned as sum of the number of ones, i's, and twice the number of $(1+i)$s in the training vector corresponding to the hidden neuron.

8. If the desired output is 0 the output layer weight is set as an inhibitory -1. If the output is 1, then the weight is set as 1.

9. The altered combination procedure of the inputs and the weights causes the hidden neuron inputs to be entirely real. Thus the activation function required at the hidden layer is simply the binary step activation function. The output layer also uses a binary step activation function.

When an input vector is presented to the network, the real and imaginary parts of each input element of the vector are multiplied to the corresponding interconnection weight's real and imaginary parts respectively. The two products are then added to obtain the individual contribution by an input element in the vector. This is done for each element and their individual contributions are then added together to obtain the total contribution of the entire vector. Using this combination procedure, each hidden neuron always receives only real inputs. Thus only a binary step activation function is required at the hidden layer. As an example consider the vector (1 1+i i). The corresponding weight vector is (1-i 1+i -1+i). The input vector now combines with this weight vector to yield a total contribution of 4. This contribution is computed as follows:

$$\begin{aligned}&(\mathrm{Re}\,(1)*\mathrm{Re}\,(1-i) + \mathrm{Im}\,(1)*\mathrm{Im}\,(1-i))\\ &\quad + (\mathrm{Re}\,(1+i)*\mathrm{Re}\,(1+i) + \mathrm{Im}\,(1+i)*\mathrm{Im}\,(1+i))\\ &\qquad + (\mathrm{Re}\,(i)*\mathrm{Re}\,(-1+i) + \mathrm{Im}\,(i)*\mathrm{Im}\,(-1+i)) = 4\end{aligned}$$

The 3C algorithm can be expressed as a set of simple **if-then** rules. The formal algorithm is as follows: -

```
for each training vector $x_m$ [n] do
    $s_m$ = no of 1 s + no of i's + 2*(no of (1+i) s) in $x_m$[1:n-1];
    for index = 1 to n-1 do                          // $w_m$ [ ]: input weights
            if Re($x_m$ [index]) = 0 then
                Re($w_m$ [index]) = -1;
            end if
            if Re($x_m$ [index]) = 1 then
                Re($w_m$ [index]) = 1;
            end if
            if Im($x_m$ [index]) = 0 then
                Im($w_m$ [index]) = -1;
            end if
            if Im($x_m$ [index]) = 1 then
                Im($w_m$ [index]) = 1;
            end if
    end for
    $w_m$ [n] = r - $s_m$ + 1;
    for index1 = 1 to k do                           // k = no of outputs y
            if $y_m$ [index1] = 0 then
                $ow_m$ [index1]  = -1;               // $ow_m$ [ ]: output wts
            else
                $ow_m$ [index1]  = 1;
            end if
        end for
    end for
```

Let r = 0, now when an input vector is presented to the network each input neuron receives each element in the vector as input. These inputs combine with their respective weights and all input neurons together, except the bias neuron, provide the hidden neuron corresponding to the input vector with a contribution equal to the s value of the vector. And since r is set as zero, the contribution from the bias neuron is equal to $-s + 1$. Thus the total input to the hidden neuron is 1. All other hidden neurons receive zero or negative input. This ensures that only one hidden neuron fires for each input.

The following examples illustrate the working of the algorithm. The first example is similar to the XOR function but 0 is replaced by i and 1 is replaced by $1+i$. The next example shows how the algorithm works with two output neurons.

Example 1

The inputs and outputs are shown below in Table 2. The 3C algorithm can be used to train a network to map these inputs to the outputs. The network architecture is shown in Figure 1 and the various network parameters are tabulated in Table 3.

Table 2. Inputs and outputs for Example 1.

Inputs		Output
X_1	X_2	Y
i	i	0
i	1+i	1
1+i	i	1
1+i	1+i	0

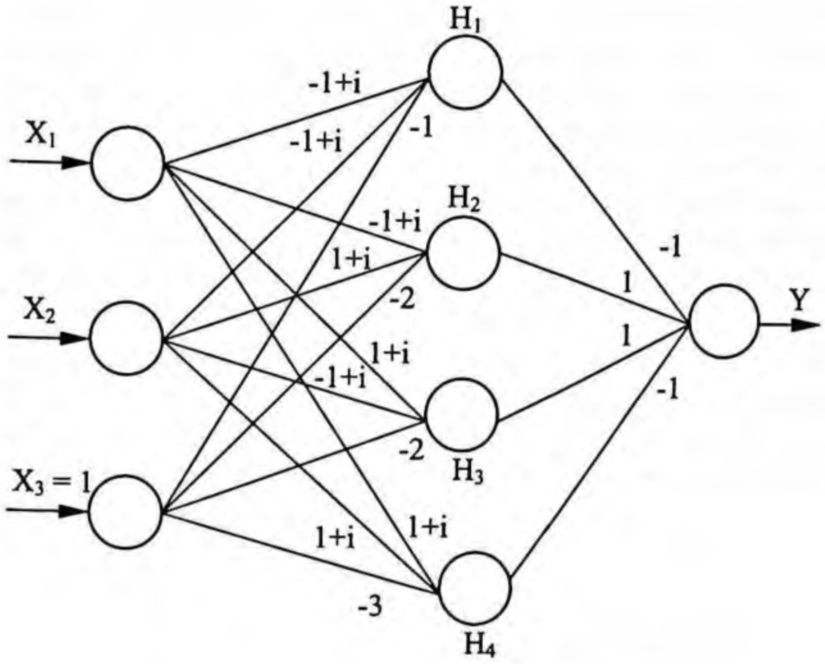


Figure 1. The Network Architecture for Example 1.

Table 3. Network Parameters in the input/output mapping of Example 1.

							Input to				Output of				Output
Inputs			s	Weights			H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4	of y
i	i	1	2	-1+i	-1+i	-1	1	0	0	-1	1	0	0	0	0
i	1+i	1	3	-1+i	1+i	-2	0	1	-1	0	0	1	0	0	1
1+i	i	1	3	1+i	-1+i	-2	0	-1	1	0	0	0	1	0	1
1+i	1+i	1	4	1+i	1+i	-3	-1	0	0	1	0	0	0	1	0

Each input vector has two elements and so three input neurons are required including the one for the bias neuron. All four samples are required for the training. Thus four hidden neurons are used. The inputs need to be mapped to just one output in each case and so only one output neuron is used here. Also since no generalization is required we have $r = 0$. The weights are assigned according to the algorithm and the network is then tested with all inputs. It is seen that all inputs have been successfully mapped to their outputs.

Example 2: Network with two output neurons

The 3C algorithm can also be used to map inputs to a network with more than one output neuron. The inputs and outputs are shown in Table 4. The input vectors have five elements and the corresponding output vectors have two elements.

Table 4. Inputs and outputs for Example 2

Inputs					Outputs	
X_1	X_2	X_3	X_4	X_5	Y_1	Y_2
0	1+i	1+i	0	i	1	1
1+i	0	1	1+i	1	0	1
1	1	i	0	1	1	0

A total of six input neurons are required including the bias neuron. All three samples need to be used for training and hence three hidden neurons are required. The output layer consists of two neurons. The input and output weights are assigned according to the algorithm as each training sample is presented. No generalization is required so $r = 0$. After the training, the network is tested for all inputs and outputs. Again it can be seen that the mapping is accomplished successfully. The network architecture is shown in Figure 2 and the various network parameters obtained during the training are tabulated in Table 5.

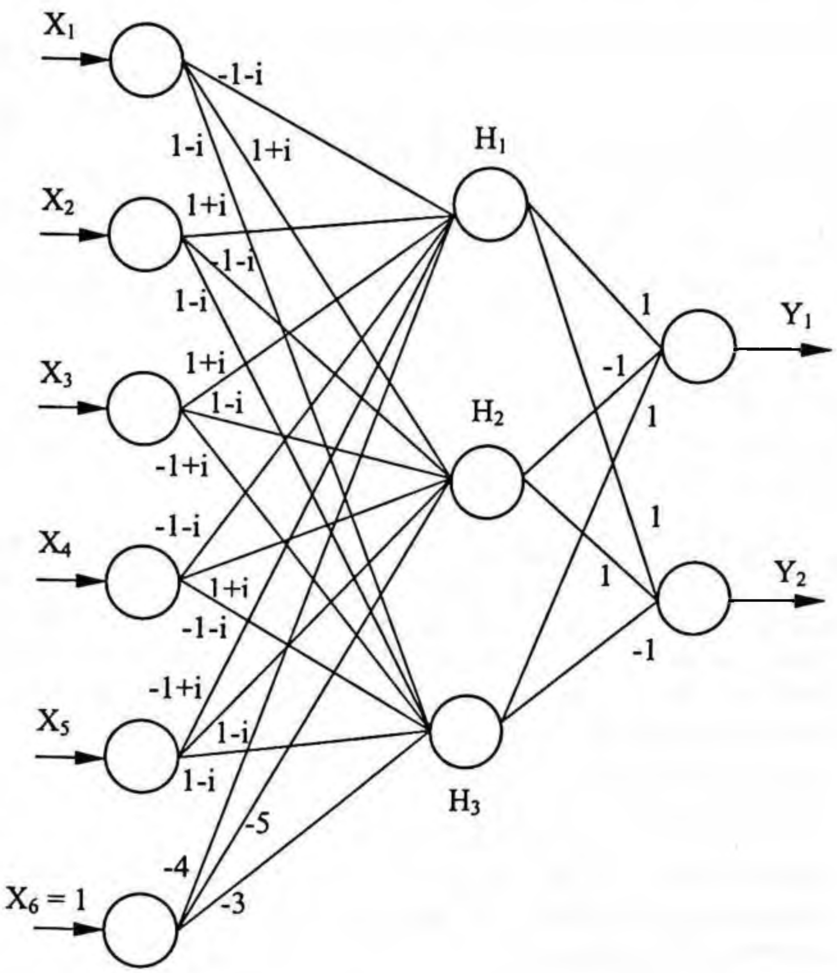


Figure 2. The Network Architecture for Example 2.

Table 5. Network Parameters in the input/output mapping of Example 2.

Inputs						*s*	Weights					
0	1+i	1+i	0	i	1	5	-1-i	1+i	1+i	-1-i	-1+i	-4
1+i	0	1	1+i	1	1	6	1+i	-1-i	1-i	1+i	1-i	-5
1	1	i	0	1	1	4	1-i	1-i	-1+i	-1-i	1-i	-3

Input to			Output of			Output	
H_1	H_2	H_3	H_1	H_2	H_3	y1	y2
1	-8	-4	1	0	0	1	1
-8	1	-5	0	1	0	0	1
-4	-5	1	0	0	1	1	0

These above examples show how well the network can be trained to store vectors and then associate the vectors with their appropriate outputs when the vectors are presented to the network again. However the generalization property cannot be observed since in both examples *r* is set to 0. This property of the 3C algorithm can be analyzed by a pattern classification experiment. The algorithm is used to train a network to separate two regions of a spiral pattern. The original pattern is shown in Figure 3 (a). The 16 by 16 area is divided into a black spiral shaped region and another white region. A point in the black spiral region is represented as a binary "1" and a point in the white region is represented by a binary "0". Any point in the region is represented by row and column coordinates. These coordinates, simply row and column numbers, are encoded using 5-character quaternary encoding. These two codes are concatenated and then a bit is added for the bias. This 11-character vector is fed as input to the network. The corresponding outputs are 1 or 0, to denote the region that the point belongs to.

The training samples are randomly selected points from the two regions of the pattern. The samples used here are shown in Figure 3 (b). The points marked "#" are the points from the black region and the points marked "o" are points from the white region. A total of 75 points are used for training. Thus the network used for this pattern classification experiment has 11 neurons in the input layer and 75 neurons in the hidden layer. The output layer requires only one neuron to display a binary "0" or "1".

After the training is done the network is tested for all 256 points in the 16 by 16 area of the pattern. The experiment is repeated then by changing the value of *r* from 1 to 4. The results for the different levels of generalization achieved are presented in Figure 3 (c), (d), (e) and (f). It can be seen that as the value of *r* is increased the network tends to generalize more points as belonging to the black region. This over generalization is because during training, the density of the samples presented from the black region was greater than the density of samples from the white region. A summary of the experiment is presented in Table 6. This table contains the number of points classified and misclassified during the testing.

Table 6. No. of points classified/misclassified in the spiral pattern

		No. of points			
		$r=1$	$r=2$	$r=3$	$r=4$
Spiral Pattern	Classified	230	232	233	220
	Misclassified	26	24	23	36

4 Time Series Prediction

The Mackey-Glass time series, originally developed to model white blood cell production as presented by Azoff (1994), is commonly used to test the performance of neural networks. The series is a chaotic time series making it an ideal representation of the nonlinear oscillations of many physiological processes. The discrete time representation of the series was used by Tang (1997) to test the performance of the CC4 algorithm. The same will be used here to test the performances of the 3C algorithm.

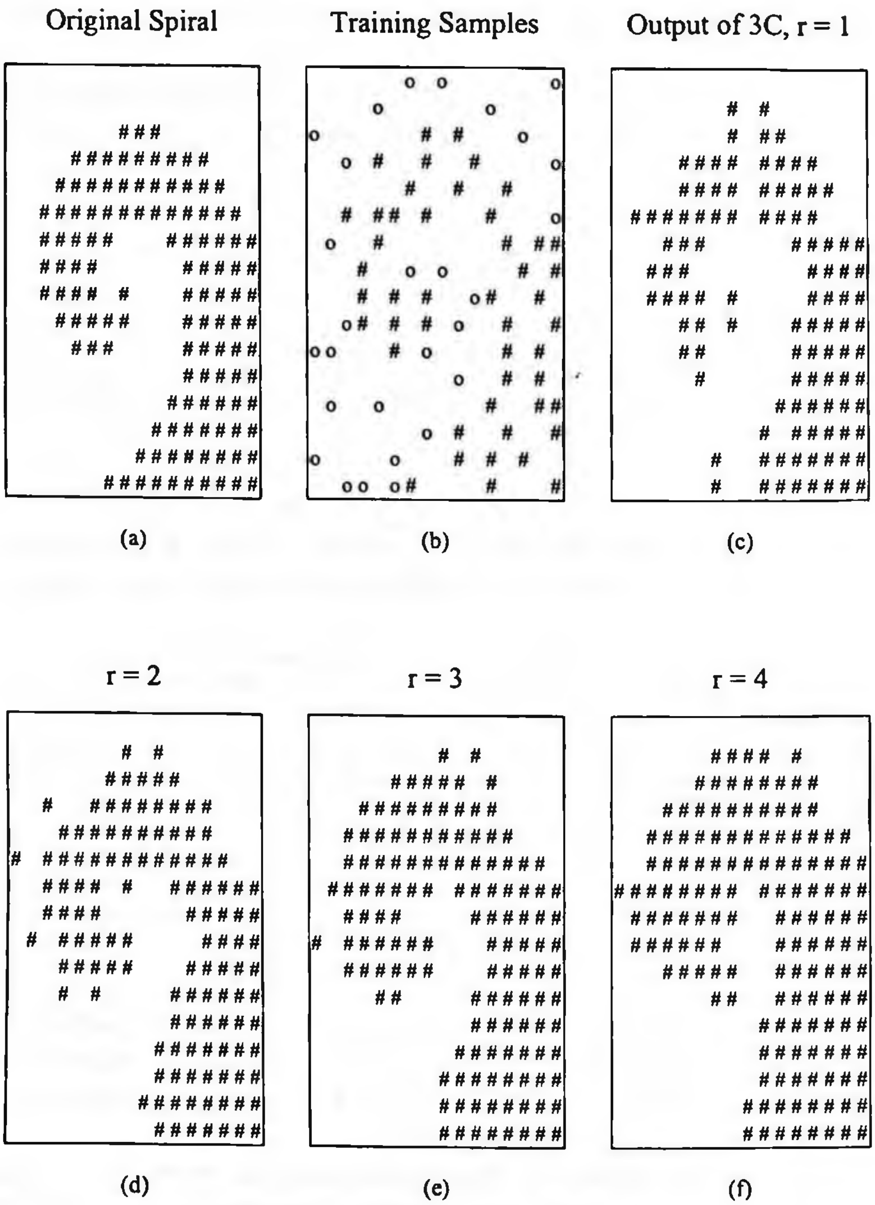


Figure 3. Results of spiral pattern classification.

The discrete time representation of the Mackey-Glass equation is given below: -

$$x\,(k+1) - x\,(k) = \acute{a}x\,(k - \hat{o}) \,/\, \{1 + x^{\tilde{a}}\,(k - \hat{o})\} - \hat{a}\,x\,(k)$$

The values of the different parameters in the equation are assigned as follows: -

$$\acute{a} = 3, \hat{a} = 1.0005, \tilde{a} = 6, \hat{o} = 3$$

Since $\hat{o} = 3$, four samples are required to obtain a new point. Thus the series is started with four arbitrary samples: -

$$x\,(1) = 1.5, \quad x\,(2) = 0.65, \quad x\,(3) = -0.5, \quad x\,(4) = -0.7$$

Using these samples a series of 200 points is generated and it oscillates within the range -2 to +2. Of these 200 points about nine tenths are fed to the network designed by the 3C algorithm for training. Then the network is tested using the remaining points. In the training and the testing four consecutive points in the series are given as input and the next point is used as the output. Thus a sliding window of size four is used at each and every step. So if nine tenths of the points are to be used for training the total number of sliding windows available is 175, where the first window consists of points 1 to 4 with the 5th point as the output, and the last window consists of points 175 to 178 with the 179th point as output.

The range of the series is divided into 16 equal regions and a point in each region can be represented by the index of the region. These indices ranging from 1 to 16 can be represented using the quaternary encoding scheme. Since four points are required in each training or testing, the 5 character codewords for each of the four inputs are concatenated together. Thus each input vector has 21 elements, where the last element in the vector represents the bias. Unlike the inputs, output points are binary encoded using four bits. This is done to avoid the possibility of generating invalid output vectors that would not belong to the class of expected vectors of the quaternary encoding scheme. Hence 21 neurons are required in the input layer, 175 in the hidden layer (one for each sliding window), and 4 in the output layer.

After the training, the network is tested using the same 175 windows to check its learning ability. Then the rest of the windows are presented to predict future values. The inputs are always points from the original series calculated by the Mackey-Glass equation to avoid an error buildup. The outputs of the network are compared against the expected values in the series. The performance of the 3C algorithm for different values of r is presented in the Figures 5, 6, 7 and 8. The values of r here are 4, 5, 6 and 7 respectively.

In each of the figures only points 160 to 200 are shown for readability. The solid line represents the original series and the lighter line represents the outputs of the network designed by the 3C algorithm. The lighter line from point 160 to 179 shows how well the network has learnt the samples for different values of r. The points predicted by the network are represented by a "×" on the lighter line. The actual points generated by the Mackey-Glass equation are represented by a "o" on the solid line. The first point that is predicted is the point number 180 using the original series points 176, 177, 178 and 179. The next point that is predicted is 181 using the points 177, 178, 179 and 180. The point number 180, which is used as input here, is the original point in the series generated by the Mackey-Glass equation and not the point predicted by the network. Similarly the last point to be predicted is the point number 200 using the actual points 196 to 199 from the series. The network always predicts one point ahead of time and most of the points from 180 to 200 are predicted with very high accuracy. Also the network is able to predict the turning points in the series efficiently. Thus the network is capable of learning the quasi-periodic property of the series. This ability is of great importance in financial applications where predicting the turning point of the price movement is more important than predicting the day to day values.

Stability of networks is another important feature in deciding the network performance and is governed by the consistency of the outputs when network parameters are changed.

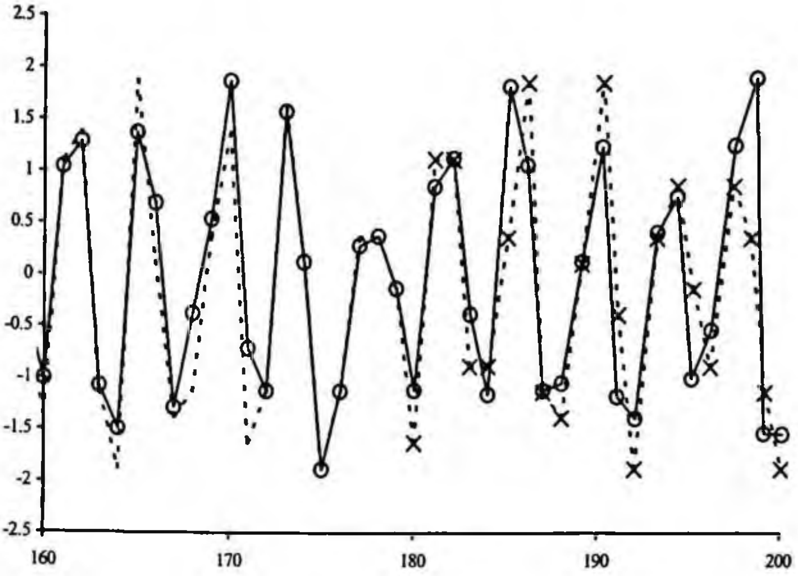


Figure 5. Mackey-Glass time series prediction using 3C, $r = 4$. Dotted line till point 180 – training samples, "o" – Actual data, "×" – predicted data.

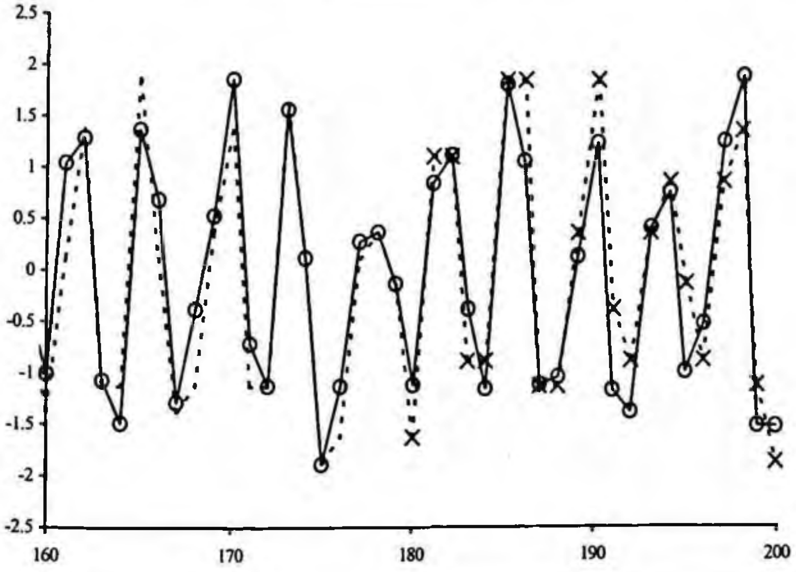


Figure 6. Mackey-Glass time series prediction using 3C, $r = 5$. Dotted line till point 180 – training samples"o" – Actual data, "×" – predicted data.

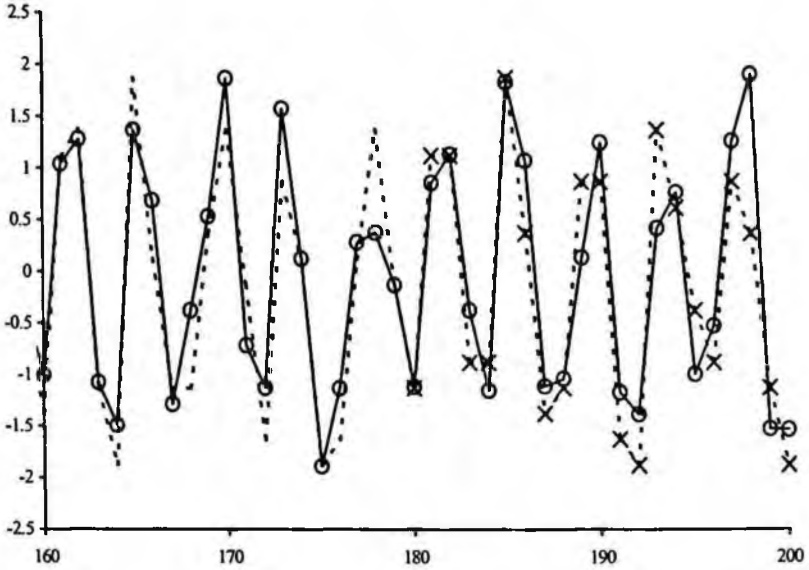


Figure 7. Mackey-Glass time series prediction using 3C, $r = 6$. Dotted line till point 180 – training samples, "o" – Actual data, "×" – predicted data.

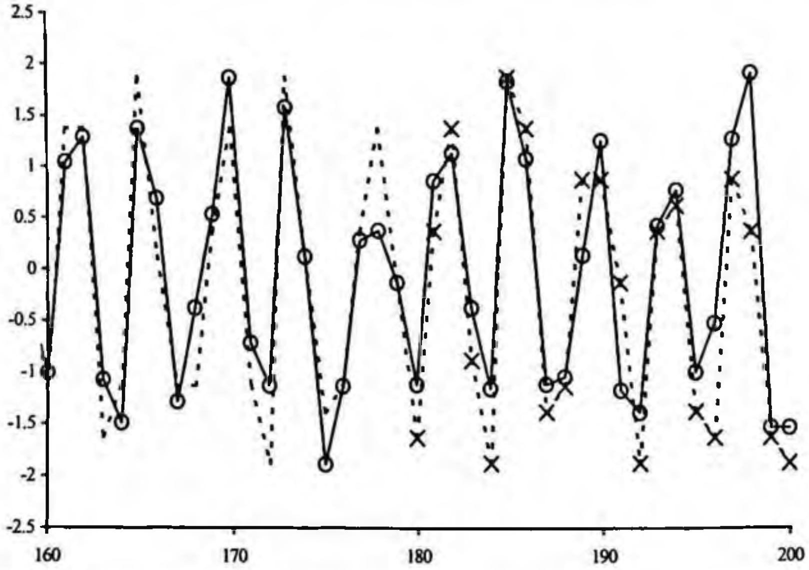


Figure 8. Mackey-Glass time series prediction using 3C, $r = 7$. Dotted line till point 180 – training samples, "o" – Actual data, "×" – predicted data.

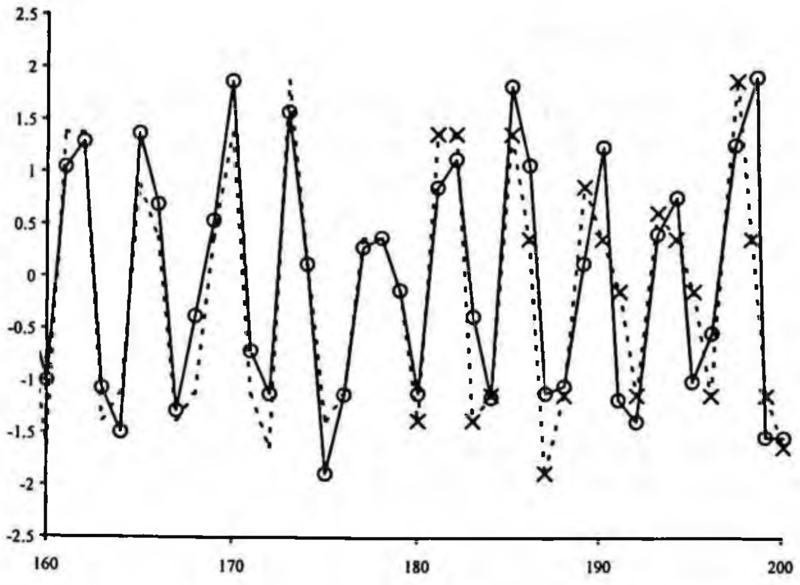


Figure 9. Mackey-Glass time series prediction using 3C, $r = 10$. Dotted line till point 180 – training samples, "o" – Actual data, "×" – predicted data.

The use of different values of r shows its robustness with regard to generalizability. The normalized mean square error of the points predicted for each value of r is shown in Table 7.

Table 7. Normalized mean square error.

Normalized Mean Square Error of predicted points for varying r				
$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 10$
0.0238	0.0112	0.0193	0.0221	0.0275

5 Conclusion

Previously, training of complex input neural networks was done using techniques like the backpropagation and perceptron learning rules. These techniques require considerable time and resources to complete the training. The 3C algorithm, which is a generalization of the CC4 algorithm accomplishes the training instantaneously and requires hardly any resources. Its performance was tested using the pattern classification and time series experiments and its generalization capability was found to be satisfactory.

A new encoding technique called quaternary encoding was introduced. This technique makes some modifications to unary encoding so as to accommodate all four characters of the input alphabet.

Like the CC4, the 3C algorithm has its limitations. First of all it can handle only four input values. Also, as with the CC4, a network of the size required by the 3C algorithm poses a problem with respect to hardware implementation. However its suitability for software implementation due to low requirement of computational resources and its instantaneous training make up for the limitations.

In the future the 3C algorithm should be modified and adapted to handle non-binary complex inputs. This would remove the need for encoding the inputs in many cases and greatly increase the number of areas of application. The 3C algorithm can be applied to applications such as financial analysis and communications and signal processing. In most financial applications it is not enough to predict the future price of an equity or commodity; it is more useful to predict when the directionality of price values will change. One could define two modes of behavior, namely the *up* and *down* trends and represent them by the imaginary 0 and 1. Within a trend, the peak and trough could be represented by the real 0 and 1. This gives us four states namely 0, 1, i and 1+i, which can be presented as inputs to the 3C algorithm.

In communications and signal processing, the complex inputs of many passband modulation schemes could be directly applied to our feedforward network.

References

Azoff, E.M. (1994), *Neural network time series forecasting of financial markets*, John Wiley and Sons, New York.

Benvenuto, N. and Piazza, F. (1992) "On the complex back propagation algorithm", *IEEE Trans. Signal Process*, vol. 40, pp. 967-969.

Georgiou, G.M. (1992), "Parallel Distributed Processing in the Complex Domain", PhD Dissertation, Department of Computer Science, Tulane University.

Kak, S. (1992), "New training algorithm in feedforward neural networks", *First International Conference on Fuzzy Theory and Technology*, Durham, N. C., October 1992. Also in Wang, P.P. (Editor), *Advance in fuzzy theory and technologies*, Durham, N. C. Bookwright Press, 1993.

Kak, S. (1993), "On training feedforward neural networks", *Pramana J. Physics*, vol. 40, pp. 35-42.

Kak, S. (1994), "New algorithms for training feedforward neural networks", *Pattern Recognition Letters*, vol. 15, pp. 295-298.

Kak, S. (1998), "On generalization by neural networks", *Information Sciences*, vol. 111, pp. 293-302.

Kak, S. (2002), "A class of instantaneously trained neural networks", *Information Sciences*, vol. 148, pp. 97-102.

Kak, S. and Pastor, J. (1995), "Neural networks and methods for training neural networks", U. S. Patent No. 5,426,721, June 20, 1995.

Kim, M.S. and Guest C.C. (1990), "Modification of Backpropagation for complex-valued signal processing in frequency domain", *International Joint Conference on Neural Networks*, (San Diego, CA), pp. III-27 – III-31.

Leung, H. and Haykin S. (1991), "The complex back propagation algorithm", *IEEE Trans. Signal Process*, vol. 39, pp. 2101-2104.

Li, C., Liao, X., and Yu, J. (2002), "Complex-Valued Recurrent Neural Network with IIR Neuron Model: Training and Applications", *Circuits Systems Signal Processing*, vol. 21, pp. 461-471.

Little, G.R., Gustafson, S.C. and Senn, R.A. (1990), "Generalization of the back propagation neural network learning algorithms to permit complex weights", *Applied Optics*, vol. 29, pp. 1591-1592.

Noest, A.J. (1988), "Discrete-state phasor neural nets", *Physical Review A*, vol. 38, pp. 2196-2199.

Sutherland, J.G. (1990), "A holographic model of memory, learning and expression", *International Journal of Neural Systems*, vol. 1, pp. 259-267.

Tang, K.W. (1997), "CC4: A new Corner Classification approach to neural network training". Master's Thesis, Department of Electrical and Computer Engineering, Louisiana State University.

Tang, K.W. and Kak, S. (2002), "Fast Classification Networks for Signal Processing", *Circuits Systems Signal Processing*, vol. 21, pp. 207-224.

Authors' address

Pritam Rajagopal and Subhash Kak
Department of Electrical & Computer Engineering
Louisiana State University
Baton Rouge, LA 70803, U.S.A

Chapter 9

Applications of Complex-Valued Neural Networks for Image Processing

Hiroyuki Aoki

We consider a complex-valued neuron model which can take K (K>=2) states on the unit circle in the complex plane as the extension of a well-known binary neuron which can take only two values (1,-1). In the network in which all the complex-valued neurons are fully connected with each other, we determine the state transition rule of each neuron under the condition that the state transition always deceases the network energy by a maximum amount. Next discussed is how a grayscale image can be expressed using the complex-valued neural network obtained above. We present a method for image representation using only phase. This is accomplished through embedding of the amplitude data into the phase data after the 2-dimensional discrete Fourier transform of an image. The embedding of the amplitude data into the phase data can be realized by a phase shift operation. We show that adjusting the phase shift quantity enables us to control the arbitrary frequency component of an input image.

We demonstrate two examples of image processing using complex-valued neural networks applying the aforementioned. They are an image filtering, and a grayscale image associative memory which can memorize grayscale images and recall one from a noisy or imperfect version of the memorized image. Note that we propose two novel types of neuron when implementing these networks. One is a

model in which a phase of complex-valued output signal is shifted by an amount corresponding to the amplitude of a complex-valued input signal. The other is a model, in which the output signal is generated as the projection to the real or imaginary axis of a complex-valued input signal.

1 A Complex-Valued Neuron Model

We consider a fully connected complex-valued neural network as an expansion of the well-known binary Hopfield network. In the Hopfield network, state transition of each neuron is basically stabilized in only two values, so it can handle two-valued images naturally. However, it cannot easily handle multi-valued images or grayscale images. On the other hand, a complex-valued neuron can take multi-values, so introducing it enables us to deal with grayscale images easily. Let us consider the fully connected complex-valued neural network, which is composed of N neurons. Each neuron can take K states on the unit circle in the complex plane (Noest 1988, Jankowski *et al.* 1996, Aizenberg 2000). The state of complex-valued neuron k, denoted by $x(k)$, is given by

$$x(k) = e^{i\theta_K s(k)}, \quad \theta_K = 2\pi/K, \quad s(k) = 0,1,\cdots,K-1. \qquad (1)$$

Figure 1 (a) illustrates a simple case where the number of states K is set at 8. Let $\mathbf{W} = (w_{kj})$ be the weight matrix among neurons, where w_{kj} denotes the connecting weight from neuron j to neuron k. Let $\mathbf{x}$ be a N dimensional column state vector of the complex-valued network. Then the energy function of the neural network for an arbitrary state $\mathbf{x}$ can be defined as follows:

$$E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{*t}\,\mathbf{W}\,\mathbf{x} \qquad (2)$$

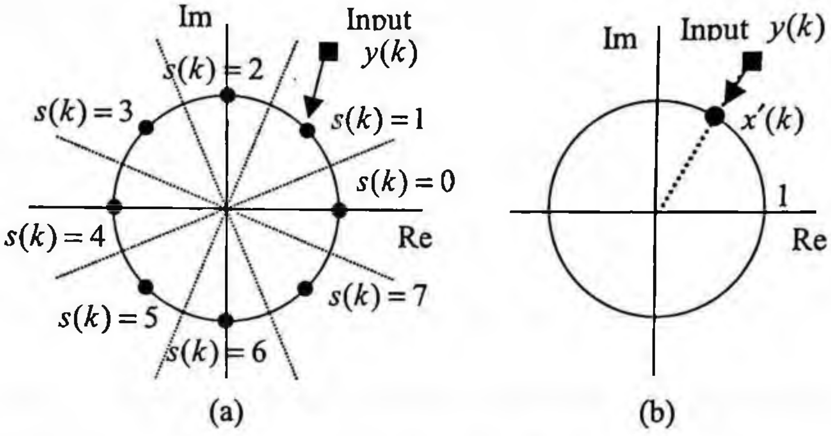


Figure 1. (a) States of a complex-valued neuron ($K=8$) and its state transition rule. (b) The state transition rule for $K \to \infty$.

where $\mathbf{x}^{*\prime}$ denotes conjugate transpose of $\mathbf{x}$. In order to obtain the state transition rule of the complex-valued neuron, we assume the following condition.

Condition: The weight matrix $\mathbf{W}$ is given by an Hermitian matrix $\left(w_{ij}^{*}=w_{ji}\right)$ and the state transition of each neuron always decreases the energy $E(\mathbf{x})$ by a maximum amount.

Energy $E(\mathbf{x})$ is calculated to be a real value since $\mathbf{W}$ is an Hermitian matrix. Now we consider the state transition rule. Let $y(k)$ be the input of neuron k, i.e., $y(k)=\sum_j w_{kj}\,x(j)$ and $x'(k)$ be the renewed state of neuron k. It is assumed that the state vector transits from $\mathbf{x}$ to $\mathbf{x}'$ due to the state transition of neuron k. Energy change ΔE can be calculated as follows:

$$\Delta E=-\mathrm{Re}\left\{\left(x'(k)-x(k)\right)y^{*}(k)\right\}. \tag{3}$$

Furthermore, assuming $x(k) = \exp(i\theta_K s(k))$, $x'(k) = \exp(i\theta_K s'(k))$ and $y(k) = |u| \exp(i\theta_u)$,

$$\Delta E = -|u| \left\{\cos(\theta_K s'(k) - \theta_u) - \cos(\theta_K s(k) - \theta_u)\right\}. \tag{4}$$

Thus, to satisfy the condition mentioned above,

$$\cos(\theta_K s'(k) - \theta_u) > \cos(\theta_K s(k) - \theta_u) \tag{5}$$

and the new state $s'(k)$ may be decided so that $|\theta_K s'(k) - \theta_u|$ may be as small as possible. From this result, we can obtain the following state transition equation for the complex-valued neuron:

$$x'(k) = \operatorname{csgn}(y(k)) \tag{6}$$

where $x'(k)$ is a renewed state after the transition and a function csgn( ) is also defined as follows:

$$\operatorname{csgn}(y(k)) = \exp(i\theta_K s'(k)) \tag{7}$$

$$s'(k) = \left[\frac{\arg(y(k)) + \frac{1}{2}\theta_K}{\theta_K}\right], \quad (\mathrm{mod}\ K) \tag{8}$$

where the symbol $[\theta]$ presents the maximum integer not exceeding θ. The operation of csgn is also illustrated in Figure 1 (a). If K takes a large number, Eqs. (7) and (8) can be expressed as the following:

$$\operatorname{csgn}(y(k)) = \exp(i \arg(y(k))). \tag{9}$$

In this chapter, a neuron having the input-output characteristic described by Eq. (9) is called $n_p(k)$ neuron. The operation is also illustrated in Figure 1 (b). The renewed state is determined by only phase of the input $y(k)$.

In order to use the network as a complex-valued associative memory (CAM), the weight matrix **W** can be given by the following form. Let P vectors $\mathbf{x}^{\alpha}$ $(\alpha = 1, 2, \cdots, P)$ be memorized into CAM and $S = \left(\mathbf{x}^{1}\,\mathbf{x}^{2} \cdots \mathbf{x}^{P}\right)$ be a $N \times P$ matrix, then

$$\mathbf{W} = \mathbf{S}\mathbf{S}^{+}, \quad \mathbf{S}^{+} = (\mathbf{S}\mathbf{S})^{-1}\mathbf{S}^{*\prime} \tag{10}$$

where notation $\mathbf{S}^{+}$ indicates the Moore-Penrose pseudoinverse matrix of $\mathbf{S}$. Note that Eq.(10) is based on the idea of the optimized associative mapping (Kohonen 1987). Making the weight matrix by this equation implies that **W** becomes an Hermitean matrix and enables the memorized vectors to be fixed points of CAM.

2 Image Representation Using Complex-Valued Neurons

In order to represent grayscale images using complex-valued neurons described in the previous section, each element of image data should take the value on the unit circle in the complex plane. So discussed in this section, is how a grayscale image can be represented by using only phase. In order to accomplish this, phase matrix image representation is proposed (Aoki *et al.* 2000).

2.1 Two-Dimensional Discrete Fourier Transform Pair

For the convenience of later discussion, we rewrite the form of the 2-dimensional discrete Fourier transform (2-D DFT) and 2-dimensional inverse DFT (2-D IDFT). The 2-D DFT pair of an $L \times L$ image $u(m,n)$ is defined as

$$v(k,l) = \frac{1}{L} \sum_{m=0}^{L-1} \sum_{n=0}^{L-1} u(m,n)\, e^{-i\theta_L(km+ln)} \ , \quad 0 \le k,l \le L-1 \tag{11}$$

$$u(m,n) = \frac{1}{L} \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} v(k,l)\, e^{i\theta_L(km+ln)} \ , \quad 0 \le m,n \le L-1 \tag{12}$$

where $\theta_L = 2\pi/L$ (Jain. 1989). Here we define a discrete function $f_{(m,n)}(k,l)$ as follows:

$$f_{(m,n)}(k,l) = e^{i\theta_L(km+ln)} . \tag{13}$$

Then a matrix of size $L \times L$ can be defined as

$$\mathbf{f}_{(m,n)} = \left\{ f_{(m,n)}(k,l) \ ; \ 0 \le k,l \le L-1 \right\} . \tag{14}$$

The set of matrices $\mathbf{f}_{(m,n)}\ (m,n = 0,1,\cdots,L-1)$ composes complete orthogonal discrete basis functions. Using Eq.(14), 2-D DFT Eq.(11) and 2-D IDFT Eq.(12) can be expressed as

$$\mathbf{v} = \frac{1}{L} \sum_{m=0}^{L-1} \sum_{n=0}^{L-1} u(m,n)\, \mathbf{f}_{(L-m,\, L-n)} = DFT(\mathbf{u}) \tag{15}$$

$$\mathbf{u} = \frac{1}{L}\sum_{k=0}^{L-1}\sum_{l=0}^{L-1} v(k,l)\,\mathbf{f}_{(k,l)} = IDFT(\mathbf{v}) \tag{16}$$

where $\mathbf{v}$ and $\mathbf{u}$ are $L\times L$ matrices, respectively. They are expressed as a linear combination of the complete orthogonal matrices $\mathbf{f}_{(k,l)}$.

2.2 Phase Matrix Image Representation

Let $v(k,l) = |v(k,l)|\,e^{i\alpha(k,l)}$, where $|v(k,l)|$ represents an amplitude and $\alpha(k,l)$ a phase. In order to represent an image using complex-valued neurons, let an $\alpha(k,l)$ correspond to a state of a complex-valued neuron $x(k,l)$. What we are concerned here with is how to handle $|v(k,l)|$ since the amplitude of the complex-valued neuron should be $|x(k,l)| = 1$. Considering conjugate symmetries, i.e., $v(k,l) = v^{*}(L-k, L-l)$ and $\mathbf{f}_{(k,l)} = \mathbf{f}^{*}_{(L-k,L-l)}$, Eq.(16) can be represented as

$$\begin{aligned}\mathbf{u} &= \frac{1}{L}\sum_{k=0}^{L-1}\sum_{l=0}^{L-1} \mathrm{Re}\left(v(k,l)\,\mathbf{f}_{(k,l)}\right)\\ &= \frac{1}{L}\sum_{k=0}^{L-1}\sum_{l=0}^{L-1} |v(k,l)|\,\mathrm{Re}\left(e^{i\alpha(k,l)}\,\mathbf{f}_{(k,l)}\right).\end{aligned} \tag{17}$$

We next consider the normalization of $|v(k,l)|$. Let $v_{\max}$ be the maximum value of $|v(k,l)|$, i.e.,

$$v_{\max} = \max\left\{\,|v(k,l)|,\ 0 \le k,l \le L-1\right\}. \tag{18}$$

For $0 \le |v(k,l)|/v_{\max} \le 1$, we can calculate $\gamma(k,l)$ as either

$$\gamma(k,l) = \cos^{-1}\left(\left| v(k,l) \right| / v_{\max}\right) \tag{19}$$

or

$$\gamma(k,l) = \sin^{-1}\left(\left| v(k,l) \right| / v_{\max}\right). \tag{20}$$

Then, we can define a new $L \times L$ phase matrix $\mathbf{x} = \{ x(k,l) \}$ as

$$x(k,l) = e^{i(\alpha(k,l)+\gamma(k,l))}. \tag{21}$$

In the equation, note that $|x(k,l)| = 1$ and $x(k,l)$ can be represented as a state of complex-valued neuron. The state is set by shifting the 2-D DFT phase $\alpha(k,l)$ by the phase shift $\gamma(k,l)$ defined in either Eq.(19) or Eq.(20). The amplitude information is embedded into the phase information. In this chapter the matrix $\mathbf{x} = \{x(k,l)\}$ is the phase matrix image representation for the original image $\mathbf{u} = \{ u(m,n) \}$.

On the other hand, the original image $\mathbf{u}$ can be reconstructed from the phase matrix $\mathbf{x}$ in the following way. The IDFT of the phase matrix $\mathbf{x}$ can be represented as

$$\begin{aligned} IDFT\{\mathbf{x}\} &= \frac{1}{L}\sum_{k=0}^{L-1}\sum_{l=0}^{L-1} e^{i(\alpha(k,l)+\gamma(k,l))}\,\mathbf{f}_{(k,l)} \\ &= \frac{1}{L}\sum_{k=0}^{L-1}\sum_{l=0}^{L-1} \mathrm{Re}\left\{e^{i\alpha(k,l)}\,\mathbf{f}_{(k,l)}\right\} e^{i\gamma(k,l)}. \end{aligned} \tag{22}$$

Note that we used the property of the conjugate symmetry, or $\gamma(k,l) = \gamma(L-k, L-l)$ and $\alpha(k,l) = -\alpha(L-k, L-l)$. Using either Eq.(19) or Eq.(20) for $\gamma(k,l)$ calculation, the original image $\mathbf{u}$ given by Eq.(17) can be represented as either

$$\mathbf{u} = v_{\max} \frac{1}{L} \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} \cos\gamma(k,l) \operatorname{Re}\left\{ e^{i\alpha(k,l)} \mathbf{f}_{(k,l)} \right\} \tag{23}$$

or

$$\mathbf{u} = v_{\max} \frac{1}{L} \sum_{k=0}^{L-1} \sum_{l=0}^{L-1} \sin\gamma(k,l) \operatorname{Re}\left\{ e^{i\alpha(k,l)} \mathbf{f}_{(k,l)} \right\}, \tag{24}$$

respectively. Substituting Eq.(22) for Eqs(23) and (24), we can obtain the following relation as either

$$\mathbf{u} = v_{\max} \operatorname{Re}\{IDFT(\mathbf{x})\} \tag{25}$$

or

$$\mathbf{u} = v_{\max} \operatorname{Im}\{IDFT(\mathbf{x})\}, \tag{26}$$

respectively. In the above equations, the variable $v_{\max}$ is just a constant value. From Eqs.(25) and (26), it follows that the original image $\mathbf{u}$ can be obtained by extracting either the real part or the imaginary part from the 2-D IDFT of the phase matrix $\mathbf{x}$.

2.3 Block Diagrams for the Phase Matrix Transform Pair

In this section we display block diagrams for the transformations between an original image and its phase matrix described in the previous section. Figure 2 and Figure 3 show the block diagrams for the transform of an image and the inverse transform of a phase matrix, respectively. What follows is an explanation of the individual blocks in the diagrams.

The transform of an image into its phase matrix :

1. The 2-D DFT of an $L \times L$ image $u(m,n)$ is performed. A complex number $v(k,l) = |v(k,l)| e^{i\alpha(k,l)}$ is obtained. For each $v(k,l)$, amplitude data $|v(k,l)|$ and phase data $\alpha(k,l)$ are separated. A $v(k,l)$ at $k=0$ and $l=0$ is calculated from Eq. (11) as
$$v(0,0) = \frac{1}{L} \sum_{m=0}^{L-1} \sum_{n=0}^{L-1} u(m,n) \tag{27}$$
and is called the Direct Current(DC) or zero-frequency component, which affects the brightness or bias of the whole image. We set $v(0,0) = 0$. This eliminates the variation in brightness or bias of the input image $u(m,n)$.
2. For amplitude data $|v(k,l)|$, $v_{\max}$ is selected. Let us assume here that the value of $v_{\max}$ is determined as a common parameter for all images, although the value of $v_{\max}$ is essentially different in each image.
3. Phase shift data $\gamma(k,l)$ is calculated by using either Eq.(19) or Eq.(20).
4. For phase data $\alpha(k,l)$, phase shift operation $\alpha(k,l) + \gamma(k,l)$ is performed. As a result, a phase matrix $\{x(k,l)\}$ is generated.
5. The quantized phase matrix $\{x_q(k,l)\}$ is quantized by the following (if it is necessary):
$$x_q(k,l) = \mathrm{csgn}\left(x(k,l)\right) . \tag{28}$$

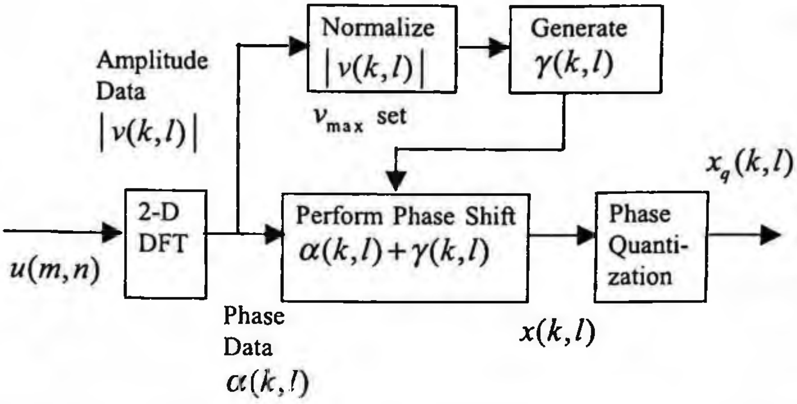


Figure 2. Block diagram for the transform of an image into its phase matrix.

The inverse transform of a phase matrix into its image:

1. The 2-D IDFT of a state of the complex valued neural network is performed. This provides a complex-valued matrix $\{u_1(m,n)\}$.
2. Either the real part or the imaginary part of the matrix $\{u_1(m,n)\}$ represents the output image $\{u_{out}(m,n)\}$.

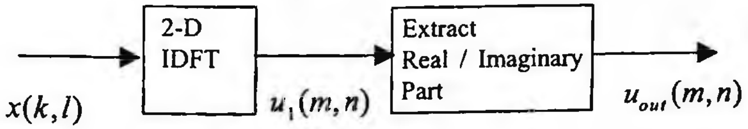
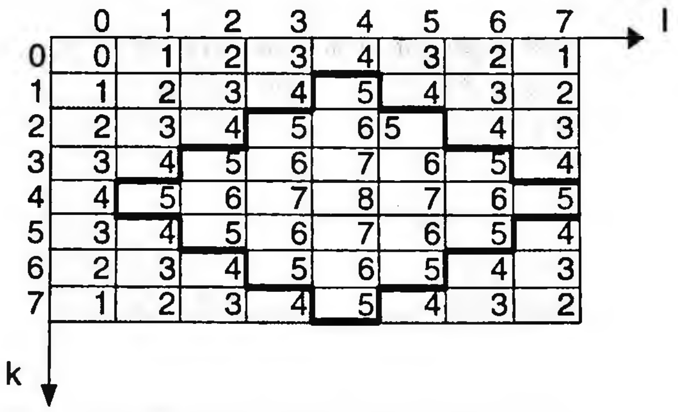


Figure 3. Block diagram for the inverse transform of a phase matrix into its image.

2.4 Reducing the Number of Complex-Valued Neurons

In a natural scene image, energy of the image tends to center on the low frequency region. Using these properties, we can remove the high frequency components in the phase matrix at the degree in which the image degradation is not conspicuous. As a result, it is possible to reduce the number of neurons. Let us explain a case in the concrete using an 8×8 image. Figure 4 shows the frequency numbers of the DFT coefficients $v(k,l)$ in the case of $L=8$. The range of the frequency numbers is from 0 to 8. The frequency number 0 is called the Direct Current (DC) or zero-frequency. The frequency number 8 is the highest frequency. A DFT coefficient $v(k,l)$ corresponds to a neuron $n_p(k,l)$. For example, if we remove the high side frequency numbers 5 to 8, only the low side ones 0 to 4 (shaded area) are employed to build a neural network. As a result, the number of employed neurons can be reduced from 64 to 39.



	0	1	2	3	4	5	6	7
0	0	1	2	3	4	3	2	1
1	1	2	3	4	5	4	3	2
2	2	3	4	5	6	5	4	3
3	3	4	5	6	7	6	5	4
4	4	5	6	7	8	7	6	5
5	3	4	5	6	7	6	5	4
6	2	3	4	5	6	5	4	3
7	1	2	3	4	5	4	3	2

Figure 4. Frequency numbers of the DFT coefficients $v(k,l)$ for an 8×8 image.

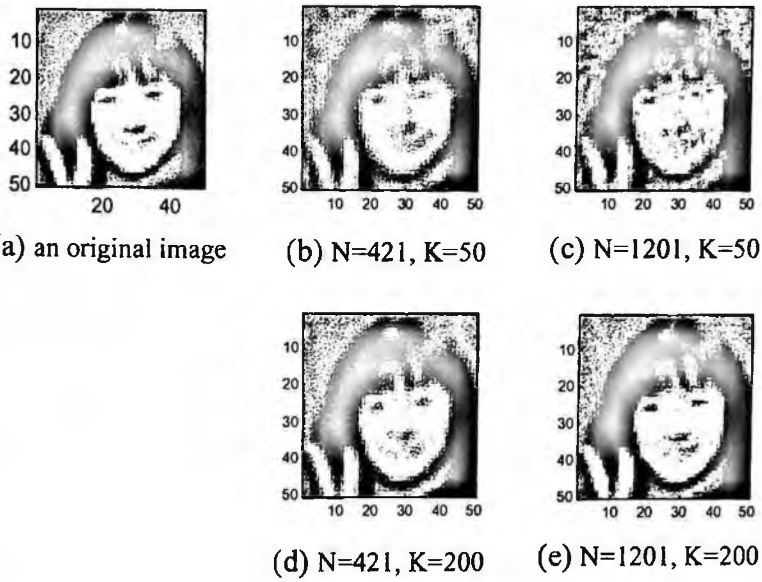


Figure 5. (a) An original image. (b)-(e) Reconstructed images from the quantized phase matrices of the original image in various conditions.

We next examine the relation between image quality and both the number of neurons N and the number of states K (phase quantization). Figure 5 shows the images reconstructed from the quantized phase matrices in various conditions. The original image is 50×50 with $[0,\ 255]$ integer-valued gray levels as shown in Figure 5 (a). It has 2500 frequency components. Removing $v(k,l)$ with high frequency numbers, we can reduce the number of neurons. Although the number of neurons is reduced to 1201 from 2500 especially in Figure 5 (e) image, image degradation is not conspicuous. From these reconstructed images, we can see that 1) it is required that K takes the value of more than 200 to obtain a clear image, however, 2) image degradation is not conspicuous, even if around 50% of all components in the high frequency bands are removed. Therefore, we can reduce the number of neurons.

3 Phase Matrix Transform Pair implementation Using Complex-Valued Neural Networks

3.1 A Phase Matrix Transform Neural Network

In this section let us consider the implementation of the phase matrix transform described in the previous section using complex-valued neural networks. Let us use Eq.(20) and Eq.(26) as the calculation of $\gamma(k,l)$ and the image reconstruction, respectively. Suppose that $|v(k,l)|/v_{\max} << 1$ holds, Eq.(20) is given by the following approximation:

$$\gamma(k,l) = |v(k,l)| / v_{\max} . \tag{29}$$

Then, each element in the phase matrix given by Eq.(21) can be rewritten as the following:

$$x(k,l) = \exp\left\{ i\left(\arg v(k,l) + |v(k,l)| / v_{\max} \right) \right\}. \tag{30}$$

Here, we introduce a novel complex-valued neuron model denoted by $n_A(k,l)$, whose input-output equation is defined by Eq.(30). In the equation, the variables $v(k,l)$ and $x(k,l)$ represent complex-valued input and output of the neuron, respectively. In the neuron, output is determined by shifting the input phase by the phase shift quantity which is proportional to the input amplitude.

Figure 6 shows the input-output operation of the n_A neuron model. Introducing the n_A neuron model easily enable us to perform the phase matrix transform using the complex-valued network as shown in Figure 7. Neurons $n_A(k,l)$ and pixels of an image $u(m,n)$ are linked with the connecting weight described by the function $f_{(k,l)}(L-m, L-n)$ defined by Eq.(13). Since $\left|f_{(k,l)}(L-m, L-n)\right|$ equals 1, only phase operation is performed and the amplitude information is invariant in all connecting weights. The phase matrix can be obtained from $n_A(k,l)$ neurons' output.

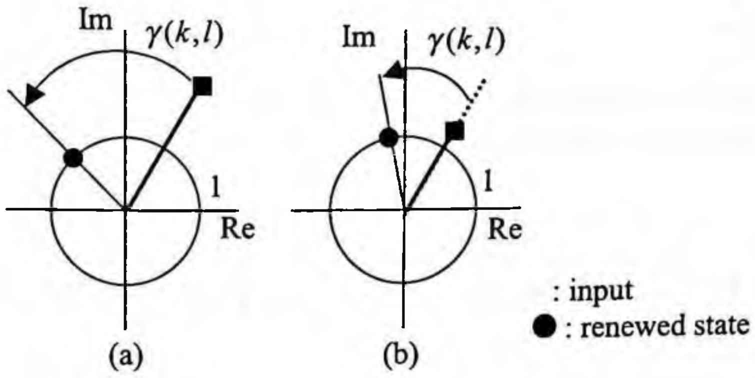


Figure 6. State renewal operation of a n_A neuron model. (a) The amount of phase shift is large. (b) The amount of phase shift is small.

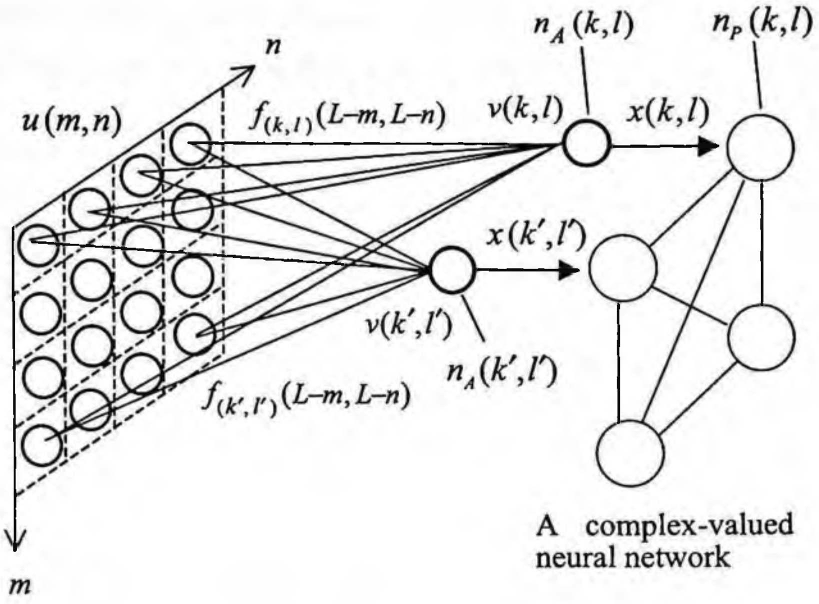


Figure 7. A complex-valued neural network to perform the phase matrix transform of an image $u(m,n)$ using n_A neuron models.

3.2 An Inverse Phase Matrix Transform Neural Network

Next, we can deal with the implementation of the inverse phase matrix transform using complex-valued neural networks. The output image $\mathbf{u}_{out}$ is obtained by Eq.(26). The equation for a pixel value $u_{out}(m,n)$ is rewritten as the following:

$$u_{out}(m,n) = v_{\max}\frac{1}{L}\operatorname{Im}\left\{\sum_{k=0}^{L-1}\sum_{l=0}^{L-1} x(k,l) f_{(k,l)}(m,n)\right\}. \qquad (31)$$

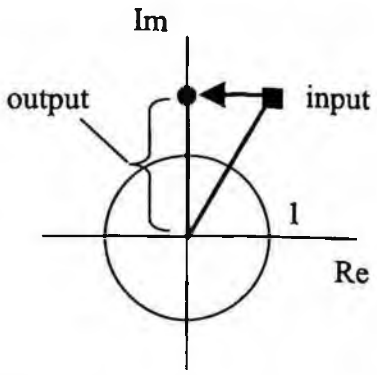


Figure 8. Input-output operation of a n_B neuron model.

Here, we introduce another type of complex-valued neuron model denoted by $n_B(m,n)$, whose input-output equation is given by the following:

$$x'(m,n) = \frac{v_{\max}}{L}\,\mathrm{Im}\{\, y(m,n) \,\}, \tag{32}$$

where the variables $y(m,n)$ and $x'(m,n)$ represent complex-valued input and real-valued output, respectively. The output of $n_B(m,n)$ neuron model is obtained from the projection onto the imaginary axis of the input. Figure 8 shows the input and output operation of the n_B neuron model. Introducing it enables us to perform the processing described by Eq.(31) using the complex-valued neural network shown in Figure 9. The output images can be obtained from output of n_B neurons directly.

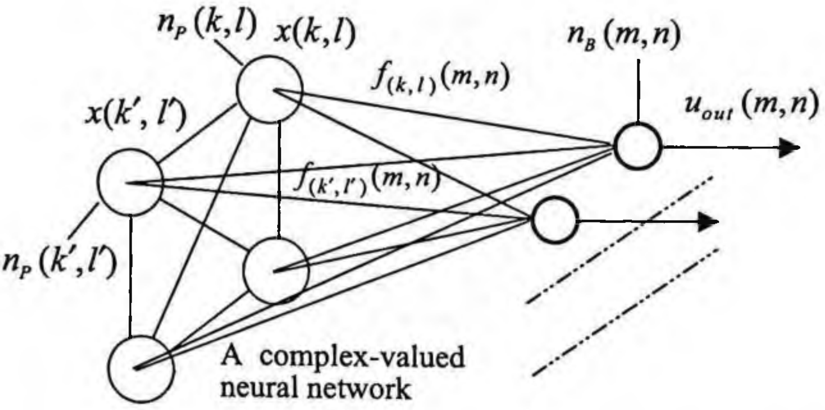


Figure 9. A complex-valued neural network to perform the inverse phase matrix transform using n_B neuron models.

4 Examples of Image Processing Using Complex-Valued Neural Networks

In this section, we present two examples of image processing using complex-valued neural networks.

4.1 DFT Filtering

Figure10 shows the basic idea of the image enhancement by DFT filtering. A pixel-by-pixel multiplication is performed on a transformed image followed by the inverse transformation.

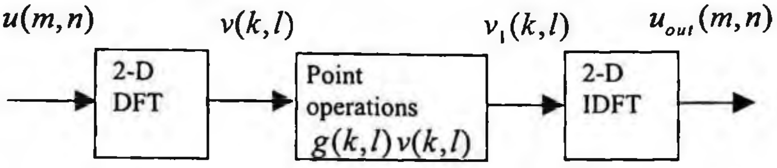


Figure 10. Image enhancement by DFT filtering.

The function of the DFT filtering shown by Figure 10 can be realized by using the complex-valued neural network which is composed of n_A and n_B neurons as shown in Figure 11. The information of $g(k,l)$ which is called a zonal mask is reflected to $\gamma(k,l)$ as the following:

$$\gamma(k,l) = g(k,l)\,|v(k,l)|/v_{\max}\,. \tag{33}$$

The operation of multiplication by $g(k,l)$ shown in Figure 10 can be replaced with the operation of a phase shift in $n_A(k,l)$ neuron. Adjusting the phase shift data in n_A neurons enables us to control freely the arbitrary frequency component of an input image $\mathbf{u}$. Consequently, image filtering can be realized. Remember that $\gamma(k,l) = \gamma(L-k, L-l)$ holds because of the conjugate symmetry of $v(k,l)$. Especially if we set $\gamma(k,l) = \gamma(L-k, L-l) = 0$, then the corresponding frequency components $\mathbf{f}_{(k,l)}$ and $\mathbf{f}_{(L-k,L-l)}$ can be eliminated from the output image $\mathbf{u}_{out}$. Alternative means of this is to replace $n_A(k,l)$ neurons with $n_p(k,l)$ neurons, respectively.

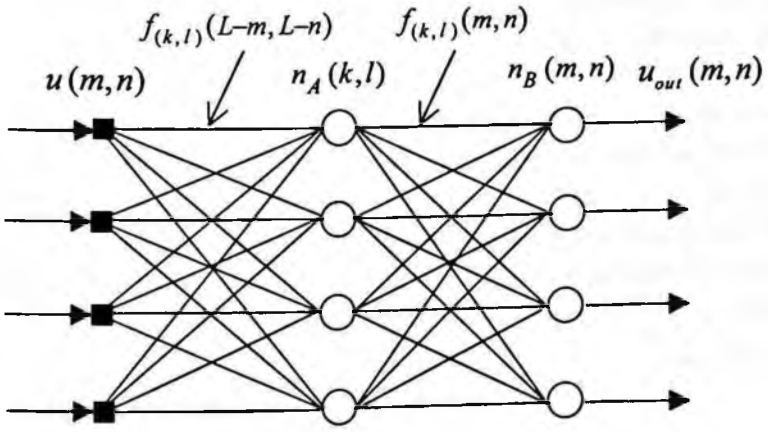


Figure 11. DFT filtering using complex-valued neural networks.

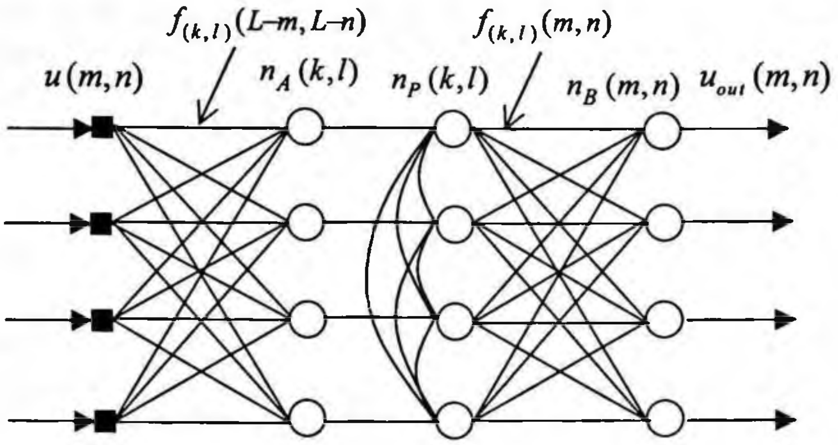


Figure 12. A Grayscale image associative memory using complex-valued neural networks.

4.2 A Grayscale Image Associative Memory

A grayscale image associative memory can be realized by combining use of CAM with the phase matrix transform and the inverse phase matrix transform. The block diagrams are as shown in Figure 12. In general, there are two phases to the operation of associative memory; they are 1) the storage phase, and 2) the retrieval phase. First, in the storage phase, memorized images are input through the phase matrix transform, and then connecting weights among n_P neurons can be, for example, determined by using Eq.(10). Secondly, in the retrieval phase, CAM will retrieve one of the corresponding complete stored images when a noisy or incomplete version of a memorized image is given as an initial image. The output image $\mathbf{u}_{out}$ is obtained through inverse phase matrix transform.

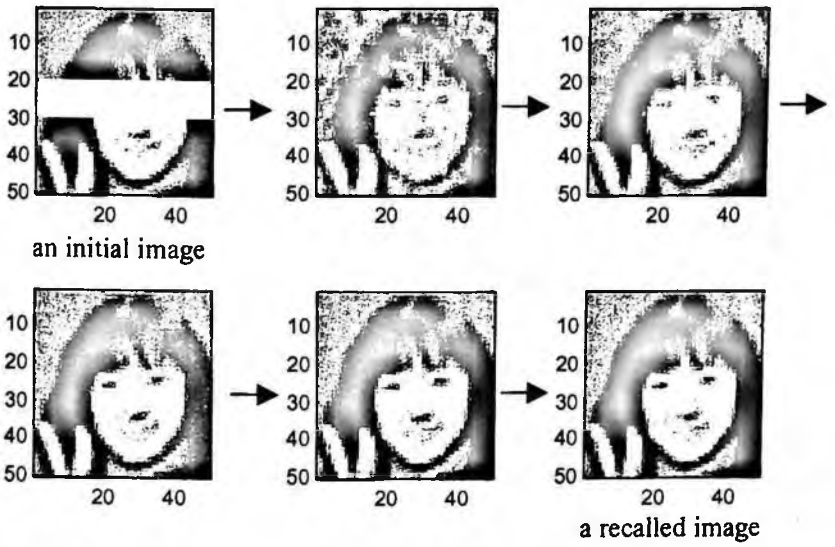


Figure 13. A recalling process of CAM, when a partial broken version of a memorized grayscale image shown in Figure 5 (a) is given as an initial image.

Let us here build a grayscale image associative memory which memorizes 120 50×50 images including the image shown in Figure 5 (a). The numbers of neurons N and of states K (phase quantization), are set at 1201 and 200, respectively. We removed 52% of all components in the high frequency bands from images. Figure13 displays a recalling process when a partial broken version of a memorized image is given as an initial image. The complete image is perfectly recalled in 18 iterations. The synchronous operation is used when iterating. (All neurons are updated simultaneously.)

Figure 14 displays the dependence of signal-to-noise ratio (SNR) in rate of successful recall when a noisy version of a memorized image is given as an initial image.

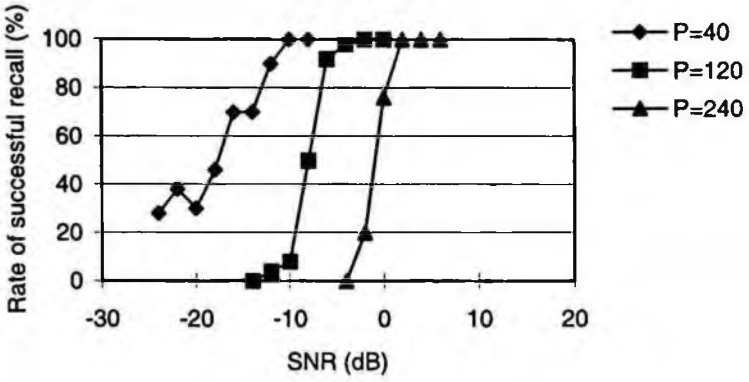


Figure 14. Dependence of SNR in rate of successful recall in the grayscale image associative memory.

The noisy image is generated by adding some white Gaussian noise to the memorized images. The SNR is defined in decibels (dB) as

$$SNR = 10\log_{10}\frac{\sigma^2}{\sigma_n^2} \tag{34}$$

where σ^2 and σ_n^2 are the variance of the original image and noise data, respectively. For three different numbers of memorized images, i.e., $P = 40,\ 120, 240$, simulations are performed. In Figure 14 the horizontal axis indicates the SNR, signal-to-noise ratio. In the number of SNR increases, the noise level is getting smaller. The vertical axis indicates the rate of successful recall. We call an event a success if the memorized image can be retrieved perfectly. In the case of $P = 240$, if the SNR is more than 2dB, the rate of successful recall reaches almost 100 %. However, if the SNR is less than -4dB, successful recall is almost impossible. The actual noisy images with 2dB and -4dB additive white Gaussian noise for the original image shown in Figure 5 (a) are shown in Figure 15 (a) and (b), respectively.

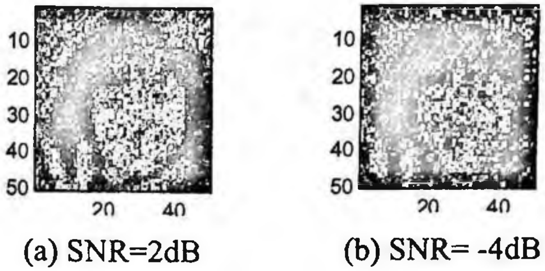


Figure 15. Images degraded by 2 dB and -4dB additive white Gaussian noise for the original image shown in Figure 5 (a).

5 Conclusions

We presented applications of complex-valued neural networks for image processing showing two examples: an image filtering and a grayscale image associative memory. We presented a method for image representation using only phase in order to easily deal with grayscale images using complex-valued neural networks. The following three types of complex-valued neuron model are demonstrated in this chapter:

$$n_p \text{ model}: \; x'(k,l) = \exp\{\, i \arg(y(k,l)) \,\},$$

$$n_A \text{ model}: \; x'(k,l) = \exp\{\, i\big(\arg(y(k,l)) + a\,|\,y(k,l)\,|\,\big) \},$$

$$\text{and } n_B \text{ model}: \; x'(k,l) = b\,\mathrm{Im}\{\, y(k,l) \,\},$$

where $y(k,l)$ and $x'(k,l)$ are input and output, and a and b are constants, respectively. In an image filtering using n_A models, letting a equal $a(k,l)$ (let a take a different value for each neuron) enables us to control the amplitude of the arbitrary frequency component of an image. Especially, setting $a(k,l) = a(L-k, L-l) = 0$, we can remove frequency components $\mathbf{f}_{(k,l)}$ and $\mathbf{f}_{(L-k,\,L-l)}$ from the image. We also showed that a grayscale image associative memory can be realized by combining a complex-valued associative memory composed of n_p neurons with both pre-processing using n_A neurons and post-processing using n_B neurons.

References

Aizenberg, I. N., Aizenberg, N. N. and Vandewalle, J. (2000), Multi-valued and universal binary neurons, Kluwer Academic Publishers.

Aoki, H, Azimi-Sadjadi, M. R. and Kosugi, Y. (2000), "Image association using a complex-valued associative memory model," IEICE Trans. vol.e83A, pp. 1824-1832.

Jain, A. K. (1989), Fundamentals of digital image processing, Prentice Hall.

Jakkowski, S., Lozowskik, A. and Zurada, J. M. (1996), "Complex-valued multistate neural associative memory," *IEEE Trans. Neural Networks*, vol. 7, pp. 1491-1496.

Kohonen, T. (1987), Self-Organization and Associative Memory, 2nd ed., Springer-Verlag, New York.

Noest, A. J. (1988), "Discrete-state phasor neural netw*orks," American Phsysical Society,* vol.38, pp.2196-2199.

Author's address
Hiroyuki Aoki: Dept. Electronic Engineering, Tokyo National College of Technology, 1220-2 Kunugida-machi, Hachioji-shi, Tokyo 193-0997 Japan

Chapter 10

Memorization of Melodies Using Complex-Valued Recurrent Neural Network

Makoto Kinouchi and Masafumi Hagiwara

To deal with temporal sequences is very important and difficult problem for applications of neural networks. In this chapter, we aim at constructing novel recurrent neural network which can process temporal sequences. The memorization ability of temporal sequences can be used in a lot of fields e.g. control, information processing, thinking support systems, and so on.

In section 1, the background and the purposes of this chapter is described.

In section 2, a Multilayer Network using Complex neurons with local Feedback (MNCF) is explained. A complex neuron model can keep previous information more easily than a conventional neuron models because of the phase component. A simple learning algorithm based on the back-propagation for temporal sequences which is named Complex Back-Propagation for Temporal sequences (CBPT) is derived. It can be considered as a generalized original back-propagation. It is shown in some computer simulations that the network has better ability than the conventional ones, including Elman's network.

In sections 3 and 4, a music retrieval system using a complex-valued recurrent neural network which is named MUSIC (MUltilayer network for Sequential Inputs using Complex neurons) is described. In the system, melodies are treated as temporal sequences. In the conventional associative memory models, melodies should be given at a time and they are treated as static patterns. MUSIC can treat a number of melodies by some smaller networks. Such architecture has an advantage that the pattern matching process is not required. MUSIC uses a part of the melodies as a key instead of the text information.

In section 5, the results mentioned in this chapter are concluded.

1 Introduction

A lot of associative memory models using neural network have been proposed. Most of them are devised to memorize static patterns. Let's think about human memory. We can memorize not only static patterns like pictures but also temporal sequences like movies, speech, melodies, and so on. For example, when we memorize a melody, we can recall it from a part. Such a behavior comes from our associative function of brain for temporal sequences. Memorization of temporal sequences is very important not only from a theoretical point of view but also from the applications: it can be used in a lot of fields e.g. control, information processing such as database of melodies and database of movies, thinking support systems, and so on.

The conventional techniques to deal with temporal sequences can be classified into the following 3 methods:

1. Turn the temporal sequences into spatial patterns and they are processed by a conventional static network.

2. Addition of time-delay elements to feedforward connections: One of the examples is the Time Delay Neural Networks (TDNN) (Waibel *et al.* 1989, Day and Davenport 1993). The longest time length to be dealt is limited by the longest delayed path.

3. Addition of time-delay elements to feedback connections: Recurrent networks are their examples (Jordan 1986, Frasconi *et al.* 1992, Nishi *et al.* 1993, Parlos *et al.* 1994, Tabuse *et al.* 1997). Several learning algorithms have been proposed. This technique has an ability to deal with longer temporal sequences because of the recurrent signal flow. However, the number of connections increases as the network becomes large. In addition, the learning algorithm also tends to be complicated and long training time is required.

Methods 1 and 2 can only deal with the limited lengths of the sequences. Moreover, method 1 cannot treat the temporal sequences in an on-line manner. Method 3. has an ability to deal with longer temporal sequences because of the recurrent signal flow.

From a technical point of view, it can be considered that the rest of the melody can be determined by leading some notes. However, the numbers of the notes which can determine the rest of the melody are not always the same. Thus method 3 is the most suitable for the purpose.

We have proposed novel multilayer neural networks using complex neurons with local feedback to deal with temporal sequences (Kinouchi and Hagiwara 1995). Since a complex number has a phase component, it can express a time concept. And the local feedback enables a recurrent signal flow. Therefore the combination of complex neurons and recurrent networks is an excellent method to deal with a time concept.

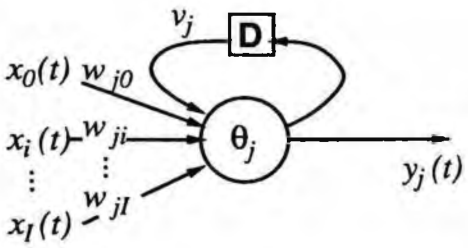


Figure 1. Complex neuron with local feedback.

2 A Recurrent Neural Network Using Complex Value

In this section, we describe novel multilayer neural networks using complex neurons with local feedback to deal with temporal sequences. It is named MNCF (Multilayer Network using Complex neurons with local Feedback).

2.1 Multilayer Network Using Complex Neurons with Local Feedback (MNCF)

Figure 1 illustrates a complex neurons with local feedback. A complex neuron uses complex numbers in the input, output, connection weights and threshold. The output of jth neuron in the nth layer at time t can be expressed by

$$y_j^{(n)}(t) = f(u_j^{(n)}(t)) \tag{1}$$

$$u_j^{(n)}(t) = \sum_i w_{ji}^{(n)} y_i^{(n-1)}(t) + v_j^{(n)} y_j^{(n)}(t-1) + \theta_j^{(n)} \tag{2}$$

where $u_j^{(n)}(t)$ is the input sum to the neuron, $w_{ji}^{(n)}$ are the connection weight from the ith neuron in the $(n-1)$th layer to the jth neuron in the nth layer. $v_j^{(n)}$ is the feedback weight and $\theta_j^{(n)}$ is the threshold. $f(x)$ is the following function for a complex number.

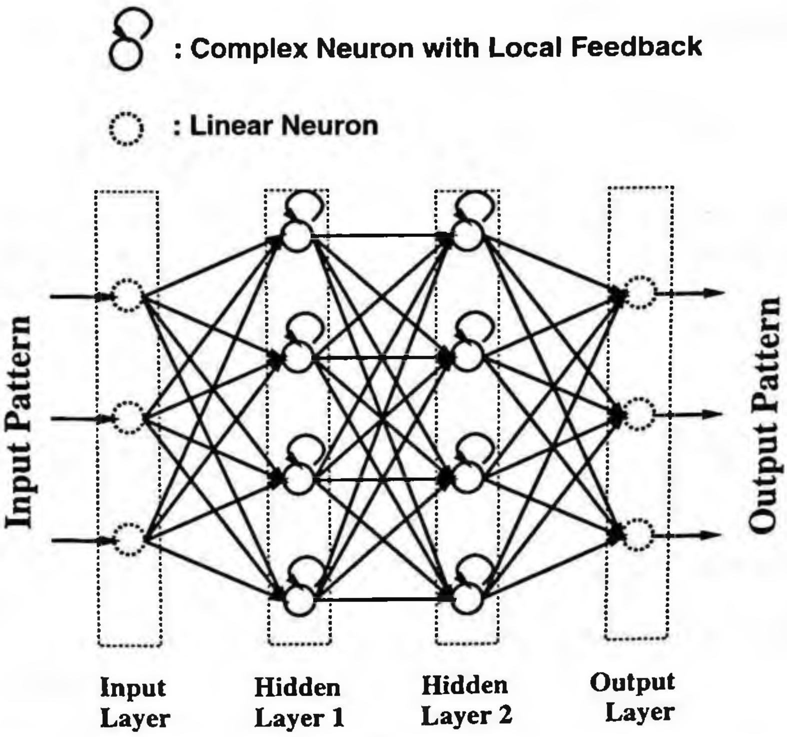


Figure 2. Multilayer Network using Complex neurons with local Feedback (MNCF).

$$f(x) \equiv g(\mathrm{Re}\, x) + jg(\mathrm{Im}\, x) \tag{3}$$

We use $g(x) = \tanh(x)$.

Figure 2 shows the structure of the Multilayer Network using Complex neurons with local Feedback (MNCF).

2.2 Complex Back-Propagation for Temporal Sequences (CBPT)

In this section, a learning algorithm for the network is briefly explained. The algorithm is based on the back-propagation.

Changing each $w_{ji}^{(n)}$ at time t is expressed as

$$\Delta w_{ji}^{(n)}(t) = -\eta \sum_{\tau=0}^{T-1} \left\{ \frac{\partial E(t)}{\partial y_j^{(n)}(t-\tau)} \frac{\partial y_j^{(n)}(t-\tau)}{\partial w_{ji}^{(n)}} \right\} \tag{4}$$

where η is the learning coefficient and T is the time length to be considered for learning. $T = \infty$ corresponds to the gradient descent rule in a strict sense. $E(t)$ is the error function at time t, which is defined as

$$E(t) \equiv \frac{1}{2} \sum_i |d_i^{(N)}(t) - y_i^{(N)}(t)|^2 \tag{5}$$

where $d_i^{(N)}(t)$ is the desired output of ith neuron in the Nth (output) layer.

We define $\delta_j^{(n)}(t,\tau)$ as follows:

(i) For hidden layers $(n \neq N)$

$$\delta_j^{(n)}(t,\tau) = \begin{cases} 0 & (\tau < 0) \\ f'(u_j^{(n)}(t-\tau)) * \Big\{\sum_k \overline{w_{kj}^{(n+1)}} \delta_k^{(n+1)}(t,\tau) \\ \qquad + \overline{v_j^{(n)}} \delta_j^{(n)}(t,\tau-1)\Big\} & (\tau \geq 0) \end{cases} \tag{6}$$

(ii) For an output layer $(n = N)$

$$\delta_j^{(n)}(t,\tau) = \begin{cases} 0 & (\tau \neq 0) \\ -\{d_j^{(N)}(t) - y_j^{(N)}(t)\} & (\tau = 0) \end{cases} \tag{7}$$

where $\overline{a}$ is a complex conjugate of a, operator $*$ is defined as

$$x * y \equiv (\text{Re}\, x\ \text{Re}\, y) + j(\text{Im}\, x\ \text{Im}\, y) \tag{8}$$

and $f'(x)$ is defined as

$$f'(x) \equiv \left\{ \frac{\partial}{\partial \text{Re}\, x} + \frac{\partial}{\partial \text{Im}\, x} \right\} f(x). \tag{9}$$

By using $\delta_j^{(n)}(t,\tau)$, Eq.(4) can be expressed as

$$\Delta w_{ji}^{(n)}(t) = -\eta \sum_{\tau=0}^{T-1} \overline{y_i^{(n-1)}(t-\tau)}\delta_j^{(n)}(t,\tau). \tag{10}$$

Changing $v_j^{(n)}$ and $\theta_j^{(n)}(t)$ is derived in the same way,

$$\Delta v_j^{(n)}(t) = -\eta \sum_{\tau=0}^{T-1} \overline{y_j^{(n)}(t-1-\tau)}\delta_j^{(n)}(t,\tau) \tag{11}$$

$$\Delta \theta_j^{(n)}(t) = -\eta \sum_{\tau=0}^{T-1} \delta_j^{(n)}(t,\tau). \tag{12}$$

Since this is a generalized algorithm, it can be reduced to the conventional real-valued back-propagation algorithm (Rumelhart *et al.* 1986).

2.3 Computer Simulation Results

We show computer simulation results of the network. We made temporal patterns by the following equation,

$$x(t) = \left\{\sum_{k=1}^{m} x(t-k)\right\} \bmod n \tag{13}$$

where $x(t)$ has an integer value and forms a cyclic sequence determined by m and n. We give some examples.

$m = 3, n = 3:$

$\ldots, \underline{1,0,0,1,1,2,1,1,1,0,2,0,2}, 1,0,0,\ldots$

(repeat)

$m = 4, n = 3:$

$\ldots, \underline{1,0,0,2,0,2,1,2,2,1,0,2,2,2,0,0,1,0,1,2,1,1,2,}$

$\underline{0,1,1}, 1,0,0,\ldots$ (repeat)

Table 1. Processing time for an epoch (MNCF=100).

T	MNCF	LFMN	RMLP		Elman's
			(full connection)	(coupled)	
1	100	69	152	79	55
2	166	117	283	134	103

Table 2. Numbers of neurons.

	input layer	hidden layers		output layer
MNCF	3	25	25	3
LFMN, RMLP	3	50	50	3
Elman's	3 + 50	50		3

$$m = 3, n = 5:$$

$$\ldots, \underline{1,3,0,4,2,1,2,0,3,0,3,1,4,3,3,0,1,4,0,0,4,4,3,}$$

$$\underline{1,3,7,1,1,4,1,1}, 1,3,0,\ldots \qquad \text{(repeat)}$$

The network is learned to predict $x(t+1)$ using $x(t)$. During the recall process, after some samples are presented to the network, the output of the network can be used for the next input. In this way, the network can recall a whole sequence. This ability is, so to speak, compared to memorizing melodies and recalling them from a part.

We compared the network (MNCF: Multilayer Network with Complex neurons with local Feedback) with the following networks:

1. Local Feedback Multilayered Network (LFMN)(Frasconi *et al.* 1992): The LFMN has local feedbacks in hidden layers. The layer is called dynamic layer.

2. Recurrent Multilayer Perceptron (RMLP)(Parlos *et al.* 1994): The RMLP is a multilayer perceptron having delayed connections among the neighboring neurons in a hidden layer. We prepared

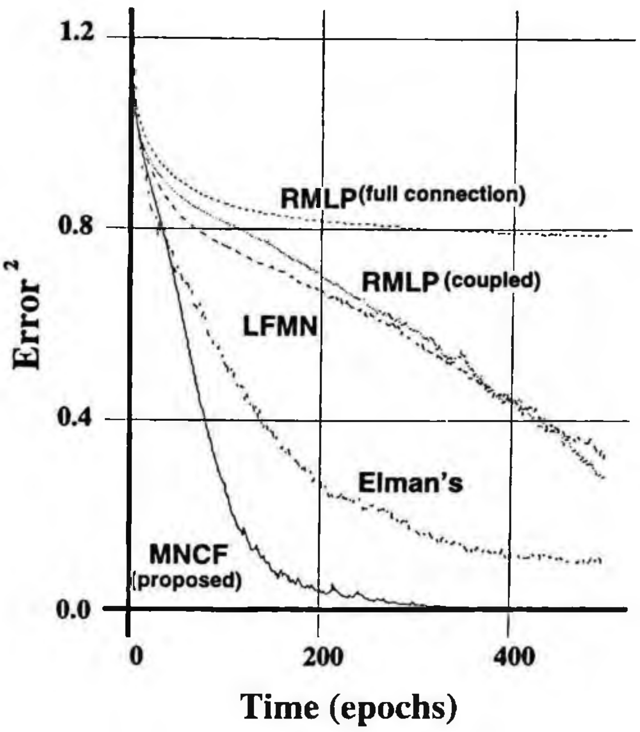


Figure 3. Learning Curve when $T = 1$.

two types: One is 'full connection' type in which all the neurons in a hidden layer are fully connected; another is 'coupled' type in which coupled neurons are interconnected. The latter one is similar to MNCF, because a complex neuron is composed of a real neuron and an imaginary neuron and they are interconnected.

3. Elman's Network (Elman 1990): There is a context layer which is copied from a hidden layer.

We did not use Jordan's network (Jordan 1986) because the problem is not suited to the Jordan's network. Learning algorithms for the networks (1)-(3) are derived from Eq.(4).

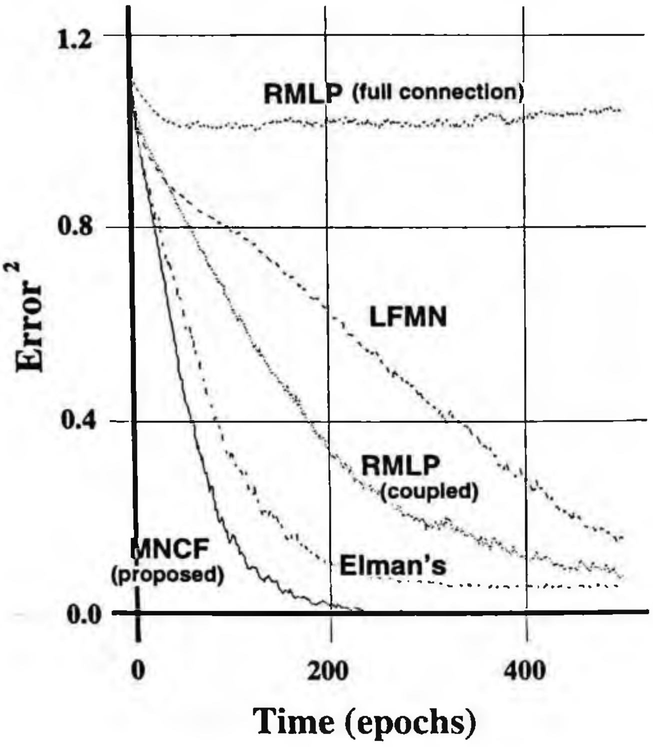


Figure 4. Learning Curve when $T = 2$.

Table 1 shows the processing time for an epoch which is one period of $x(t)$. Each network used the same number of neurons as shown in Table 2.

Figure 3 and figure 4 show the learning curves, where $m = 5$ and $n = 3$ in both cases. Each line is based on average of 100 trials. It can be observed from these figures that learning of MNCF is very fast and that the residual error is extremely small. In addition, learning curve of MNCF when $T = 1$ is better than those of the other networks even when $T = 2$. This is owing to the superior learning ability of MNCF.

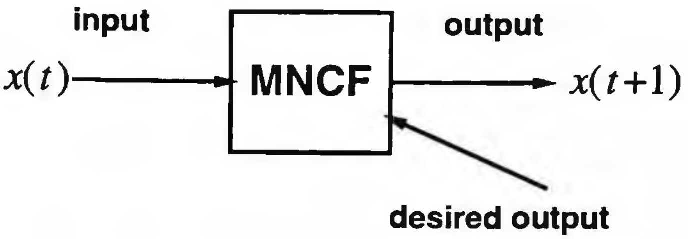


Figure 5. Memorization.

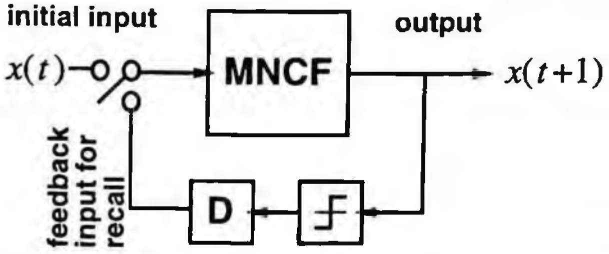


Figure 6. Recall.

3 MUltilayer Network for Sequential Inputs Using Complex Neurons (MUSIC)

In this section, we apply MNCF to memorize melodies and demonstrate the effectiveness and practicability. It is named MUSIC (MUltilayer network for Sequential Inputs using Complex neurons).

3.1 Coding

The notes and rests are coded into 9-bit bipolar patterns. 7 bits are used for ‘a’ to ‘g’ (A to G) and 1 bit is used for ‘r’ (rest). The remaining 1 bit ‘-’ means the continuance of the previous note. Each code does not include information on the length of the note.

Table 3. 7 melodies from Bartók's *Mikrokosmos*.

#1	cde-fedrefgfedc-
#2	cdedefg-fedcd-e-fgfedcd-efedc---
#3	edcdcba-bcded-c-babcded-cbaba---
#4	a-g-fefga---g-f-gfede---f-g-g-e-fga-g---feded---
#5	bcd-cde-dcdef-e-fed-e-dcbcd-c---
#6	abc-d-cba---bcd-c-bab---cbaga-b-cdcba-g-abc-b-aga---
#7	gab-cba-babcd---cdc-bab-rabcb-a-g---

3.2 Memorization

Figure 5 shows how to memorize the melodies. The recurrent networks are learned by giving the next note as the desired output.

3.3 Recall

Figure 6 shows how to recall the melodies. The network has been learned so that it can predict the next note. After some notes are presented to the network, the output of the network can be considered as the next input. Therefore, the output can used for the input. As a result, the network can recall the rest of the sequence.

3.4 Computer Simulation Results

In this section, computer simulation results are shown. We used 7 melodies from *Mikrokosmos Vol. 1* composed by Béla Bartók. Table 3 shows the coded melodies. To show the superior ability of MNCF, we compared it with the Elman's network. The numbers of neurons used for the simulation are 9-30-30-9 in the MNCF and 9+60(context)-60-9 in the Elman's Network. The learning coefficient is $\eta = 0.01$ and the time length to be considered for learning is $T = 2$ in both networks.

Figure 7 shows the learning curves of them. It can be observed from this figure that the learning of the MNCF is much better than that of

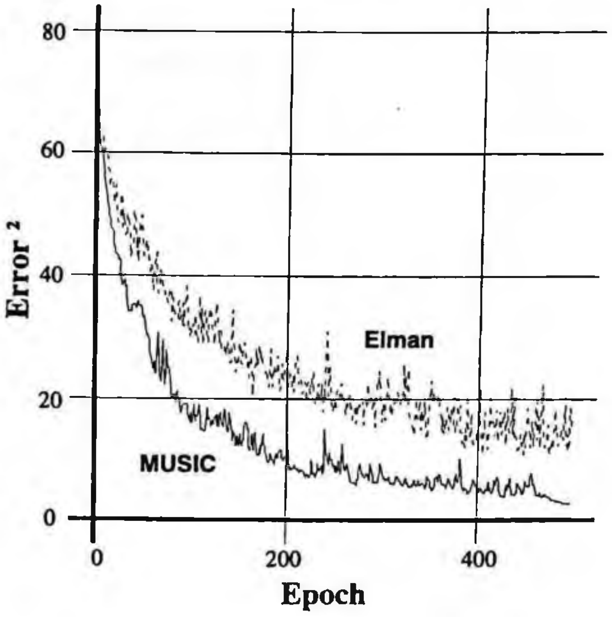


Figure 7. Learning curves.

the Elman's.

Figure 8 shows the numbers of the incorrect outputs when the whole notes of the melodies are given after learning. It can be observed that the MNCF can recall almost all of the correct notes. The Elman's Network recalls much more incorrect notes because the residual error is not small enough (see Figure 7). In the case of the MNCF, there are some (only 3) incorrect outputs produced, however, they cannot be recalled theoretically because some melodies are exactly common at the beginning.

Table 4 shows some samples of the recall of the melodies from some parts. It can be observed that the MNCF can recall the melodies perfectly and that the Elman's network cannot recall the melodies correctly because of the incorrect outputs.

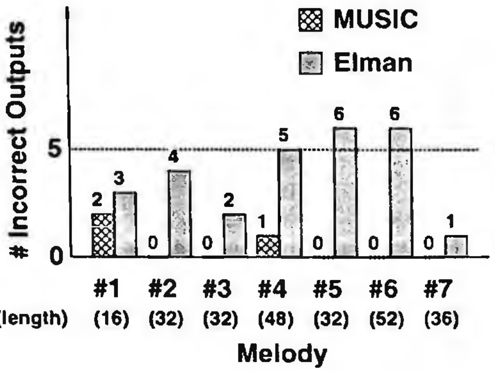


Figure 8. Number of incorrect notes.

Table 4. Presented and recalled notes. Incorrect notes are underlined. 'M' and 'E' represent MNCF and the Elman's Network, respectively.

presented	recalled
#2 cded	M efg-fedcd-e-fgfedcd-efedc---
	E efg-fedcd-e-fgfedc---f-g-f-g
#3 edcd	M cba-bcded-c-babcded-cbaba---
	E cba-bcded-c-bab---cbaga-b-cd
#4 a-g-	M fefga---g-f-gfede---f-g-g-e-fga-g---feded---
	E fefga---g-f-gfed--b-aga----bcd---cdc-bab-rab
#5 bcd-	M cde-dcdef-e-fed-e-dcbcd-c---
	E cde-f-gc-b-cdcba-g---fbacgg-
#6 abc-	M d-cba---bcd-c-bab---cbaga-b-cdcba-g-abc-b-aga---
	E d-cbaga-b-cbaga-b-cdcba-gabcd---cdc-bab-a-g---fe
#7 gab-	M cba-babcd---cdc-bab-rabcb-a-g---
	E cba-babcd---cdc-bab-gcde---cbaga

It has been confirmed that the MNCF can recall the 7 melodies separately in one network when some notes are given. That is not the same by the Elman's network.

4 Melody Retrieval System Using MUSIC

When MUSIC is applied to actual melody retrieval system, the number of melodies memorized in MUSIC must be increased. Generally, a memory capacity of neural network depends on the number of neurons. The method for enlarging MUSIC itself is considered. However, there are the following problems.

1. Since the number of neurons increases, the calculation time per input becomes enormous.
2. Since the number of memorizing patterns increases, the calculation time per epoch becomes enormous.
3. Since the leaning of MUSIC is pattern by pattern, it becomes unstable easily. The learning coefficient must be small and therefore the learning speed slows down.
4. When the correlation between memorized patterns is large, learning seldom completely finishes.

As mentioned above, there will be a limit in order to memorize many melodies even if a large-scale MUSIC is used. In order to handle the large-scale problem, we use small-scale and multiple MUSICs. As the technique that divides and integrates small-scale and multiple neural networks, Comb NET and the improved CombNET-II (Iwata *et al.* 1991) are proposed. These have succeeded in handling the large and static pattern.

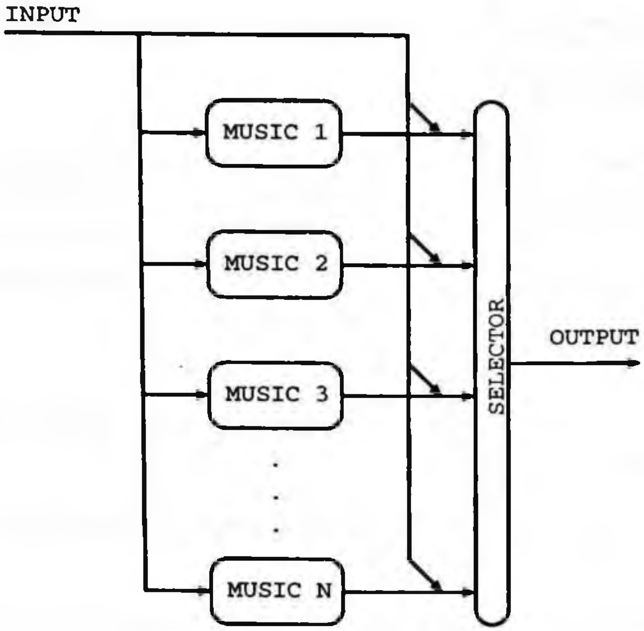


Figure 9. Melody retrieval system using multiple MUSICs.

4.1 Melody Retrieval System Using Multiple MUSICs

Figure 9 shows the melody retrieval system using multiple MUSICs. The melodies are divided into N groups, and N MUSICs are prepared. The number of melodies in a group is decided so that the irrational may not exist in the learning of MUSIC. Each group is memorized only by one MUSIC.

In a recall process, a key input is given for all MUSICs. Each MUSIC starts to output the notes in proportion to the inputs. When they are part of a melody that is memorized, the MUSIC may gradually output the note equal to the next input. In the meantime, the random note is output, when it is not memorized. So, when the input-output

Original Melody (Zousan to Korisu)

```
      g-edc-ccf-
      e-ddd-ddd-ggg-eeedccc-d-g-cde-d-g-cde-ddd-ddd-
                                               dgggccc-
      e-ddd-ddd-ggg-eee<-- presented as a key
#1     +*+*****+***+***
#2     ++**********+***
#3     *******+*+******
#4     *++*+**+++++++++dccc-d-g-cde-d-g-cde-ddd-ddd-
#5     *********++****+                        dgggccc-
#6     +**+***+********
#7     +***********+**+     output without key input
#8     +*************+*          (Recall Mode)
#9     ********+**++***
#10    ++****+*+**+****

+: MUSIC outputs the correct note.
*: MUSIC outputs the incorrect note.
```

Figure 10. Example 1 of the melody retrieval.

equalization in a MUSIC is observed, the MUSIC can be chosen. By switching the chosen MUSIC to the self recall mode, the melody after the key input is output, and it becomes the retrieval result. The above aspect is shown by two examples.

In figure 10, a part of a melody is input as a key. Within 10 MUSICs, this melody is memorized only in #4. Each MUSIC carries out an output one by one, when the key input is given. In figure 10, + represents that the output is equal to the next input (correct), and * represents that the output is not equal to the next input (incorrect). It can be observed that correct notes are not output continuously before 7th note of the key input. After 8th note of the key input, MUSIC #4 outputs the correct notes continuously, although the other MUSICs output the incorrect notes. Therefore, it is shown that this melody is memorized in MUSIC #4. After the end of the key input, in making chosen MUSIC in the recall mode, the continuance of the melody is output. Here, it is shown that the melody has been reproduced cor-

Original Melody (Haru no Ogawa)

```
egagegccaagecdere
gagegccaagedecrdedgaagaccbaggeregagegccaagedecr

gagegccaagedecr←— presented as a key
#1    **************
#2    *+***+********
#3    *+++*+*++***+*
#4    *****+*******+
#5    *+*****+****+*
#6    *****+*++**++*
#7    *+++*+++++++++dedgaagaccbaggeregagegccaagedecr
#8    *++*****++*++*                output without key input
#9    *+****+****+**                      (Recall Mode)
#10   *+***+********

+: MUSIC outputs the correct note.
*: MUSIC outputs the incorrect note.
```

Figure 11. Example 2 of the melody retrieval.

rectly by MUSIC #4.

In figure 11, a part of another melody is input as a key. This melody is memorized in MUSIC #7. On the beginning of the input, MUSIC #3 and #7 output 3 correct notes continuously. However, MUSIC #3 outputs incorrect notes after that, although MUSIC #7 outputs the correct notes. So, MUSIC #7 is chosen. As in Figure 10, after the end of the key input, in making chosen MUSIC in the recall mode, the continuance of the melody is output correctly.

4.2 Computer Simulation Results

We chose 79 melodies for Japanese children. All melodies contain no modulation. Each melody consists of 32-192 notes. There are 5862 notes in total. We divided 79 melodies into 10 groups. Ten MUSICs were learned to memorize 7-8 melodies. The number of neurons in MUSICs are 9-30-30-9.

Table 5. The number of incorrect notes.

# incorrect outputs	0	1	2	3	4-
# melodies	24	34	14	4	3

The learning coefficients are $\eta = 0.01$, and the time lengths to be considered for learning are $T = 3$. Because the internal state of the MUSIC is reset before each melody is input, the beginning of the melody is learned as there is no note before it.

The learning situation was checked at every 200 epochs. The error of every MUSIC did not decrease after 600 epochs. MUSICs were learned 1000 epochs. Table 5 shows the number of the incorrect notes when the correct notes of the whole melody are presented. The number of the memorized notes in 10 MUSICs was 5862 and the number of the incorrect notes was 101.

These errors occurred by the following reasons:

1. There is the theoretically inevitable error since the beginnings of some melodies are exactly the same.
2. In order to treat refrains, some mechanism is required.

On the recall, when 8 outputs (considered as a phrase) of a MUSIC were continuously the same as the next inputs, the MUSIC was chosen as it memorizes the presented melody. In this condition, the system was investigated by many parts of the 79 melodies.

In 60 melodies, only a MUSIC outputs the correct notes. When the feedback of the outputs were presented as the input, the MUSIC was able to recall the whole melody.

In the rest 19 melodies, only one MUSIC was responded. However, after the feedback was started, the recall of MUSIC was looped or shorten by the omission of the refrain.

The defect of the system is the vulnerability by refrain in a melody. When there is a long refrain part in a melody, the system is not able to know the number of the refrains. However, when human recalls a melody, the same error often might occur.

5 Conclusions

We have explained a Multilayer Network using Complex neurons with local Feedback (MNCF). A complex neuron can keep previous information by the phase component. We have derived a simple learning algorithm based on the back-propagation to deal with temporal sequences. It has shown in computer simulations that the network has much better ability than the conventional ones, including Elman's network.

We have applied MNCF to memorize some melodies. It is called MUSIC (MUltilayer network for Sequential Inputs using Complex neurons). It can be considered as an associative memory for the temporal sequences. It has shown by computer simulations that the network can memorize plural melodies and recall them correctly from any part.

A melody retrieval system using multiple MUSICs has been constructed. It can treat a number of melodies by some smaller networks. Such architecture has an advantage that the pattern matching process is not required. The system uses a part of the melodies as a key instead of the text information.

Melody is an example of temporal sequences. Memorization of temporal sequences can be used in a lot of fields e.g. control, information processing, thinking support systems, and so on.

References

Day, S.T. and Davenport, M.R. (1993), "Continuous-time temporal back-propagation with adaptive time delays," *IEEE Trans. Neural Networks*, vol. 4, pp. 348-354.

Elman, J.L. (1990), "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179-211.

Frasconi, P., Gori, M., and Sada, G. (1992), "Local feedback multi-layered networks," *Neural Computation*, vol. 4, pp. 120-130.

Iwata, A., Hotta, K., Matuo, H., and Suzumura, N. (1991), "A Large Scale Neural Network "CombNET-II"," *Proc. Int. Joint Conf. on Neural Networks*, Seattle, USA, II A-932.

Jordan, M.I. (1986), "Serial order: a parallel distributed processing approach," *ICS Report*, 8604, UC San Diego.

Kataoka, M., Kinouchi, M., and Hagiwara, M. (1998), "Music information retrieval system using complex-valued recurrent neural networks," *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, San Diego, California, USA, pp. 4290-4295.

Kinouchi, M. and Hagiwara, M. (1995), "Learning temporal sequences using complex neurons with local feedback," *Proc. IEEE Int. Conf. on Neural Networks*, Perth, Australia, pp. 3165-3169.

Kinouchi, M. and Hagiwara, M. (1996), "Memorization of melodies by complex-valued recurrent network," *Proc. IEEE Int. Conf. on Neural Networks*, Washington, D.C., USA, pp. 1324-1328.

Nishi, M., Furuya, J. and Nakamura, T. (1993), "Backpropagation networks including time delay elements (BPD)," *Proc. Int. Joint Conf. Neural Networks*, pp. 1681-1684.

Parlos, A., Chong, K.T., and Atiya, A.F. (1994), "Application of the recurrent multilayer percceptron in modeling complex process dynamics," *IEEE Trans. Neural Networks*, vol. 5, pp. 255-266.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986), "Learning internal representation by error propagation," *Parallel Distributed Processing*, vol. 1, MIT Press, pp. 318-362.

Tabuse, M., Kinouchi, M., and Hagiwara, M. (1997), "Recurrent neural network using mixture of experts for time series processing," *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics*, Orlando, Florida, USA, pp. 536-541.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K.J. (1989), "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech & Signal Process.*, vol. 37, pp. 328-339.

Authors' address

Makoto Kinouchi: Department of Bio-System Engineering, Faculty of Engineering, Yamagata University, 4-3-16 Jounan, Yonezawa, Yamagata 992-8510, Japan.
Masafumi Hagiwara: Department of Information and Computer Science, Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi, Yokohama, Kanagawa 223-8522, Japan.

Chapter 11

Complex-Valued Generalized Hebbian Algorithm and Its Applications to Sensor Array Signal Processing

Yanwu Zhang

Principal component extraction is an efficient statistical tool that is applied to feature extraction, data compression, and signal processing. The Generalized Hebbian Algorithm (GHA) (Sanger 1992) can be used to iteratively extract principal eigenvectors in the real domain. In some scenarios such as sensor array signal processing, we encounter complex data. The Complex-valued Generalized Hebbian Algorithm (CGHA) (Zhang *et al.* 1997) is presented in this chapter. Convergence of CGHA is proved. Like GHA, CGHA can be implemented by a single-layer linear neural network. An application of CGHA to sensor array signal processing is demonstrated through Direction of Arrival (DOA) estimation.

1 Review of Principal Component Extraction and the Generalized Hebbian Algorithm (GHA)

Consider the autocorrelation matrix of an N-dimensional random column vector X: $R_{XX} = E[XX^H]$ where H stands for conjugate transpose. R_{XX} can be expressed as (Strang 1993):

$$R_{XX} = \sum_{i=1}^{N} \lambda_i U_i U_i^H \tag{1}$$

where U_i is the ith eigenvector (column vector) and λ_i is the corresponding eigenvalue. Here the eigenvectors are normalized and orthogonalized. In the real domain, conjugate transpose H reduces to transpose T.

If we sort the eigenvectors by their associated eigenvalues in descending order, the leading eigenvectors are called the principal eigenvectors because they span the major portion of R_{XX}.

Due to their statistical significance, principal eigenvectors find applications in various realms. Their usefulness to sensor array signal processing will be demonstrated in Section 3.

Data compression also relies on principal eigenvectors. The technique of principal component extraction (also called principal component analysis) (Sanger 1992), (Oja 1992), (Haykin 1994) linearly reduces the dimensionality of input data while retaining major statistical information (Bannour *et al.* 1995), (Plumbley 1995), shown as follows.

Consider an N-dimensional zero-mean random vector

$$X = [x_1 \quad x_2 \quad \cdots \quad x_N]^T \tag{2}$$

We desire to reduce its dimension from N to M ($M < N$). First, we find the M principal eigenvectors of the input's autocorrelation matrix R_{XX}, denoted as $U_1, U_2, \cdots, U_M$ (orthonormalized and arranged in descending order of their associated eigenvalues). These column vectors constitute an $N \times M$ mapping matrix

$$Q = [U_1 \ \vdots \ U_2 \ \vdots \ \cdots \ \vdots \ U_M] \tag{3}$$

Then we map the N-dimensional input vector X to an M-dimensional output vector Y through Q:

$$Y = Q^H X \tag{4}$$

Elements of vector Y are called principal components.

Since $M < N$, dimensionality of the input vector space is reduced. Data are thus compressed. Compression generally induces information loss, but it can be proven (Hornik *et al.* 1992) that the linear transform in Equation (4) is optimal in the sense that it minimizes the mean squared error when reconstructing X:

$$\hat{X} = QY \tag{5}$$

Sanger presented the Generalized Hebbian Algorithm (GHA) (Sanger 1992) to iteratively derive principal eigenvectors using a single-layer linear neural network. The algorithm can be summarized as follows.

The $N \times 1$ input vector X is expressed in Equation (2). The eigenvectors of R_{XX} are denoted as $U_1, U_2, \cdots, U_N$ (arranged in descending eigenvalue order). We randomly initialize $N \times 1$ vectors $V_1, V_2, \cdots, V_N$. GHA then updates V_j iteratively. The updating rule for V_j at iteration step n is given by Equation (6) and Equation (7):

$$V_j(n+1) = V_j(n) + \mu(n) y_j(n)[X(n) - y_j(n)V_j(n) - \sum_{i<j} y_i(n)V_i(n)] \tag{6}$$

$$y_j(n) = V_j^T(n)X(n) \tag{7}$$

where $\mu(n)$ is a learning rate factor. Sanger proved that V_j converges to U_j (Sanger 1992).

GHA possesses the following features: snapshot-based processing instead of eigendecomposition, parallel processing, and good expandability. Hence this algorithm facilitates fast and distributed processing. It has been applied to image coding and texture segmentation (Sanger 1992).

2 Complex-Valued Generalized Hebbian Algorithm (CGHA)

GHA is applicable only in the real domain. In some scenarios, we encounter complex data. For example, in sensor array signal processing, typically the received real signal at each sensor is quadrature demodulated to a complex signal (Van Trees 2002). We are interested in the principal eigenvectors of the array's autocorrelation matrix because they contain key information about the signals' incoming directions. Therefore, an extension of GHA to the complex domain is needed.

2.1 Formulation of CGHA

Now we present the Complex-valued Generalized Hebbian Algorithm (CGHA) (Zhang *et al.* 1997). Randomly initialize $N \times 1$ vectors $V_1, V_2, \cdots, V_N$. CGHA then updates V_j iteratively. The updating rule for V_j at iteration step n is given by Equation (8) and Equation (9):

$$V_j(n+1) = V_j(n) + \mu(n) y_j^*(n)[X(n) - y_j(n)V_j(n) - \sum_{i<j} y_i(n)V_i(n)] \tag{8}$$

$$y_j(n) = V_j^H(n)X(n) \tag{9}$$

where $y_j^*(n)$ denotes the complex conjugate of $y_j(n)$, and $\mu(n)$ is a learning rate factor. $V_j(n)$ will converge to the jth normalized eigenvector of R_{XX}. Proof is given in the next subsection.

As shown in Figure 1, each block represents an N-dimensional weight vector V_j $(j = 1, 2, \cdots, N)$. The output of the jth block is $y_j = V_j^H X$. As the input vector X flows through each block, $y_j V_j$ is subtracted from X to form the updating vector that is contained in the bracket in Equation (8).

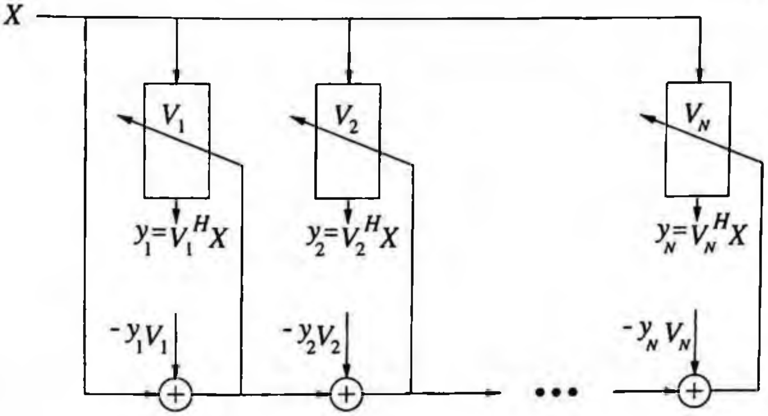


Figure 1. Implementation of CGHA.

2.2 Convergence of CGHA

Convergence analysis of CGHA extends that of GHA (Sanger 1992) to the complex domain. Let us rewrite Equation (8) in matrix form to include all eigenvectors:

$$\begin{aligned} W(n+1) &= W(n) + \mu(n)\Big\{X(n)X^H(n)W(n) \\ &\quad -W(n)\{UT[Y(n)Y^H(n)]\}\Big\} \end{aligned} \tag{10}$$

where $W(n)$ is an $N \times N$ matrix:
$W(n) = [V_1(n) \;\vdots\; V_2(n) \;\vdots\; \cdots \;\vdots\; V_N(n)]$, and

$$Y(n) = W^H(n)X(n) \tag{11}$$

Operator $UT[\cdot]$ (standing for upper triangle) sets all elements below the diagonal of the square matrix to zero, thereby producing an upper triangular matrix.

Assume temporal stationarity. Then $R_{XX} = E[X(n)X^H(n)]$ does not vary with n. Taking expectation on both sides of Equation (10) and incorporating Equation (11), we have

$$\begin{aligned} W(n+1) &= W(n) + \mu(n)\Big\{R_{XX}W(n) \\ &\quad -W(n)\{UT[W^H(n)R_{XX}W(n)]\}\Big\} \end{aligned} \tag{12}$$

The convergence property of the above discrete-time difference equation is the same as that of the following continuous-time differential equation:

$$\frac{d}{dt}W(t) = R_{XX}W(t) - W(t)\{UT[W^H(t)R_{XX}W(t)]\} \tag{13}$$

We prove the convergence in two steps:

§1 Prove that $V_1(t)$ converges to the eigenvector associated with the largest eigenvalue.

V_1 is the first column of matrix W. According to Equation (13), variation of $V_1(t)$ is governed by

$$\frac{d}{dt}V_1(t) = R_{XX}V_1(t) - V_1(t)[V_1^H(t)R_{XX}V_1(t)] \tag{14}$$

Assume R_{XX} is positive definite with N distinct eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_N$ which correspond to orthonormalized eigenvectors $E_1, E_2, \cdots, E_N$. Note that since R_{XX} is Hermitian, all of its eigenvalues are real.

Expand $V_1(t)$ as

$$V_1(t) = \sum_{k=1}^{N} c_k(t)E_k \tag{15}$$

E_k $(k = 1, 2, \cdots, N)$ is an orthonormal base. Premultiply E_k^H to both sides of Equation (15) and we have

$$c_k(t) = E_k^H V_1(t) \tag{16}$$

Plugging Equation (16) together with $R_{XX}E_k = \lambda_k E_k$ into Equation (14) gives

$$\sum_{k=1}^{N} \frac{dc_k(t)}{dt} E_k = \sum_{k=1}^{N} c_k(t)\lambda_k E_k - [\sum_{l=1}^{N} |c_l(t)|^2\lambda_l] \sum_{k=1}^{N} c_k(t)E_k \tag{17}$$

where $|\cdot|$ denotes norm of a complex variable.

Premultiply E_k^H to both sides of Equation (17). The orthonormality of base E_k leads to

$$\frac{dc_k(t)}{dt} = c_k(t)[\lambda_k - \sum_{l=1}^{N} |c_l(t)|^2\lambda_l] \tag{18}$$

We now study the convergence of $c_k(t)$ in two cases: when $k > 1$ and when $k = 1$.

(a) When $k > 1$.

Define $r_k(t) = \frac{c_k(t)}{c_1(t)}$, assuming $c_1(t) \neq 0$. Take differentiation of $r_k(t)$ with respect to t:

$$\frac{dr_k(t)}{dt} = \frac{1}{c_1(t)}[\frac{dc_k(t)}{dt} - r_k(t)\frac{dc_1(t)}{dt}] \tag{19}$$

Plugging Equation (18) into Equation (19), we have

$$\begin{aligned}\frac{dr_k(t)}{dt} &= \frac{1}{c_1(t)}\Big\{c_k(t)[\lambda_k - \sum_{l=1}^{N}|c_l(t)|^2\lambda_l] \\ &\quad -r_k(t)c_1(t)[\lambda_1 - \sum_{l=1}^{N}|c_l(t)|^2\lambda_l]\Big\}\end{aligned} \tag{20}$$

which can be simplified to

$$\frac{dr_k(t)}{dt} = r_k(t)(\lambda_k - \lambda_1) \tag{21}$$

The solution to the above differential equation is

$$r_k(t) = r_k(0)\, e^{(\lambda_k - \lambda_1)t} \tag{22}$$

We know that $\lambda_k < \lambda_1$ for $k > 1$. Therefore, with any initial value, $r_k(t)$ exponentially decays to 0 when $k > 1$.

(b) When $k = 1$.

According to Equation (18) and definition $r_k(t) = \frac{c_k(t)}{c_1(t)}$, we have

$$\frac{dc_1(t)}{dt} = c_1(t)[\lambda_1 - |c_1(t)|^2\lambda_1 - |c_1(t)|^2\sum_{l=2}^{N}|r_l(t)|^2\lambda_l] \tag{23}$$

We have shown that $r_k(t)$ exponentially decays to 0 when $k > 1$. So at a large t, $|c_1(t)|^2\lambda_1$ dominates over $|c_1(t)|^2 \sum_{l=2}^{N} |r_l(t)|^2 \lambda_l$. For convergence analysis, we can thus drop the last term. Then Equation (23) reduces to

$$\frac{dc_1(t)}{dt} = c_1(t)[\lambda_1 - |c_1(t)|^2\lambda_1] \tag{24}$$

Let us define another function

$$F(t) = [|c_1(t)|^2 - 1]^2 \tag{25}$$

Utilizing Equation (24), we have

$$\frac{dF(t)}{dt} = -4\lambda_1 |c_1(t)|^2 [|c_1(t)|^2 - 1]^2 \tag{26}$$

Equation (25) gives that $F(t) \geq 0$, and Equation (26) shows that $\frac{dF(t)}{dt} \leq 0$. Therefore $F(t)$ must converge to 0. Equivalently, $|c_1(t)|$ converges to 1, according to Equation (25).

By Equation (15) and definition $r_k(t) = \frac{c_k(t)}{c_1(t)}$, we have $V_1(t) = c_1(t)E_1 + c_1(t)\sum_{k=2}^{N} r_k(t)E_k$. In 1a, it is shown that $r_k(t)$ converges to 0 when $k > 1$. In 1b it is shown that $|c_1(t)|$ converges to 1. Hence $V_1(t)$ converges to E_1 with a complex factor of norm one.

§2 Prove that for $j > 1$, $V_j(t)$ converges to the eigenvector associated with the jth largest eigenvalue.

We resort to the method of induction. Given that $V_1(t) \rightarrow E_1$, we only need to show that if $V_k(t)$ converges to E_k for $k = 1, 2, \cdots, j-1$, $V_j(t)$ converges to E_j.

According to Equation (13), variation of $V_j(t)$ is governed by

$$\frac{d}{dt}V_j(t) = R_{XX}V_j(t) - \sum_{k \leq j} V_k(t)[V_k^H(t)R_{XX}V_j(t)] \tag{27}$$

$V_k(t)$ can be expressed as

$$V_k(t) = E_k + \epsilon_k(t) G_k(t) \tag{28}$$

where $G_k(t)$ is a unit-length vector and $\epsilon_k(t)$ is a scalar.

The premise of the induction is that $V_k(t)$ converges to E_k for $k < j$, so $\epsilon_k(t)$ converges to 0 for $k < j$. Combining Equation (28) and Equation (27), we have

$$\begin{aligned} \frac{d}{dt} V_j(t) &= R_{XX} V_j(t) - V_j(t)[V_j^H(t) R_{XX} V_j(t)] \\ &\quad - \sum_{k<j} E_k [E_k^H(t) R_{XX} V_j(t)] \\ &\quad + O(\epsilon) + O(|\epsilon|^2) \end{aligned} \tag{29}$$

where $O(\epsilon)$ represents a term converging to 0 at least as fast as the slowest vanishing $\epsilon_k(t)$ for $k < j$. $O(|\epsilon|^2)$ has a similar meaning. At a large t, we neglect terms $O(\epsilon)$ and $O(|\epsilon|^2)$.

Expand $V_j(t)$ as

$$V_j(t) = \sum_{k=1}^{N} b_k(t) E_k \tag{30}$$

where $b_k(t) = E_k^H V_j(t)$

Plugging Equation (30) together with $R_{XX} E_k = \lambda_k E_k$ into Equation (29) (neglecting vanishing terms), we have

$$\begin{aligned} \sum_{k=1}^{N} \frac{db_k(t)}{dt} E_k &= -\sum_{k<j} [\sum_{l=1}^{N} |b_l(t)|^2 \lambda_l] b_k(t) E_k \\ &\quad + \sum_{k=j}^{N} [\lambda_k - \sum_{l=1}^{N} |b_l(t)|^2 \lambda_l] b_k(t) E_k \end{aligned} \tag{31}$$

Premultiplying E_k^H to both sides of Equation (31), and utilizing the orthonormality of E_k, we have

$$\frac{db_k(t)}{dt} = -b_k(t)\sum_{l=1}^{N}|b_l(t)|^2\lambda_l \qquad \text{for } k < j \tag{32}$$

$$\frac{db_k(t)}{dt} = b_k(t)[\lambda_k - \sum_{l=1}^{N}|b_l(t)|^2\lambda_l] \qquad \text{for } k \geq j \tag{33}$$

(a) For $k < j$, the solution to differential equation (32) is

$$b_k(t) = b_k(0)e^{-[\sum_{l=1}^{N}|b_l(t)|^2\lambda_l]t} \tag{34}$$

R_{XX} is positive definite, so $\lambda_l > 0$. Hence $-[\sum_{l=1}^{N}|b_l(t)|^2\lambda_l] < 0$. Consequently, $b_k(t)$ exponentially decays to 0 for $k < j$.

(b) For $k > j$, define $s_k(t) = \frac{b_k(t)}{b_j(t)}$, assuming $b_j(t) \neq 0$. Then Equation (33) leads to

$$\begin{aligned}\frac{ds_k(t)}{dt} &= \frac{1}{b_j(t)}\{b_k(t)[\lambda_k - \sum_{l=1}^{N}|b_l(t)|^2\lambda_l] \\ &\quad -s_k(t)b_j(t)[\lambda_j - \sum_{l=1}^{N}|b_l(t)|^2\lambda_l]\}\end{aligned} \tag{35}$$

which can be simplified to

$$\frac{ds_k(t)}{dt} = s_k(t)(\lambda_k - \lambda_j) \tag{36}$$

The solution to the above differential equation is

$$s_k(t) = s_k(0)e^{(\lambda_k - \lambda_j)t} \tag{37}$$

Since $\lambda_k < \lambda_j$ for any $k > j$, $s_k(t)$ exponentially decays to 0 for $k > j$.

(c) For $k = j$, Equation (33) becomes

$$\begin{aligned}\frac{db_j(t)}{dt} &= b_j(t)[\lambda_j - |b_j(t)|^2\lambda_j \\ &\quad -|b_j(t)|^2\sum_{l>j}|s_l(t)|^2\lambda_l - \sum_{l<j}|b_l(t)|^2\lambda_l] \qquad (38)\end{aligned}$$

It has been shown in 2a that $b_l(t) \rightarrow 0$ for $l < j$, and in 2b that $s_l(t) \rightarrow 0$ for $l > j$. We thus drop the last two terms in Equation (38) for a large t, and the equation reduces to

$$\frac{db_j(t)}{dt} = b_j(t)[\lambda_j - |b_j(t)|^2\lambda_j] \qquad (39)$$

To show that $b_j(t)$ converges, we define another function

$$H(t) = [|b_j(t)|^2 - 1]^2 \qquad (40)$$

Using Equation (39), we have

$$\frac{dH(t)}{dt} = -4\lambda_j|b_j(t)|^2[|b_j(t)|^2 - 1]^2 \qquad (41)$$

Equation (40) gives that $H(t) \geq 0$, and Equation (41) shows that $\frac{dH(t)}{dt} \leq 0$. Therefore $H(t)$ must converge to 0. Equivalently, $|b_j(t)|$ converges to 1, according to Equation (40).

We know the expansion $V_j(t) = b_j(t)E_j + \sum_{k<j} b_k(t)E_k + \sum_{k>j} b_k(t)E_k$. It has been shown that $b_k(t) \rightarrow 0$ for $k < j$, $b_k(t) \rightarrow 0$ for $k > j$ (because $s_k(t) \rightarrow 0$ for $k > j$), and $|b_j(t)| \rightarrow 1$. Therefore $V_j(t)$ converges to E_j with a complex factor of norm one.

Analyses in §1 and §2 establish convergence of CGHA: $V_j(n)$ converges to E_j for $j = 1, 2, \cdots, N$.

2.3 Implementation of CGHA

As illustrated in Figure 1, CGHA can be implemented by a single-layer linear neural network. The slanted arrows in the figure represent updating of V_j, $j = 1, 2, \cdots, N$.

Like GHA, CGHA possesses the following features:

§1 There is no need to estimate the autocorrelation matrix R_{XX}. Its eigenvectors are derived directly from the input vector. In sensor array signal processing, the input vector is just a snapshot of received signals at all sensors at one sampling instant.

§2 The implementation architecture has good expandability. Updating of V_j is affected by V_k of $k < j$, but not by V_k of $k > j$. If convergence has been reached for the first M eigenvectors, the additional learning of the $(M + 1)$th eigenvector will leave intact the preceding M eigenvectors.

§3 The algorithm can be carried out by parallel processing. Equation (8) can be rewritten as

$$V_j(n+1) = V_j(n) + \mu(n) y_j^*(n)[X_j(n) - y_j(n)V_j(n)] \qquad (42)$$

where

$$X_j(n) = X(n) - \sum_{i<j} y_i(n) V_i(n) \qquad (43)$$

can be deemed the "net" input for updating V_j.

Equation (42) gives a uniform rule for updating V_j. Thus CGHA can be carried out by multiple processors in parallel.

3 Application of CGHA to Sensor Array Signal Processing

3.1 Transformation of Real Signal to Complex Signal by Quadrature Demodulation

Consider a linear array composed of N equally spaced sensors with identical directivity. Suppose D narrowband signals impinge on the array as plane waves from directions $\theta_1, \theta_2, \cdots, \theta_D$, as illustrated in Figure 2. Assume the received noise is spatially white with zero mean, and is uncorrelated with the signals.

Suppose the narrow-band signals have a carrier frequency f_0. At sensor No. N, the received narrow-band signal of incident angle θ_k can be expressed as

$$s_{kN}(t) = f_k(t) cos[2\pi f_0 t + \phi_k(t)] \tag{44}$$

where amplitude $f_k(t)$ and phase $\phi_k(t)$ have narrow bands that satisfy (Van Trees 2002)

$$B_{f_k} \, \Delta T_k \ll 1 \tag{45}$$

$$B_{\phi_k} \, \Delta T_k \ll 1 \tag{46}$$

where B_{f_k} and B_{ϕ_k} are the bandwidth of $f_k(t)$ and $\phi_k(t)$, respectively. ΔT_k is the plane wave's travel time from sensor No. 1 to No. N. Hence variations of $f_k(t)$ and $\phi_k(t)$ over ΔT_k can be deemed negligible.

Relative to the signal received by sensor No. N, the received signal at sensor No. m $(m = 1, 2, \cdots, N-1)$ has a time delay:

$$s_{km}(t) = f_k(t - \tau_{km})\, cos[2\pi f_0(t - \tau_{km}) + \phi_k(t - \tau_{km})] \qquad (47)$$

where

$$\tau_{km} = \frac{[(N - m)d]sin(\theta_k)}{c} \qquad (48)$$

is the delay; d is the spacing between adjacent sensors, and c is the propagation speed of the plane waves.

Under the narrow-band conditions in Equation (45) and Equation (46), we have

$$f_k(t - \tau_{km}) \approx f_k(t) \qquad (49)$$
$$\phi_k(t - \tau_{km}) \approx \phi_k(t) \qquad (50)$$

Then Equation (47) is simplified to

$$s_{km}(t) = f_k(t)\, cos[2\pi f_0(t - \tau_{km}) + \phi_k(t)] \qquad (51)$$

The received signal is carried at a center frequency f_0. To lower the sampling frequency and hence reduce the system's cost, the signal is typically quadrature demodulated (Van Trees 2002) prior to analog-to-digital conversion. Quadrature demodulation is illustrated in Figure 2. Two branches of the received signal are multiplied by $2cos(2\pi f_0 t)$ and $-2sin(2\pi f_0 t)$, respectively, and then low-passed to keep only the base-band components. The two branches then add up to a complex base-band signal. After quadrature demodulation, real signal $s_{km}(t)$ in Equation (51) is transformed to a complex signal

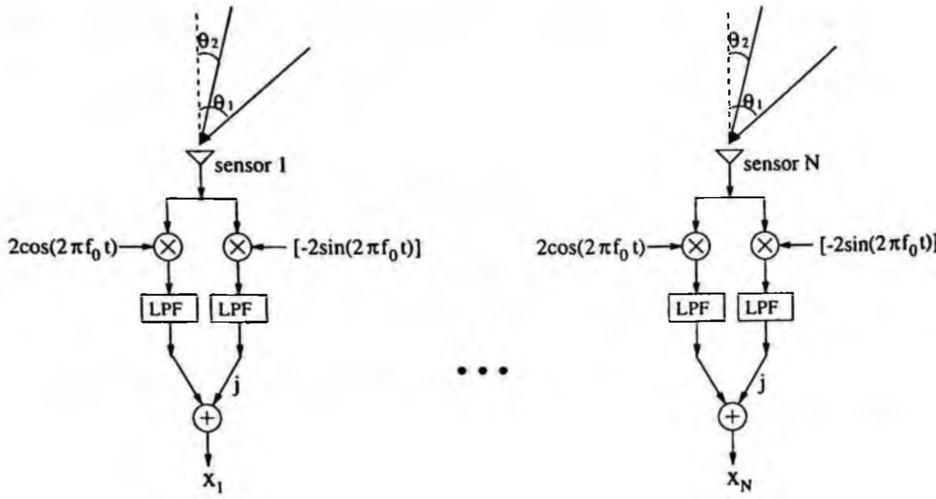


Figure 2. Quadrature demodulation of sensor array signals. At each sensor's output, the imaginary part (the right branch) is multiplied by j prior to summation.

$$\begin{aligned}\tilde{s}_{km}(t) &= f_k(t)\, cos[\phi_k(t) - 2\pi f_0 \tau_{km}] \\ &\quad + j\Big\{ f_k(t)\, sin[\phi_k(t) - 2\pi f_0 \tau_{km}]\Big\} \\ &= f_k(t)\, e^{j\phi_k(t)}\, e^{-j2\pi f_0 \tau_{km}} \qquad (52)\end{aligned}$$

where the real part $f_k(t) cos[\phi_k(t) - 2\pi f_0 \tau_{km}]$ is called the in-phase component, and the imaginary part $f_k(t) sin[\phi_k(t) - 2\pi f_0 \tau_{km}]$ is called the quadrature component.

The original signal $s_{km}(t)$ can be restored from $\tilde{s}_{km}(t)$ by using both the in-phase and the quadrature components:

$$\begin{aligned}s_{km}(t) &= Re\{\tilde{s}_{km}(t)\, e^{j2\pi f_0 t}\} \\ &= Re\Big\{[f_k(t)\, e^{j\phi_k(t)}\, e^{-j2\pi f_0 \tau_{km}}]\, e^{j2\pi f_0 t}\Big\} \qquad (53)\end{aligned}$$

where $Re\{\cdot\}$ takes the real part of the argument.

Note that $s_{km}(t)$ cannot be restored by the in-phase component or the quadrature component alone. Hence we need to keep both components to preserve the information contained in $s_{km}(t)$. Complex signal $\tilde{s}_{km}(t)$ in Equation (52) is thus an equivalent representation of the original real signal $s_{km}(t)$.

3.2 Direction-Of-Arrival (DOA) Derived from Principal Eigenvectors of Array's Autocorrelation Matrix

Equation (52) gives the signals' phase-shift relationship between sensors. Now let us combine Equation (52) and Equation (48), and introduce notation

$$\gamma_k = \frac{f_0 d sin(\theta_k)}{c} \tag{54}$$

Then for the signal of incident angle θ_k, demodulated signals at all sensors can be expressed by an N-element column vector:

$$\begin{aligned} \tilde{S}_k(t) &= f_k(t)e^{j\phi_k(t)}[e^{-j2\pi f_0 \tau_{k1}} \; \cdots \; e^{-j2\pi f_0 \tau_{kN}}]^T \\ &= f_k(t)e^{j\phi_k(t)}[e^{-j2\pi(N-1)\gamma_k} \; \cdots \; e^{-j2\pi(N-N)\gamma_k}]^T \\ &= f_k(t)e^{j\phi_k(t)}e^{-j2\pi(N-1)\gamma_k}[1 \; \cdots \; e^{j2\pi(N-1)\gamma_k}]^T \\ &= \tilde{f}_k(t)Z_k \end{aligned} \tag{55}$$

where $\tilde{f}_k(t) = f_k(t)e^{j\phi_k(t)}e^{-j2\pi(N-1)\gamma_k}$

$$Z_k = \begin{bmatrix} 1 \\ e^{j2\pi\gamma_k} \\ \vdots \\ e^{j2\pi(N-1)\gamma_k} \end{bmatrix} \tag{56}$$

is the steering vector of the kth signal. It carries information of the signal's direction, yet is independent of t.

The sum of the D signals received by the array is

$$\begin{aligned} \tilde{S}(t) &= \sum_{k=1}^{D} \tilde{S}_k(t) \\ &= \sum_{k=1}^{D} \tilde{f}_k(t) Z_k \\ &= \left[Z_1 \vdots Z_2 \vdots \cdots \vdots Z_D \right] \begin{bmatrix} \tilde{f}_1(t) \\ \tilde{f}_2(t) \\ \vdots \\ \tilde{f}_D(t) \end{bmatrix} \end{aligned} \tag{57}$$

With addition of noise, the total output of the array is represented by an N-element column vector $X(t)$:

$$\begin{aligned} X(t) &= \tilde{S}(t) + \tilde{B}(t) \\ &= \left[Z_1 \vdots Z_2 \vdots \cdots \vdots Z_D \right] \begin{bmatrix} \tilde{f}_1(t) \\ \tilde{f}_2(t) \\ \vdots \\ \tilde{f}_D(t) \end{bmatrix} + \begin{bmatrix} \tilde{b}_1(t) \\ \tilde{b}_2(t) \\ \vdots \\ \tilde{b}_N(t) \end{bmatrix} \end{aligned} \tag{58}$$

where column vector $\tilde{B}(t) = [\tilde{b}_1(t)\ \tilde{b}_2(t)\ \cdots\ \tilde{b}_N(t)]^T$ denotes noise received by the N sensors. The noise is assumed to be spatially white with zero mean and variance σ^2, and uncorrelated with the signals.

The autocorrelation matrix of $X(t)$ is $R_{XX} = E[X(t)X^H(t)]$, an $N \times N$ matrix. Denote the eigenvectors as V_j, $j = 1, 2, \cdots, N$. Assuming the signals do not contain coherent pairs (Van Trees 2002), it can be shown (Schmidt 1986) that D eigenvectors have eigenvalues larger than σ^2. V_j ($j = 1, 2, \cdots, D$) span the signal subspace (Van Trees 2002).

Signals' directions can be estimated from these principal eigenvectors (Tufts 1998). Utilizing properties of Vandermonde vectors (the steering vectors Z_k in Equation (56) are Vandermonde vectors), Reddi proposed a method (Reddi 1979) to estimate the signals' directions based on the signal-subspace eigenvectors.

It can be shown (Cadzow 1988) that the signal-subspace eigenvectors are linear combinations of Z_k, $k = 1, 2, \cdots, D$:

$$V_j = \sum_{k=1}^{D} \alpha_{jk} Z_k \qquad j = 1, 2, \cdots, D \text{ with } \lambda_j > \sigma^2 \tag{59}$$

where α_{jk} is a coefficient.

By Equation (54) and Equation (56), Z_k can be regarded as a sinusoid of spatial frequency $\gamma_k = \frac{f_o d \sin(\theta_k)}{c}$ and of length N. Then V_j ($j = 1, 2, \cdots, D$) can be regarded as a one-dimensional sequence composed of multiple sinusoids.

Utilizing Equation (59), we look for signals' directions in three steps:

- Obtain the signal-subspace eigenvectors V_j ($j = 1, 2, \cdots, D$) by CGHA.

- Find the frequency components γ_k $(k = 1, 2, \cdots, D)$ of V_j.
- Derive θ_k by relationship $\gamma_k = \frac{f_0 d sin(\theta_k)}{c}$ (Equation (54)).

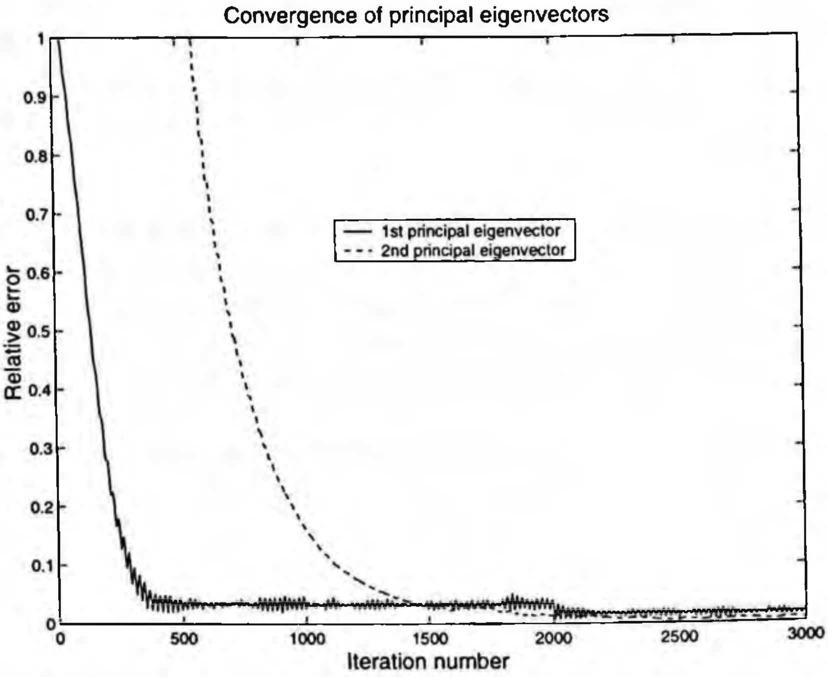


Figure 3. Learning curves of the first and the second principal eigenvectors.

Consider a 15-sensor uniform linear array. Two acoustic signals impinge on the array. The first signal is of normalized frequency $f_1 = 0.2$ (normalized by the sampling frequency) and incident angle $\theta_1 = 10°$. The second signal is of normalized frequency $f_2 = 0.15$ and incident angle $\theta_2 = 40°$. The sensor spacing is $d = \frac{1}{2}\lambda_1 = \frac{3}{8}\lambda_2$ where $\lambda_1 = \frac{c}{f_1}$ and $\lambda_2 = \frac{c}{f_2}$ are the wavelengths of the two signals, and c is the sound speed. Signal-to-Noise-Ratio (SNR) is 20 dB for the first signal, and 14 dB for the second. Using demodulation frequency 0.2, the two signals are quadrature demodulated to $f_1' = 0$ and $f_2' = -0.05$, respectively.

Running CGHA, the simultaneous learning curves of the first and the second principal eigenvectors of R_{XX} are shown in Figure 3. The relative error of the kth principal eigenvector is defined as

$$\text{Relative error} = \frac{\| V_{k,precise} - V_{k,CGHA} \|}{\| V_{k,precise} \|} \tag{60}$$

where $V_{k,precise}$ is the precise eigenvector, and $V_{k,CGHA}$ is the eigenvector learned by CGHA. $\| \cdot \|$ denotes Euclidean norm. After 3000 iterations, the relative errors are 2% for the first principal eigenvector and 1% for the second. Using AutoRegressive (AR) modeling (Kay 1988) to analyze the principal eigenvectors obtained with CGHA, we get the spatial spectra of the first and the second principal eigenvectors, as shown in Figure 4.

With the first principal eigenvector, the two spectral peaks lie at spatial frequencies 0.087 and 0.240. By the spatial frequency definition in Equation (54), the peak frequencies correspond to direction estimates $\hat{\theta}_1 = 10.0°$ and $\hat{\theta}_2 = 39.8°$. With the second principal eigenvector, the two spectral peaks lie at spatial frequencies 0.085 and 0.241, corresponding to $\hat{\theta}_1 = 9.8°$ and $\hat{\theta}_2 = 40.0°$. These estimates are very close to the true values $\theta_1 = 10°$ and $\theta_2 = 40°$.

4 Conclusion

The Complex-valued Generalized Hebbian Algorithm (CGHA) is presented in this chapter. Its convergence is proved. The algorithm can be implemented by a single-layer linear neural network. An application of CGHA to sensor array signal processing is demonstrated. Converged principal eigenvectors provide good estimates of signals'

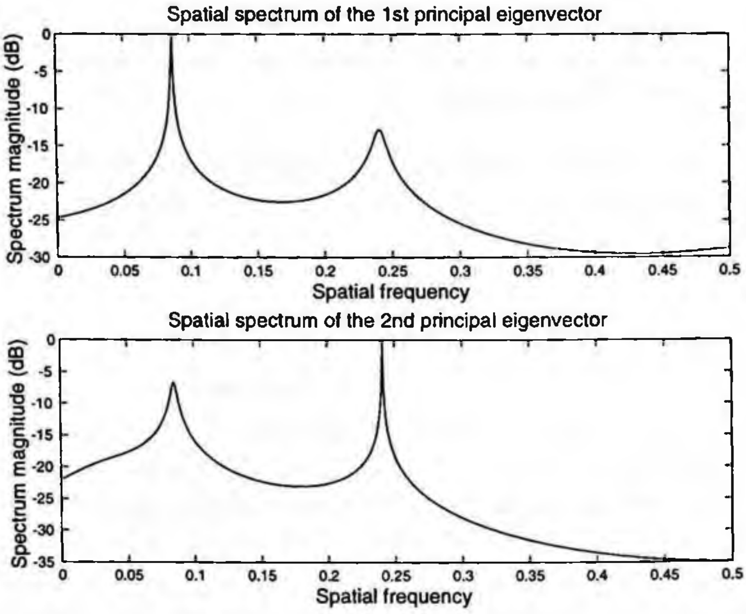


Figure 4. AR spectra of the first and the second eigenvectors.

directions.

Acknowledgment

CGHA was first presented in (Zhang *et al.* 1997). For developing Section 3 of this book chapter, the author thanks his MIT Ph.D. thesis advisor Prof. Arthur Baggeroer for the teaching in course "Sonar, Radar, and Seismic Signal Processing" for which the author was a teaching assistant. Sincere thanks go to Prof. Akira Hirose for offering the opportunity to incorporate the work into this book.

References

Bannour, S. and Azimi-sadjadi, M. (1995), "Principal Component Extraction Using Recursive Least Squares Learning," *IEEE Transactions on Neural Networks*, vol. 6, No. 2, pp. 457-469.

Cadzow, J. A. (1988), "A High Resolution Direction-of-Arrival Algorithm for Narrow-Band Coherent and Incoherent Sources," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, No. 7, pp. 965-979.

Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Company, Inc., New York.

Hornik, K. and Kuan, C. (1992), "Convergence Analysis of Local Feature Extraction Algorithms," *Neural Networks*, vol. 5, No. 2. pp. 229-240.

Kay, S. M. (1988), *Modern Spectral Estimation: Theory and Application*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

Oja, E. (1992), "Principal Components, Minor Components, and Linear Neural Networks," *Neural Networks*, vol. 5, No. 6, pp. 927-935.

Plumbley, M. (1995), "Lyapunov Functions for Convergence of Principal Component Algorithms," *Neural Networks*, vol. 8, No. 1, pp. 11-23.

Reddi, S. S. (1979), "Multiple Source Location - A Digital Approach," *IEEE Transactions on Aerospace and Electronic Systems*, , vol. AES-15, No. 1, pp. 95-105.

Sanger, T. D. (1992), "Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network," *Neural Networks*, vol. 2, No. 6, pp. 459-473.

Schmidt, R. O. (1986), "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Transactions on Antennas and Propagation*, vol. AP-34, No. 3, pp. 276-280.

Strang, G. (1993), *Introduction to Linear Algebra*, Wellesley-Cambridge Press, Wellesley, MA.

Tufts, D. W. (1998), "A Perspective on the History of Underwater Acoustic Signal Processing," *IEEE Signal Processing Magazine*, pp. 23-27, July.

Van Trees, H. L. (2002), *Optimum Array Processing*, John Wiley & Sons, Inc., New York.

Zhang, Y. and Ma, Y. (1997), "CGHA for Principal Component Extraction in the Complex Domain," *IEEE Transactions on Neural Networks*, vol. 8, No. 5, pp. 1031-1036.

Author's address

Yanwu Zhang: c/o Aware Inc., 40 Middlesex Turnpike, Bedford, MA 01730, U.S.A. Email: yanwu@alum.mit.edu

Chapter 12

Phasor Model with Application to Multiuser Communication

Teruyuki Miyajima and Kazuo Yamanaka

In this chapter, we introduce a phasor model of neural networks where the state of each neuron possibly takes the value at the origin as well as on the unit circle and show some important properties concerning the stability of an equilibrium. Moreover, an application of the phasor model to multiuser detection in code-division multiple-access (CDMA) communication is considered. In the CDMA system considered, transmitted data take complex values, and users are allowed to be in the inactive mode as well as the active mode. Simulation results show that a detector using the phasor model can outperform a conventional detector.

1 Introduction

In some recently proposed neural network models, the state of each neuron takes the value on the unit circle on the complex plane (Noest 1988, Hirose 1992, Zemel *et al.* 1995, Aoki 1995, Agu *et al.* 1996, Jankowski *et al.* 1996). The network model with a local state (state of a neuron) on the unit circle is called the phasor model. On the other hand, neurons are allowed to have the resting state "0" as well as the firing state "1" in the binary networks proposed earlier, e.g. (Hopfield 1982). This observation makes us recognize the lack of the resting states in phasor models. We have proposed a new model in which neurons are allowed to take the value at the origin as well as on the unit circle (Miyajima *et al.* 2000). The model considered here will be referred to as a *phasor model with resting states*.

In neural network applications such as associative memory or optimization problems, an equilibrium corresponds to a memory or an optimum solution. However, an equilibrium is useful as a memory or an optimum solution only if it is stable. Agu *et al.* (Agu *et al.* 1996) studied a stability property of equilibria in terms of the energy landscape of the phasor model, and showed a relation between the neural connection and the stability of an equilibrium. We have considered the same issue for the phasor model with resting states, and have obtained a stronger result (Miyajima *et al.* 2000).

In this chapter, we begin with a review of a phasor model with resting states and its stability properties and then consider an application of the phasor model with resting states to a communication system (Miyajima *et al.* 2001). Neural networks have been successfully applied to the problems of detection in communications, i.e., demodulation of transmitted data symbols (Yuan *et al.* 1990, Yuhas *et al.* 1994, Bang *et al.* 1996). We focus on the detection in code-division multiple-access (CDMA) systems which have been employed in many wireless communication standards such as IS-95 and IMT-2000. In CDMA systems, many users transmit data symbols simultaneously through a channel using a preassigned code. Because these codes are not always an orthogonal set, the transmitted signals from the separate users can interfere with each other. This interference is called multiple-access interference (MAI). At the receiver side, a multiuser detector is used to detect the data symbols from received signals corrupted by MAI. A real-valued Hopfield model was used for multiuser detection in CDMA systems where the data take real values and all users are always active (Miyajima *et al.* 1993, Kechriotis *et al.* 1996).

In more practical situations, effective modulation schemes such as QPSK where the data takes complex values are employed, and users are allowed to be inactive as well as active; e.g., the voice-activation technique (Gilhousen 1991). The phasor model with resting states is expected to be successfully applied for multiuser detection in such a CDMA system, because the complex-valued data of an active user can be represented by the state of an active neuron, and that of an inactive user can be represented by the resting state of a neuron. We derive a multiuser detector using a phasor model with resting states by relating the likelihood function of the optimum multiuser detector to the energy function of the phasor model. Moreover, we show through simulation examples that the detector provides better performance than a conventional detector.

This chapter is organized as follows. In Section 2, we describe the phasor model with resting states. We then discuss the stability properties of an equilibrium in Section 3. In Section 4, an application to CDMA communication is discussed.

2 Phasor Model with Resting States

Consider a neural network with N neurons. The local state (output) of the jth neuron is denoted by x_j. A local state is allowed to be either at the origin, $x_j = 0$, or on the unit circle, $x_j = \exp(i\phi_j)$, where ϕ_j denotes the argument of x_j.

A local state is assumed to be asynchronously updated based only on its membrane potential given by

$$u_j = \sum_{i=1}^{N} w_{ji} x_i, \tag{1}$$

where w_{ji} represents the complex-valued connection weight between the jth and ith neuron. It is assumed that the Hermitian property $w_{ji} = w_{ij}^*$ holds, where $(\cdot)^*$ stands for the complex conjugate. It is usually assumed that there is no self connection; i.e., $w_{jj} = 0$. The following discussion, however, applies to the case with self-

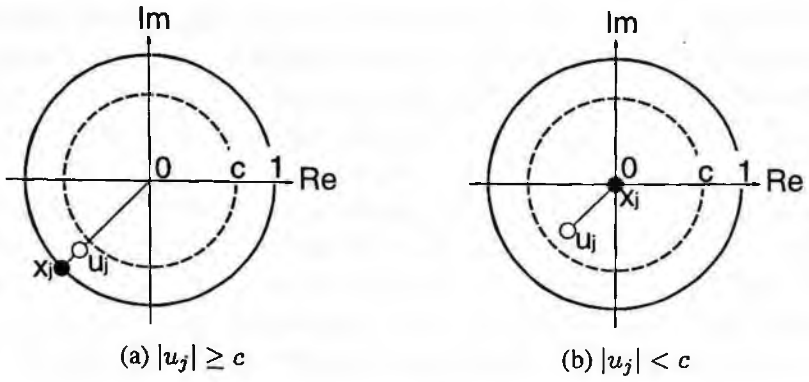


Figure 1. Updated state.

connections as well as the usual case. When a neuron is updated, the new local state is determined by the following rule:

$$x_j = \begin{cases} \exp(i\phi_j), \quad \phi_j \triangleq \arg(u_j) & (|u_j| \geq c) \\ 0 & (|u_j| < c) \end{cases}, \tag{2}$$

where $c > 0$ is a given threshold. Figure 1 illustrates the relation between the destination of the state and the membrane potential of a neuron. To summarize the updating rule, we can state as follows: If the magnitude of the membrane potential is smaller than the threshold, let the local state be at the origin; otherwise, let the local state be on the unit circle with the same argument as the membrane potential.

3 Stability Analysis

An equilibrium state is a global state where none of the local states ever change. The phasor model with resting states has equilibrium states, and here we consider a stability property of an equilibrium state. That is, we investigate whether an equilibrium state is retrievable against a small disturbance.

3.1 Energy Function

To discuss a stability property along the lines of Lyapunov's method, let us consider an energy function defined by

$$U(\mathbf{x}) \triangleq -\frac{1}{2}\sum_{j=1}^{N}\sum_{i=1}^{N} x_j^* w_{ji} x_i, \tag{3}$$

where $\mathbf{x} \triangleq (x_1 \ x_2 \ \cdots \ x_N)$ denotes a global state. The energy decreases monotonically as the global state changes in the phasor model without resting states (Agu *et al.* 1996), while this is not always true in the phasor model with resting states because of the possible local-state transition between the unit circle and the origin (see Appendix). However, the energy function can play a similar role as long as local properties are considered as described below.

Consider an equilibrium $\bar{\mathbf{x}} = (\bar{x}_1 \ \bar{x}_2 \ \cdots \ \bar{x}_N) = (\exp(i\bar{\phi}_1) \ \exp(i\bar{\phi}_2) \ \cdots \ \exp(i\bar{\phi}_N))$. Let the set of indices of the local states on the unit circle be denoted by I, i.e.,

$$I \triangleq \{j | \, |\bar{x}_j| = 1\},$$

and the number of these neurons be denoted by $|I|$. We assume

$$0 < c < \min_{j \in I} |\bar{u}_j|, \tag{4}$$

where $\bar{u}_j$ is the jth neuron's membrane potential at $\bar{\mathbf{x}}$. This assumption is needed for $\bar{\mathbf{x}}$ to be an equilibrium. We define δ-neighborhood of the equilibrium as

$$\Omega \triangleq \{\mathbf{x} | \, x_j = 0, \ j \notin I \text{ and } |\arg(x_j) - \arg(\bar{x}_j)| < \delta, \ j \in I\},$$

for an arbitrary real number $\delta > 0$. That is, the δ-neighborhood of $\bar{\mathbf{x}}$ is the set composed of the global states obtained from $\bar{\mathbf{x}}$ by changing $\arg(x_j), j \in I$ within $\pm\delta$ from $\bar{\phi}_j$ and leaving $x_j, j \notin I$ at the origin.

In the following we write "neighborhood Ω" if there is no need to specify δ. Considering the energy change in Ω, we now obtain the following result.

Lemma 1: *The function U decreases so long as the state transition occurs in a neighborhood Ω of an equilibrium.*

Proof: Suppose the jth component of $\mathbf{x} = (x_1 \cdots x_j \cdots x_N) \in \Omega$ changes to get to the new state $\mathbf{x}' = (x_1 \cdots x'_j \cdots x_N) \in \Omega$. Then, the change of the energy ΔU can be written as

$$\begin{aligned} \Delta U &= U(\mathbf{x}') - U(\mathbf{x}) \\ &= \mathrm{Re}(u_j^* x_j) - \mathrm{Re}(u_j^* x'_j)\,. \end{aligned} \tag{5}$$

By assumption, if the jth local state x_j is on the unit circle, then x'_j is also on the unit circle. Since the argument of x'_j is the same as that of u_j from (2), the first term in (5) which is the inner product of u_j and x_j is always smaller than the second term. Hence, $\Delta U < 0$. ■

3.2 Stability Property

3.2.1 Stability

In phasor models, an equilibrium $\bar{\mathbf{x}}$ has an inherent property, i.e., the global state $\bar{\mathbf{x}} \exp(i\theta)$ obtained by multiplying $\bar{\mathbf{x}}$ by $\exp(i\theta)$ with an arbitrary θ is also an equilibrium. This can be easily verified since the energy $U(\bar{\mathbf{x}})$ is equal to $U(\bar{\mathbf{x}} \exp(i\theta))$. As for the phasor model with N neurons (Agu *et al.* 1996), any one of the local states is clamped and the behavior of the other $N-1$ local states in the neighborhood of the equilibrium is considered. Despite the discontinuity of the phasor model time evolution, it has been shown in (Agu *et al.* 1996) that if the energy function is locally convex in the neighborhood of the equilibrium as a function of an arbitrary set of $N-1$ phase variables ϕ_j, the state stays in the neighborhood regardless of a small initial disturbance. This leads to the notion of persistent equilibria. This "persistency" is a property similar to stability in a weak sense.

As for the phasor model with resting states, neurons at the origin can be ignored since the neurons never affect the dynamics so long as the state transition occurs in a neighborhood Ω of an equilibrium. In other words, the phasor model with resting states can be regarded as a phasor model with $|I|$ neurons. Hence, we can expect to get a result regarding stability for the phasor model with resting states similar to that for the phasor model. In fact, we get the following result.

Proposition 2: *Choose an arbitrary $i \in I$. Suppose that ϕ_i is fixed to $\bar{\phi}_i$ and U is convex in a neighborhood Ω as a function of the other $|I| - 1$ variables $\phi_j, j \in I - \{i\}$. Let $\Omega_1 \subset \Omega$ be an arbitrary neighborhood of $\bar{\mathbf{x}}$. Then, all the sequences of the global states with initial states in a neighborhood $\Omega_0 \subset \Omega$ stay in Ω_1.*

Unlike in the phasor model, we have to take into consideration the state transition between the origin and the unit circle. If such a transition never occurs, we can prove Proposition 2 in a similar way as for the phasor model (Agu *et al.* 1996). To prove Proposition 2, we first show that such a transition never occurs during one-step transition.

Now, let a sequence of the global states generated by (1) and (2) be denoted by $\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k, \cdots$, where the subscript k increases when the global state changes. This state transition can be described formally as follows:

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k). \tag{6}$$

Then, we have the following result.

Lemma 3: *The function Φ is continuous at $\bar{\mathbf{x}}$ in terms of the neighborhood defined above.*

Proof of Lemma 3: Let $\delta' > 0$ be arbitrarily given and Ω' be the δ'-neighborhood of $\bar{\mathbf{x}}$. Take a sufficiently small $\epsilon > 0$. Then, $|\Delta u_j| < \epsilon$ means

$$|\Delta u_j| < ||\bar{u}_j| - c|, \quad j = 1, \cdots, N$$

and

$$|\arg(\bar{u}_j + \Delta u_j) - \arg(\bar{x}_j)| < \delta', \quad j \in I.$$

Furthermore, take a sufficiently small $\delta > 0$, which may depend on ϵ. Then, $|\phi_j - \bar{\phi}_j| < \delta$ assures $|\Delta u_j| < \epsilon$, because of the continuous dependence of u_i on ϕ_j. This implies that if $\mathbf{x}$ is taken in the δ-neighborhood Ω of $\bar{\mathbf{x}}$ then $\Phi(\mathbf{x}) \in \Omega'$. ■

Lemma 3 means that the state transition between the unit circle and the origin never occurs during a one-step state transition. Lemma 3 also means that the updated state still stays near $\bar{\mathbf{x}}$ regardless of a small initial disturbance at $\bar{\mathbf{x}}$. By Lemma 1 and 3, we can prove Proposition 2 which means that the state always stays near $\bar{\mathbf{x}}$ provided the initial state is sufficiently close to $\bar{\mathbf{x}}$.

Proof of Proposition 2: The local states $\bar{x}_j$ on the unit circle are renumbered as $\bar{x}_j, j = 1, \cdots, |I|$ and $\bar{x}_j$ at the origin are $\bar{x}_j, j = |I| + 1, \cdots, N$. Without loss of generality, the state of the 1st neuron is fixed to $\exp(i\bar{\phi}_1)$ and we consider the behavior of the other $N - 1$ phase variables ϕ_j , $j = 2, \cdots, N$ after they are slightly disturbed from $\bar{\mathbf{x}}$. We define the new state vectors as

$$\mathbf{z}^1 \triangleq (\phi_2 - \bar{\phi}_2 \ \cdots \ \phi_{|I|} - \bar{\phi}_{|I|}), \tag{7}$$

$$\mathbf{z}^0 \triangleq (x_{|I|+1} \ \cdots \ x_N), \tag{8}$$

and

$$\mathbf{z} \triangleq (\mathbf{z}^1 \ \mathbf{z}^0). \tag{9}$$

$\mathbf{z}$ has a one-to-one correspondence to $\mathbf{x}$ under the condition that ϕ_1 is fixed to $\bar{\phi}_1$. We define a function V as

$$V(\mathbf{z}^1) \triangleq U(\bar{\phi}_1 \, \phi_2 \cdots \phi_{|I|} \, 0 \cdots 0) - U(\bar{\phi}_1 \, \bar{\phi}_2 \cdots \bar{\phi}_{|I|} \, 0 \cdots 0). \tag{10}$$

Let δ be an appropriate positive number. Then, $|\mathbf{z}^1| < \delta$ and $\mathbf{z}^0 = \mathbf{0}$ is equivalent to $\mathbf{x} \in \Omega$ and $x_1 = \bar{x}_1$ where Ω is the δ-neighborhood of $\bar{\mathbf{x}}$ and $|\mathbf{z}^1| < \delta$ means $|\phi_i - \bar{\phi}_i| < \delta, i = 2, \cdots, |I|$. From the assumption regarding the convexity of U, V is convex in $|\mathbf{z}^1| < \delta$ as a function of $|I| - 1$ phase variables $\phi_j, j = 2, \cdots, |I|$. Therefore, we can take positive numbers a, b and ϵ_0 such that

$$a^2|\mathbf{z}^1|^2 \leq V(\mathbf{z}^1) \leq b^2|\mathbf{z}^1|^2, \quad \text{for} \quad |\mathbf{z}^1| < \epsilon_0. \tag{11}$$

Take an arbitrary $\epsilon \in (0, \epsilon_0)$ and let $\Omega_1 \subset \Omega$ be the neighborhood of $\bar{\mathbf{x}}$ defined by $|\mathbf{z}^1| < \epsilon$ and $\mathbf{z}^0 = \mathbf{0}$. Clearly, $\mathbf{x} \in \Omega_1$ means $\mathbf{z}^0 = \mathbf{0}$ since $\Omega_1 \subset \Omega$. According to the prescribed updating rule, the sequence of the state $\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k, \cdots$ can be obtained. Then, the function V is monotonically decreasing along this sequence so long as $\mathbf{x}_k$ remains in Ω_1. From (11) and the non-increasing property of V, we have the relation

$$a^2|\mathbf{z}_k^1|^2 \leq V(\mathbf{z}_k^1) < V(\mathbf{z}_0^1) \leq b^2|\mathbf{z}_0^1|^2, \quad k = 1, 2, \cdots, \tag{12}$$

so long as $\mathbf{x}_k \in \Omega_1$ and $x_1 = \bar{x}_1$. From Lemma 3, we can take a positive number $h < \epsilon$ such that $|\mathbf{z}_k^1| < h$ and $\mathbf{z}_k^0 = \mathbf{0}$ assure $|\mathbf{z}_{k+1}^1| < \epsilon$ and $\mathbf{z}_{k+1}^0 = \mathbf{0}$. Now choose $|\mathbf{z}_0^1| < \delta_0 \triangleq h(a/b)$ and $\mathbf{z}_0^0 = \mathbf{0}$, and suppose that $\mathbf{x}_\kappa$ jumps out of Ω_1, i.e. $|\mathbf{z}_\kappa^1| \geq \epsilon$ and/or $\mathbf{z}_\kappa^0 \neq \mathbf{0}$, for the first time. Then $\mathbf{x}_0, \cdots, \mathbf{x}_{\kappa-1}$ lie in Ω_1, i.e. $|\mathbf{z}^1| < \epsilon$ and $\mathbf{z}^0 = \mathbf{0}$, and $|\mathbf{z}_{\kappa-1}^1| < h$ and $\mathbf{z}_{\kappa-1}^0 = \mathbf{0}$ must hold due to (12); this contradicts $\mathbf{x}_\kappa \notin \Omega_1$ from the definition of h. Hence, all the sequences starting from $|\mathbf{z}_0^1| < \delta_0$ and $\mathbf{z}_0^0 = \mathbf{0}$ stay in Ω_1. ■

Proposition 2 ensures that the global state stays near an equilibrium if the initial disturbance is small enough. Hence, we can conclude that the equilibrium is stable if U is locally convex.

3.2.2 Asymptotic Stability

In the above discussion, we have considered the stability of an equilibrium. Now, we are interested in the asymptotic stability of the equilibrium. We have the following result.

Proposition 4: *Choose an arbitrary $i \in I$. Suppose that ϕ_i is fixed to $\bar{\phi}_i$ and U is convex in a neighborhood Ω as a function of the other*

$|I|-1$ variables ϕ_j. Then, there exists a neighborhood $\Omega_0 \subset \Omega$ such that

$$\mathbf{x}_k \to \bar{\mathbf{x}} \quad (k \to \infty)$$

whenever the sequence of the global states starts from the inside of neighborhood Ω_0.

Proof of Proposition 4: From Proposition 2, it is sufficient to consider only $|I|$ neurons, described by $\mathbf{z}^1$. Take a sufficiently small $\delta > 0$. The transition in the sequence $\mathbf{z}_0^1, \mathbf{z}_1^1, \cdots, \mathbf{z}_k^1, \cdots$ can be described formally as

$$\mathbf{z}_{k+1}^1 = \Psi(\mathbf{z}_k^1), \quad \mathbf{z}_0^1 \neq \mathbf{0}. \tag{13}$$

Clearly, $\Psi(\mathbf{0}) = \mathbf{0}$. The Lyapunov function V defined in (10) satisfies $V(\Psi(\mathbf{z}^1)) < V(\mathbf{z}^1)$ for $0 < |\mathbf{z}^1| < \delta$ by Lemma 1. Moreover, the function V is positive definite in $|\mathbf{z}^1| < \delta$ by assumption. Then, although the system (13) is a discrete-time system, we can adopt a stability analysis similar to that used for continuous-time systems (Krasovskii 1963).

Since the sequence $V(\mathbf{z}_k^1)$ has a lower bound and is monotone decreasing, there exists

$$\beta \triangleq \lim_{k\to\infty} V(\mathbf{z}_k^1) \geq 0.$$

Suppose that β is not 0. Define a set B as follows with sufficiently small $\alpha > 0$.

$$B \triangleq \{\mathbf{z}^1 | \beta \leq V(\mathbf{z}^1) \leq \beta + \alpha\} \subset \{0 < |\mathbf{z}^1| < \delta\}.$$

Then we can take $\mu > 0$ such that

$$V(\mathbf{z}^1) - V(\Psi(\mathbf{z}^1)) \geq \mu, \ \mathbf{z}^1 \in B,$$

since the function $V(\mathbf{z}^1) - V(\Psi(\mathbf{z}^1))$ is continuous and positive except at $\mathbf{z}^1 = \mathbf{0}$. On the other hand, since $\mathbf{z}_k^1 \in B$ for all k after some number, we have the relation

$$\lim_{k\to\infty}(V(\mathbf{z}_k^1) - V(\mathbf{z}_{k+1}^1)) \geq \mu > 0.$$

This creates a contradiction. Thus, β must be 0. ■

Proposition 4 means that the equilibrium is asymptotically stable if the energy is locally convex.

3.3 Sufficient Condition

Now a sufficient condition for the local convexity of the energy is derived as in the phasor model (Agu *et al.* 1996). The condition is that all the $(|I|-1)\times(|I|-1)$ principal sub-matrices of the Hessian matrix of the energy are positive definite. A sufficient condition for the positive definiteness of the principal sub-matrices is

$$\mathrm{Re}(\bar{x}_j^* w_{ji}\bar{x}_i) > 0 \quad j, i = 1, 2, \cdots, |I|. \tag{14}$$

For example, this condition is satisfied when the connection weights are determined using the Hebbian rule to store a single pattern as $\bar{\mathbf{x}}$.

3.4 Simulation Examples

In this section, we show simulation examples to illustrate the results described above. We consider two networks consisting of five neurons. Both networks have an equilibrium at

$$\bar{\mathbf{x}} = (1\ i\ \ -i\ 0\ 0).$$

We consider the motion of x_2 and x_3 with a small initial disturbance when the state of the first neuron is fixed at $x_1 = \bar{x}_1$.

First, we show a stable case. The connection weights of the first network are

$$(w_{ij}) = \begin{pmatrix} 0 & -3i & 6i & -1 & 2 \\ 3i & 0 & -9 & 2 & -i \\ -6i & -9 & 0 & i & 2 \\ -1 & 2 & -i & 0 & i \\ 2 & i & 2 & -i & 0 \end{pmatrix}.$$

The weights satisfy the condition in (14). Therefore, the energy function is convex in the neighborhood of the equilibrium. The threshold c is set to 6.5, which is smaller than $\min_{j\in I} |\bar{u}_j| = 9$. Figure 2a shows the behavior of x_2 and x_3 starting from

$$\mathbf{x}_0 = (1 \quad ie^{i3\pi/20} \quad -ie^{i\pi/10} \quad 0 \quad 0).$$

From this figure, one can observe that the disturbed state approaches the equilibrium asymptotically. Moreover, we observed that states x_4 and x_5 remain at 0.

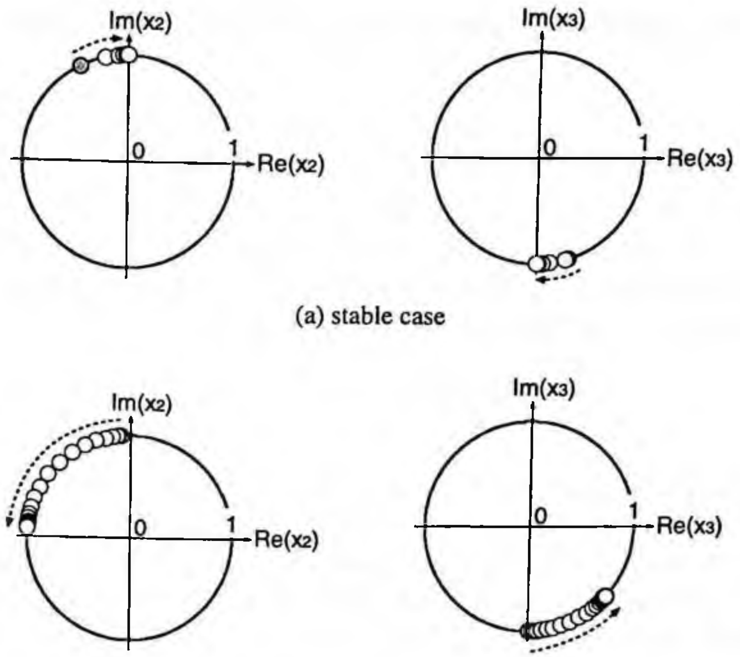
Second, we show an unstable case. The connection weights of the second network are

$$(w_{ij}) = \begin{pmatrix} 0 & 3i & 4i & -0.1 & 0.2 \\ -3i & 0 & -5 & 0.2 & -0.1i \\ -4i & -5 & 0 & 0.1i & 0.2 \\ -0.1 & 0.2 & -0.1i & 0 & 0.1i \\ 0.2 & 0.1i & 0.2 & -0.1i & 0 \end{pmatrix}.$$

The threshold c is set to 0.75, which is smaller than $\min_{j\in I} |\bar{u}_j| = 1$. Figure 2b shows the behavior of x_2 and x_3 starting from

$$\mathbf{x}_0 = (1 \quad ie^{i\pi/100} \quad -ie^{i\pi/200} \quad 0 \quad 0).$$

From this figure, one can observe that although the initial state is very close to the equilibrium the disturbed state moves far away from the equilibrium.



(a) stable case

(b) unstable case

Figure 2. State transition.

4 Application to Multiuser Communication

4.1 Communication System

Let us consider a synchronous direct-sequence/code-division multiple-access (DS/CDMA) system (Verdú 1998). The system structure is shown in Figure 3. QPSK is used as a modulation scheme. The kth user has a preassigned code denoted by $a_{kl} \in \{+1, -1\}$, $l = 0, 1, \cdots, L_c - 1$, where L_c is the length of the code.

The kth spreading waveform is generated using the code $\{a_{kl}\}$.

$$s_k(t) = \sum_{l=0}^{L_c-1} a_{kl} P_{T_c}(t - lT_c), \quad 0 \le t < T_b, \tag{15}$$

where $P_{T_c}(t)$ is a rectangular chip pulse defined by $P_{T_c}(t) = 1 (0 \le t < T_c), 0$ (otherwise), T_b is the data symbol duration, and $T_c (= T_b/L_c)$ is the chip duration. Then, the complex baseband representation of the transmitted signal can be written as

$$r_k(t) = \sum_i A'_k b_k(i) s_k(t - iT_b),$$

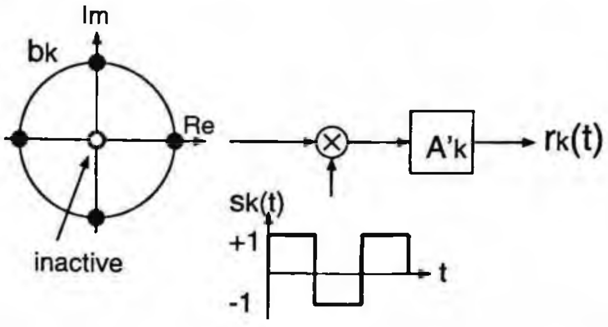
where $b_k(i)$ is a complex-valued data symbol which takes a value from $\{1, e^{j\pi/2}, e^{j\pi}, e^{j3\pi/2}\}$ with equal probability or 0, and $A'_k \in R$ is the transmitted signal amplitude. Users are allowed to be either active or inactive. If the ith user is inactive, both the transmitted symbol and amplitude are zero, $b_i = 0$ and $A'_i = 0$.

The received signal can be written as

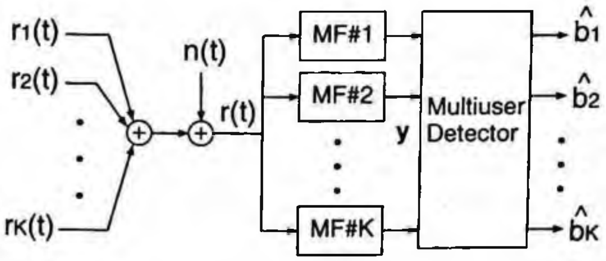
$$r(t) = \sum_{k=1}^{K} A_k \sum_i b_k(i) s_k(t - iT_b - \tau_k) + n(t), \tag{16}$$

where A_k is the received signal amplitude, which is the product of A'_k and the channel gain, $n(t)$ is a zero mean white complex Gaussian noise process with power spectral density $N_0/2$, K is the maximum number of users and τ_k is the delay of the kth user's signal. In this chapter, a synchronous CDMA system — i.e., $\tau_1 = \tau_2 = \cdots = \tau_K$ — is considered, which is a typical transmission from a base station to mobile stations in cellular communication systems.

A conventional receiver uses a matched filter (MF) which maximizes the output signal-to-noise power ratio in an additive white Gaussian noise channel with no interfering users. The output of the matched



(a) transmitter



(b) receiver

Figure 3. System structure.

filter is given by

$$\begin{aligned} y_k(i) &= \frac{1}{T_b}\int_{iT_b}^{(i+1)T_b} r(t)s_k(t-iT_b)dt \\ &= A_k b_k(i) + \sum_{l\neq k}^{K} A_l b_l(i) h_{kl} + n_k(i), \end{aligned} \tag{17}$$

where h_{kl} is the cross-correlation between the waveform $s_k(t)$ and $s_l(t)$ defined by

$$h_{kl} \triangleq \frac{1}{T_b}\int_0^{T_b} s_k(t)s_l(t)dt, \tag{18}$$

and $n_k(i)$ is the noise component given by

$$n_k(i) = \frac{1}{T_b} \int_{iT_b}^{(i+1)T_b} n(t) s_k(t - iT_b) dt.$$

In the following, we will consider the demodulation of the $i = 0$th data and omit the index i for simplicity.

In (17), the first term represents the desired component, while the second term represents MAI. Clearly if a set of orthogonal codes can be chosen, there will be no MAI. In practice, however, orthogonality cannot be ensured because of the asynchronous arrival of signals, the bandwidth limitation, and so on. Thus, codes with small cross-correlations, such as the Gold codes, are usually used. However, even if the cross-correlations are small, the effect of MAI cannot be ignored if the interference signal amplitudes A_l ($l \neq k$) are large compared to the desired signal's amplitude A_k. This is called the near-far problem which is the main disadvantage of DS/CDMA systems.

To overcome this problem, many researchers have proposed multiuser detectors; these detect the data symbols from MAI corrupted signals by using knowledge regarding the codes and signal amplitudes for all users (Verdú 1998). The most important of these is the optimum multiuser detector which selects the most probable data symbols $\{\hat{b}_k,\ k = 1, \cdots, K\}$ for active users given the received signal $r(t)$ observed over the time interval $0 \leq t \leq T_b$ (Varanasi 1995). The optimum maximum-likelihood multiuser detector computes the log-likelihood function of $\hat{\mathbf{b}} = (\hat{b}_1 \ \cdots \ \hat{b}_K)^T$

$$L(\hat{\mathbf{b}}) = \int_0^T \left| r(t) - \sum_{k=1}^{K} A_k \hat{b}_k s_k(t) \right|^2 dt,$$

where $(\cdot)^T$ represents the transpose of a matrix. If we expand the integral in the above equation, we obtain

$$L(\hat{\mathbf{b}}) = 2\text{Re}\{\mathbf{y}^H \mathbf{A} \hat{\mathbf{b}}\} - \hat{\mathbf{b}}^H \mathbf{A} \mathbf{R} \mathbf{A} \hat{\mathbf{b}}, \tag{19}$$

where $\mathbf{A} = \mathrm{diag}(A_1 \ \cdots \ A_K)$, $\mathbf{R}$ is the correlation matrix whose (k,l) component is h_{kl} defined in (18), $\mathbf{y} = (y_1 \ \cdots \ y_K)^T$, and $(\cdot)^H$ represents the conjugate transpose of a matrix. There are $4^{K'}$ possible choices of the symbols where K' is the number of active users. This optimum detector has a complexity that grows exponentially with the number of users K. When K is large, this approach is too complex computationally to be implemented in practice. Thus, this has motivated us to develop a lower complexity sub-optimum detector.

4.2 Multiuser Detector Using the Phasor Model with Resting States

We will now show how a phasor model with resting states can be used for low-complexity sub-optimum multiuser detection. The key idea is that the phasor model is used to solve the maximization problem of (19).

If there is the Nth neuron whose output is always 1, the energy function can be rewritten as

$$U(\mathbf{x}) = -\frac{1}{2}\sum_{j=1}^{N-1}\sum_{i=1}^{N-1} w_{ji}x_j^* x_i - \sum_{i=1}^{N-1} \mathrm{Re}(w_{Ni}x_i) - \frac{1}{2}w_{NN}|x_N|^2. \quad (20)$$

Comparing the energy in (20) and the likelihood function in (19), the parameters of the phasor model can be determined as follows,

$$\begin{aligned} N &= K+1, \\ w_{ij} &= \begin{cases} -2h_{ij}A_iA_j, & i,j \neq N, \\ 2y_iA_i, & i \neq N, j = N, \\ 0, & i = j = N, \end{cases} \\ x_N &= 1 \end{aligned} \quad (21)$$

Given these parameters and appropriate initial state, the phasor model updates its state according to the rule described in Sec.2. The output of the kth neuron, x_k, after the dynamics converge is the estimation of the kth user's symbol, $\hat{b}_k$.

In practice, the treatment of the self-connections $\{w_{ii}\}$ is an important issue. The self-connections determined by (21) are not zero. The properties presented in the previous section hold for non-zero self-connections. For the real-valued case, though, it has been recommended that the self-connections be set to zero to obtain better performance (Miyajima *et al.* 1993). The contribution of the self-connections to the energy function, which is

$$\sum_{j=1}^{N-1} w_{jj}|x_j|^2,$$

is constant as long as the state transition between the origin and the unit circle does not occur. Therefore, the relative energy landscape can be maintained even if the self-connections are forced to zero. In our simulation, the self-connections were set to zero.

An interesting interpretation of the detector is obtained by looking into the membrane potential of the ith neuron with zero self-connection:

$$u_i = 2A_i \left(y_i - \sum_{j \neq i}^{K} A_j h_{ij} x_j \right).$$

Since x_j corresponds to the estimate of the data symbol b_j, the second term in the parentheses can be regarded as a replica of the MAI. Hence, the ith neuron estimates the data symbol b_i by canceling the MAI replica from the corresponding matched filter output y_i given in (17). This cancellation principle is used in common among conventional multiuser detectors, such as the parallel interference canceller, serial interference canceller, and so on (Verdú 1998). Note that the data symbol estimate is refined iteratively through the dynamics of the phasor model in the phasor-model detector.

Note also that the optimum solution can be obtained only if the state of the phasor model converges to the global minimum of the energy function. If the state converges to one of the local minima, the performance of the phasor-model detector may deteriorate. However, we expect the performance deterioration to be as little as in the real-valued cases (Miyajima *et al.* 1993). The convergence depends strongly on the initial state. Fortunately, there exists a reasonable initialization strategy in this application. The output of the matched filters can be used as an initial state. To obtain better performance, we may use the outputs of the other conventional multiuser detectors such as the decorrelator or minimum mean-square-error detector (Verdú 1998) as an initial state.

The hardware complexity of the proposed detector will clearly increase linearly with the number of users. On the other hand, the computational complexity is hard to evaluate since it depends on the number of iterations needed for convergence. The number of iterations depends not only on the number of users K, but also on other factors such as the received signal amplitudes A_k, the magnitude of the channel noise N_0, the codes a_{kl}, and so on. In practice, as in the conventional iterative detectors, the proposed detector may provide acceptable performance even if the iteration is terminated at the predetermined number.

Note that the threshold c should be chosen carefully. The condition (4) must be satisfied to make the desired point an equilibrium. However, the receiver cannot know the minimum magnitude of $\bar{u}_j$ in advance since the desired point is unknown. Thus, in our simulation, we used a threshold of

$$c = c' \left(\min_{j \in I} \sum_{i=1}^{N} |w_{ji}| \right),$$

where $I = \{j |\, A_j \neq 0\}$ and $0 < c' \leq 1$. Since

$$|\bar{u}_j| = \left| \sum_{i=1}^{N} w_{ji} \bar{x}_i \right| \leq \sum_{i=1}^{N} |w_{ji}|,$$

c' should be chosen so that it is not too large to satisfy condition (4).

Moreover, if the threshold is too large, the detector fails to detect active users. Thus, the threshold should be set as small as possible.

4.3 Simulation Results

Through computer simulation we have evaluated the performance of the detector using the phasor model with resting states. The maximum number of users was $K = 4$, and the fourth user could be inactive; i.e., $A_4 = b_4 = 0$. The signal energy per bit for the first user was equal to that for the third user; i.e., $E_1 = E_3$. The second user's signal energy E_2 was varied. The spreading sequences used were Gold sequences of length 7:

$$\begin{array}{l} (\ 1\ \ 1\ -1\ -1\ \ 1\ -1\ -1\ ) \\ (\ 1\ \ 1\ \ 1\ -1\ -1\ \ 1\ \ 1\ ) \\ (\ 1\ -1\ \ 1\ \ 1\ \ 1\ -1\ \ 1\ ) \\ (\ -1\ -1\ -1\ -1\ -1\ -1\ \ 1\ ) \end{array}.$$

The dynamics were terminated at 1,000 iterations regardless of whether they had converged. The results were obtained by averaging over 100,000 symbols.

First, we considered the effect of the threshold. Figure 4 shows the detection probability of the active users. Observe that for a threshold parameter larger than 0.2, the detector failed to detect active users. In the following simulation, the threshold parameter c' was set to 0.01.

Two examples of the transmitted symbols and the dynamics of four neurons are shown in Figures 5a and b. In both figures, the upper rows show the data symbols, and the lower rows show the time development of neural states. In the first case in Figure 5a where all users are active, the state of the neurons correctly converged to the point corresponding to the transmitted data symbol. In the second case in Figure 5b where the fourth user was inactive, the state of the fourth neuron converged to the origin.

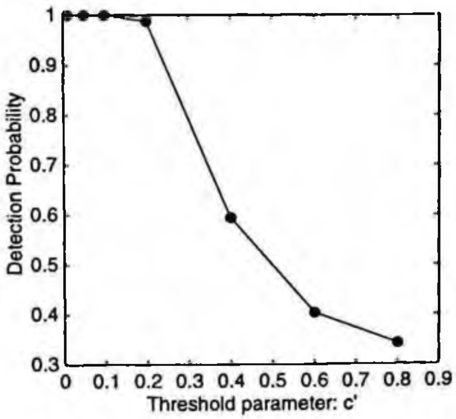
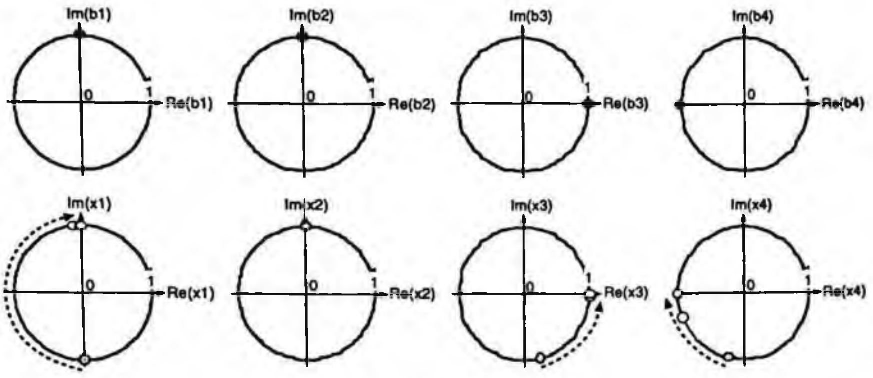


Figure 4. Detection probability.

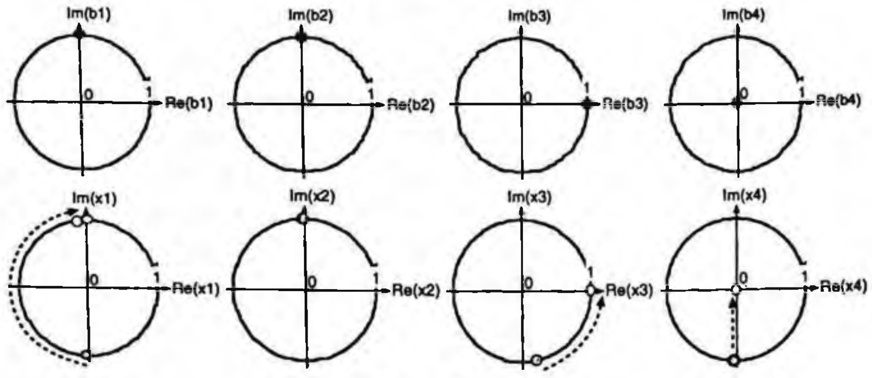
Finally, we compared the performance of the proposed detector with that of the conventional matched filter detector. In Figures 6a and b, the bit error rate performance for the first user is shown. The performance of the conventional detector deteriorated as the interference became stronger, and the proposed detector performed better than the conventional one. State x_4 always converged to 0.

5 Conclusion

We have introduced a phasor model where the state of each neuron can take the value at the origin as well as that on the unit circle, and have shown that its equilibrium is asymptotically stable if the energy function is locally convex. Moreover, we have considered the application of this model to multiuser detection in a CDMA system where the data take complex or zero values. Simulation results showed that use of the new phasor model enabled the detector to outperform the conventional matched filter in terms of interference reduction.



(a) all users are active



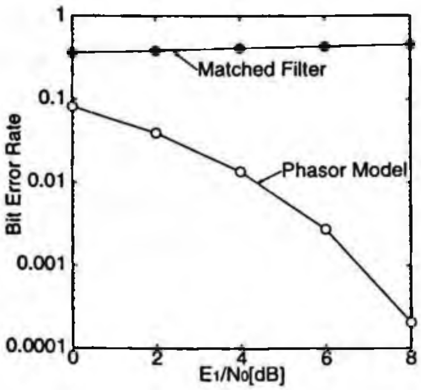
(b) fourth user is inactive

Figure 5. Data symbols and neuron dynamics.

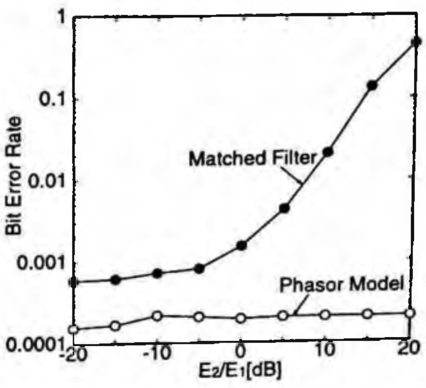
Appendix: Energy Change for Global State Transition

Suppose the jth neuron changes its state. Denote the state before and after the transition as $\mathbf{x} = (x_1 \cdots x_j \cdots x_N)$ and $\mathbf{x}' = (x_1 \cdots x_j' \cdots x_N)$, respectively. Then, the energy change is given by

$$\Delta U = \mathrm{Re}(u_j^* x_j) - \mathrm{Re}(u_j^* x_j').$$



(a) BER vs. E_1/N_0 ($E_2/E_1 = 20$dB)



(b) BER vs. E_2/E_1 ($E_1/N_0 = 8$dB)

Figure 6. Performance comparison.

If the jth neuron changes its state from the origin to the unit circle, then

$$\Delta U = -|u_j| \cos(\arg(x'_j) - \arg(u_j)),$$

Since $\arg(x'_j) = \arg(u_j)$, the energy decreases. If the jth neuron changes its state from the unit circle to the origin, then

$$\Delta U = |u_j| \cos(\arg(x_j) - \arg(u_j)),$$

which can be positive. This implies that the energy does not necessarily decrease.

References

Agu, M., Yamanaka, K., and Takahashi H. (1996), "A local property of the phasor model of neural networks," *IEICE Trans. on Inform. & Syst.*, vol. E79-D, pp.1209–1211.

Aoki, H. (1995), "Analysis of equilibrium states of Hopfield associative memory extended to complex number field," *IEICE Trans. A*, vol.J78-A, pp.1238-1241 (in Japanese).

Bang, S. H., and Shen, B. J. (1996), "A neural network for detection of signals in communication," *IEEE Trans. Circuits and Syst. I*, vol.43, no.8, pp.644–655.

Gilhousen, K.S., Jacobs, I.M, Padovani, R., Viterbi, A.J., Weaver, Jr. L.A., and Wheatley III, C.E. (1991), "On the capacity of a cellular CDMA system," *IEEE Trans. Vehicul. Technol.*, vol.44, no.2, pp.303–312.

Hirose, A. (1992), "Dynamics of fully complex-valued neural networks," *Electronics Lett.*, vol.28, pp.1492-1494.

Hopfield, J. J. (1982), "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol.79, pp.2554-2558.

Jankowski, S., Lozowski, A., and Zurada, J. M. (1996), "Complex-valued multistate neural associative memory," *IEEE Trans. on Neural Networks*, vol.7, pp.1491-1496.

Kechriotis, G. I., and Manolakos, E. S. (1996), "Hopfield neural network implementation of the optimal CDMA multiuser detectors," *IEEE Trans. Neural Networks*, vol.7, no.1, pp.131–141.

Krasovskii, N. N. (1963), *Stability of motion,* Stanford Univ. Press, Stanford, California.

Miyajima, T., Hasegawa, T., and Haneishi M. (1993), "On the multiuser detection using a neural network in code-division multiple-access communications," *IEICE Trans. on Commun.*, vol. E76-B, pp.961-968.

Miyajima, T., Baisho, F., Yamanaka, K., Nakamura, K., and Agu, M. (2000), "A phasor model with resting states," *IEICE Trans. on Inform. & Syst.*, vol. E83-D, pp. 299-301.

Miyajima, T., and Yamanaka, K. (2001), "An application of a phasor model with resting states to multiuser detection," *Knowledge-based Intelligent Information Engineering Systems & Allied Technologies (KES'2001)*, Part I, pp.571–575.

Noest, A. J. (1988), "Associative memory in sparse phasor neural networks," *Europhys. Lett.*, vol. 6, pp. 469-474.

Varanasi, M. K. (1995), "Group detection for synchronous Gaussian code-division multiple-access channels," *IEEE Trans. on Inform. Theory*, vol. 41, pp.1083-1096.

Verdú, S. (1998), *Multiuser detection,* Cambridge Univ. Press, Cambridge.

Yuan, J., and Chen, C.S. (1990), "Neural net decoders for some block codes," *IEE Proc.*, vol.137, Pt.I, no.5, pp.309–314.

Yuhas, B., and Ansari, N. (1994), *Neural network in telecommunications*, Kluwer Academic Publishers.

Zemel, R. S., Williams, C. K. I., and Mozer, M. C. (1995), "Lending direction to neural networks," *Neural Networks,* vol.8, pp.503-512.

Authors' address

Teruyuki Miyajima: Department of Systems Engineering, Faculty of Engineering, Ibaraki University, 4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan.
Kazuo Yamanaka: Department of Systems Engineering, Faculty of Engineering, Ibaraki University, 4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan.

Chapter 13

Adaptive Interferometric Radar Image Processing by Using Complex-Valued Neural Network

Andriyan Bayu Suksmono and Akira Hirose

This Chapter presents an application of the complex-valued neural network (CVNN) for interferometric radar (InSAR-Interferometric Synthetic Aperture Radar) image processing. The InSAR image, whose pixels are naturally represented by complex numbers, is modeled by CMRF (Complex-valued Markov Random Field). Then the InSAR image that is corrupted by noise-induced singular points (SP) is mapped to a CMRF lattice neural network (CMRF-LNN). By evolving the states of the CMRF-LNN toward a minimum energy value, the SP number is eventually reduced. The advantages for phase unwrapping are demonstrated for a simulated as well as real InSAR images.

1 Introduction

Radar imaging system is a valuable tool to study the earth in a global scale. The usage of radio waves in the radar system is an advantage due to its capability of observing the earth in all weather conditions, day or night.

By using aperture synthesis technique, a SAR (Synthetic Aperture Radar) system is capable of achieving high resolution with a moderately small antenna. The SAR system transmits coherent microwave radio signals, receiving the echoes reflected by terrain surface and storing the data. Then, a process to synthesize large aperture is performed to obtain the radar image. By applying interferometry principle, the SAR system in the InSAR (Interferometric SAR) is capable to measure terrain altitude. Figure 1 shows InSAR imaging system by a satellite.

There are two kinds of information obtained by the InSAR system, the amplitude and the phase. The amplitude corresponds to surface reflectivity, while the phase to height variation. Usually these data are used separately for different purposes. For example, the amplitude is used for land classification while the phase is for DEM (Digital Elevation Map) construction.

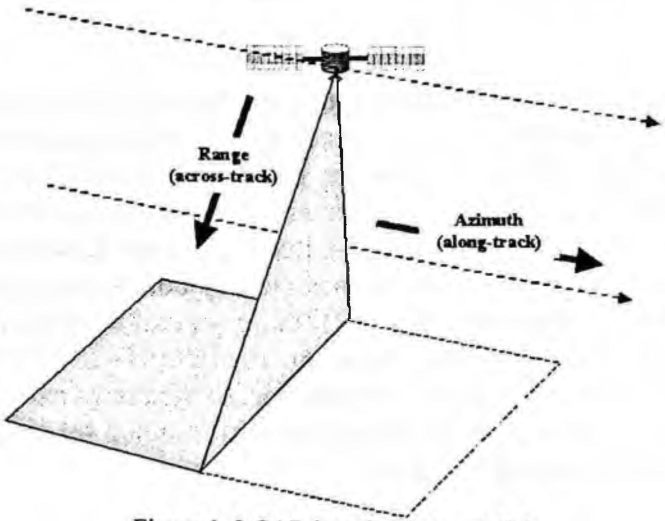


Figure 1. InSAR imaging by a satellite.

Artificial neural network (ANN) is an interconnected multiprocessor system that can be used for adaptive signal processing applications with many advantages. Recent progress shows that the ANNs that possess complex-valued weights, the CVNN (Complex Valued Neural Network), have more advantages in some applications, especially for those that involved complex-valued information to be manipulated (Hirose 1999). The possibility to apply the CVNN in a phase sensitive adaptive radar imaging system has been proposed in (Hirose and Sugiyama 1998).

This Chapter presents a novel method for adaptive processing of InSAR images by treating the amplitude and phase image as a single complex-valued image. Extension of the Markov random field (MRF) to complex-valued MRF (CMRF) as a model of the complex image is performed. The CMRF will be embedded in a lattice neural network (LNN), resulting in CMRF-LNN architecture, to manipulate the InSAR images.

This Chapter is organized as follows. Section 2 introduces the basic of InSAR imaging concepts and coherent phase noise. The CMRF model as an extension of conventional MRF is introduced in Section 3. In Section 4, two new methods that utilize the CMRF-LNN system to filter the InSAR image adaptively are proposed; the first method employs stochastic Monte Carlo Metropolis algorithm and the second one utilizes the steepest descent. Experiments that demonstrate the capability of the methods are also presented in this Section. Section 5 concludes this Chapter.

2 InSAR Imaging and Phase Noise

To measure the topography by means of electromagnetic sensing, we need phase difference of the electromagnetic wave reflected by the object observed from two nearby positions. There are two main techniques in the InSAR imaging: single-pass interferometry and repeat-pass interferometry. In the single-pass interferometry, an aircraft or spacecraft (such as in SRTM–the Shuttle Radar Topographic Mission) is equipped with two receiver antennas that capable of measuring phase difference between the received electromagnetic waves. In the repeat-pass technique, the radar system visits a region in two different times at a slightly different position such that the phase difference can be obtained. The InSAR image used in this experiment is obtained by the repeat-pass interferometry of JERS-1 SAR satellite.

Figure 2 shows a simple geometry of InSAR observation, relating some important parameters. In the figure, S_1 and S_2 are two satellite positions that are separated by a baseline b. The normal component of the baseline, b_n, is called the effective baseline. The distance between a radar source in S_1 and point P is r, and the distance between a radar source in S_2 with P is $r+\Delta r$. The phase difference ϕ, the interferometric phase, is related to the wavelength λ and Δr as:

$$\phi = \phi_1 - \phi_2 = -\frac{4\pi}{\lambda}\Delta r \qquad (1)$$

where ϕ_1 and ϕ_2 are the phase received at S_1 and S_2 respectively. By knowing S_1, S_2 and the satellite altitude h from the orbit information provided by single look complex (SLC) data, the position of point P can be determined. It is obvious that P and its height from a predetermined reference point on the ground Δz depend on the interferometric phase ϕ. The further it is from the satellite, the greater is the phase. Therefore, to construct a topography map, an absolute phase value is needed.

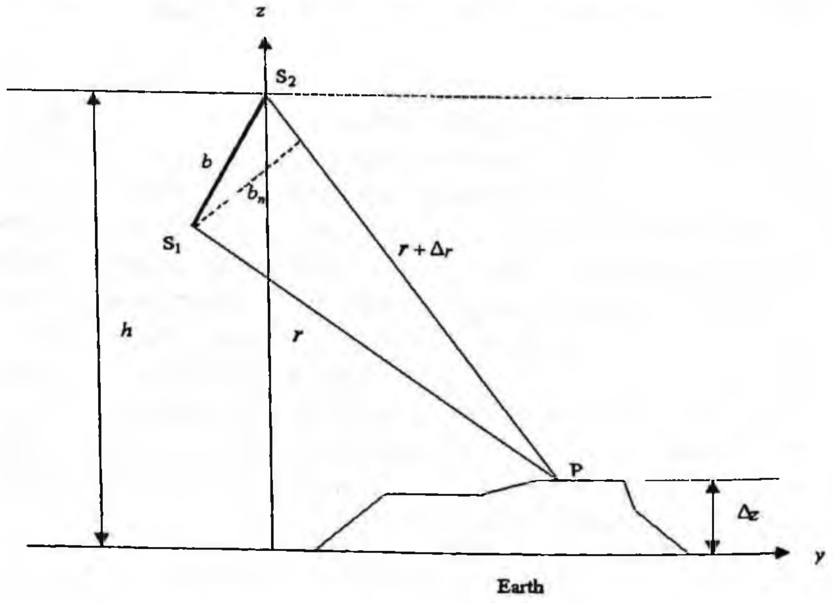


Figure 2. A Simple InSAR observation geometry.

To obtain the interferometric phase ϕ, two SLC data that represent observation from two different positions are needed. Then these SLC data are co-registered, which result in two complex images: $z_1 = |z_1|.e^{j\phi_1}$ and $z_2 = |z_2|.e^{j\phi_2}$, where $j = \sqrt{-1}$. Phase image is obtained by taking the argument of the product of the first image with the complex conjugate of the second one:

$$z_{\text{int}} = z_1 \cdot z_2^{*} = |z_1||z_2|e^{j(\phi_1 - \phi_2)} \qquad (2)$$

The phase difference (ϕ_1-ϕ_2) is determined from the real and imaginary parts of z_{int} by *arctan* function. Therefore we only get the principal (wrapped) value ψ of the phase:

$$\psi = W(\phi_1 - \phi_2) \qquad (3)$$

where W is an operator that wraps the phase into interval $(-\pi,\pi]$.

Coherence interference due to reflection by random scatterers degrades the complex (-valued) image. Using the complex-amplitude model, the noise can be treated consistently as complex–valued multiplicative noise. It is obvious that the noise effect on the amplitude will be multiplicative while the effect on the phase will be additive.

To construct a topography map, we must obtain the absolute value of the phase image by phase unwrapping (PU) process. It is performed by adding/subtracting a multiple of 2π whenever a fringe edge is detected. However, this procedure will become complicated in the presence of residues/singular points (SP).

The SPs are the points where $\nabla \times \psi \neq 0$, i.e., the points that violate the nature of the conservation of altitude. In the presence of SPs, error at one location is propagated to the entire image.

The SPs can be detected as follows (Ghiglia et.al 1996). Let ψ_{mn} denotes the wrapped phase image of size $M\times N$. Horizontal and vertical phase differences f_{mn} and g_{mn} are defined as:

$$f_{mn} = \nabla_m \psi_{mn} = \begin{cases} W\{\psi_{m+1\,n} - \psi_{mn}\}, & m = 0,\dots,M-2;\ n = 0,\dots,N-1 \\ 0 & \text{, otherwise} \end{cases} \tag{4}$$

$$g_{mn} = \nabla_n \psi_{mn} = \begin{cases} W\{\psi_{m\,n+1} - \psi_{mn}\}, & m = 0,\dots,M-1;\ n = 0,\dots,N-2 \\ 0 & \text{, otherwise} \end{cases} \tag{5}$$

The SPs are detected by computing

$$r_{mn} = g_{mn} - g_{m+1\,n} + f_{mn} - f_{m+1\,n} \tag{6}$$

If $r_{mn} > 0$, we have positive SP at (m,n). If $r_{mn} < 0$, negative SP, and if $r_{mn} = 0$, then there is no SP.

In the following Section, we present an adaptive method that capable to reduce the SPs significantly by employing CMRF-LNN. After the SPs in the phase image are reduced, the PU process is become easier to be performed.

3 MRF and CMRF Model for Image Processing

3.1 Conventional MRF Model

We follow the definition of MRF in (Chellappa et.al. 1991) and (Cross and Jain 1983). A site (pixel position) is denoted by *s*, while the intensity (gray scale) value is denoted by x_s. The 'colour' of the pixel corresponds to the brightness.

Let a finite rectangular $M \times N$ lattice L represents an image. A G-levels *colouring* of lattice L denoted by x_s is a function at the site-s in L to the set $\{0,1, \dots , G\text{-}1\}$. Sites $t \in N_s$ are called neighbours of site s if the conditional probability of the colour x_s depends only on the sites $t \in N_s$, i.e.,

$$P\left(x_s \middle| x_1, x_2, \dots, x_{s-1}, x_{s+1}, \dots, x_{M \times N}\right) = P\left(x_s \middle| x_t ; t \in N_s\right) \quad (7)$$

Markov random field (MRF) is defined as a field that has a joint probability density on the set of all possible colourings x_s for all sites-s of the lattice L subject to the conditions of positivity, Markovianity and homogeneity.

According to the Ising model (Cross and Jain 1983), the probability value is determined by the pixel's energy $E_f(x_s)$ and environment's temperature T. The energy depends on the neighbourhood configuration, order of the model and interaction strength between sites. The probability of a site-s, whose intensity is x_s, with a neighborhood t $(\in N_s)$ can be expressed as:

$$P(x_s) = \frac{1}{Z} e^{\frac{-E(x_s)}{T}} \quad (8)$$

$$E(x_s) = -\left(\alpha_s x_s + \sum_{t \in N_s} \Theta_{st} x_s x_t \right) \quad (9)$$

where Z is the partition function for normalization, α_s and Θ_{st} are MRF model parameters, and N_s is the neighborhood set of site s. The model parameter Θ_{st} is the interaction strength between sites s and t, while α_s corresponds to the strength of the external field.

This model is called autobinomial model, which is an extension of the two-valued Ising model to multi-valued one. In (Chellappa et.al. 1991), the definition of energy has a slightly modified form. Assuming a homogenous field, a similar energy E can be defined as:

$$P(x_s) = \frac{1}{Z} e^{-E(x_s)} \tag{10}$$

$$E(x_s) = \frac{1}{2\sigma^2(T)} \left(x_s^2 - 2 \sum_{t \in N_s} \Theta_{st} x_s x_t \right) \tag{11}$$

where $\sigma^2(T)$ is variance that corresponds to the temperature T. We use these expressions (8)-(11) to obtain the energy function based on the CMRF model and the restoration procedure in the next section.

3.2 Complex-Valued MRF (CMRF)

An InSAR image consists of amplitude and phase images. Therefore, we have to combine both of these images into a single complex-valued image because the amplitude and the phase are physically inseparable properties of the electromagnetic wave used in the radar system. We are going to model the combined image using the CMRF. We cannot directly extend equation (9) by replacing the real-valued pixel x with a complex-valued pixel z because it will give an ambiguous complex-valued energy $E(z)$. This can be verified by observing its terms, z_s and the product $(\theta_{st} z_s z_t)$, which are generally complex-valued. We propose two possibilities.

1) *The Error Energy*: $E(x_s)$ in eq. (11) is a function of x_s. Assuming that $\Theta_{st}x_t$ is invariant, we can reinterpret $E(x_s)$ as a squared-error energy

$$E(x_s) = \frac{1}{2\sigma^2(T)}(x_s - \hat{x}_s)^2$$

$$\hat{x}_s \equiv \sum_{t \in N_s} \Theta_{st} x_t$$

where $\hat{x}_s$ is the estimated value of x_s. Therefore, generalization into CMRF will be:

$$E(z_s) = \frac{1}{2\sigma^2(T)}\|z_s - \hat{z}_s\|^2 \tag{12}$$

$$\hat{z}_s \equiv \sum_{t \in N_s} \Lambda_{st} z_t \tag{13}$$

The CMRF parameters Λ_{st} can be estimated by least square method. In this method, the estimation criteria is the minimum mean square error (MMSE), that is to say, it has to minimize the following function:

$$E_\varepsilon = \frac{1}{2\sigma^2(T)(M \times N)} \sum_{s \in L} \|z_s - \hat{z}_s\|^2 \tag{14}$$

Based on the least square method (Chellappa et.al. 1991) for the parameter estimation, it can be verified that the complex-valued forms are given by the following equations:

$$\hat{\Lambda}^* \equiv \left[\sum_{s \in L} z_s Q_s^*\right]\left[\sum_{s \in L} Q_s Q_s^*\right]^{-1} \tag{15}$$

$$\hat{\sigma}(T)^2 \equiv \frac{1}{M \times N} \sum_{s \in L} \left\| z_s - \hat{\Lambda}^* Q_s \right\|^2 \tag{16}$$

$$Q_s = \begin{bmatrix} z_{s+\tau_1} \\ z_{s+\tau_{-1}} \\ \cdots\cdots \\ z_{s+\tau_{12}} \\ z_{s+\tau_{-12}} \end{bmatrix} \tag{17}$$

where Q_s is neighborhood vector consisting of 5th order neighbor values $z_{s+\tau_t}$, $\hat{(.)}$ means the estimated value, $(.)^*$ is complex conjugate operation, and $\hat{\sigma}(T)^2$ is the estimated variance for the initial image. Here we consider 5th order neighborhood N_s that has the structure shown in Fig. 3.

$z_{s+\tau_{-12}}$	$z_{s+\tau_{-11}}$	$z_{s+\tau_1}$	$z_{s+\tau_2}$	$z_{s+\tau_3}$
$z_{s+\tau_{-10}}$	$z_{s+\tau_{-9}}$	$z_{s+\tau_4}$	$z_{s+\tau_5}$	$z_{s+\tau_6}$
$z_{s+\tau_{-8}}$	$z_{s+\tau_{-7}}$	z_s	$z_{s+\tau_7}$	$z_{s+\tau 8}$
$z_{s+\tau_{-6}}$	$z_{s+\tau_{-5}}$	$z_{s+\tau_{-4}}$	$z_{s+\tau_9}$	$z_{s+\tau_{10}}$
$z_{s+\tau_{-3}}$	$z_{s+\tau_{-2}}$	$z_{s+\tau_{-1}}$	$z_{s+\tau_{11}}$	$z_{s+\tau_{12}}$

Figure 3. MRF (or CMRF) neighborhood configuration.

2) *CMRF complex Ising energy*: A paper on complex-valued Hopfield network (a type of fully connected neural network) described in (Hirose 1992) gives a clue to solve this problem by taking the real part of a complex-valued energy $E(z)$. It is well known that the Hopfield neural network is isomorphic to the Ising (spin glass) model (refer to Haykin 1994, pp.308 for example), whose energy is in the form of (9) with a restriction that the pixel value is -1 or +1. We will follow this analysis by taking the case of α_s=0 (no external field).

Based on equation (5) of (Hirose 1992), by changing the matrix notation from $\boldsymbol{W}$ into $\boldsymbol{\Lambda}$, the vector $\boldsymbol{x}$ by z (in component form: x_s), and by regarding only the neighborhood interaction such that our field satisfies the Markovianity condition instead of the fully connected network (which means interaction of a site with all other sites), we can write the complex-valued lattice energy (complex-valued Ising energy) E_{CI} as

$$E_{\mathrm{CI}} = -\sum_{s\in L}\frac{1}{2}\mathrm{Re}\left(\sum_{t\in N_s}\Lambda_{st} z_s^{*} z_t\right) = -\sum_{s\in L}\frac{1}{2}\mathrm{Re}\left(z_s^{*}\sum_{t\in N_s}\Lambda_{st} z_t\right) \tag{18}$$

When the value z_s is ideally estimated by its neighbors and the CMRF parameters, the estimate $\hat{z}_s$ should become (13). Then the energy (without the external field, i.e. α_s=0) can be rewritten as

$$E_{\mathrm{CI}} = -\sum_{s\in L}\frac{1}{2}\mathrm{Re}\left(z_s^{*}\hat{z}_s\right) \tag{19}$$

Both the E_{CI} and E_{ε} are useful in Monte Carlo method because the two energies are almost the same. However, by observing the following expression:

$$\|z_s - \hat{z}_s\|^2 = \|z_s\|^2 + \|\hat{z}_s\|^2 - 2\,\mathrm{Re}\left[z_s^{*}\hat{z}_s\right] \tag{20}$$

we see that E_ε is more suitable in the present application because the brightness $\|z_s\|$ and $\|\hat{z}_s\|$ may vary largely.

4 Adaptive Phase Noise Filtering and InSAR Image Restoration Using CVNN

4.1 System Construction and Neurodynamics

In the InSAR images, each pixel at a certain location of the amplitude image is associated with its counterpart in the phase image. They are both combined to yield a complex-valued image.

Each pixel in the image corresponds to a point of a two-dimensional complex-valued lattice. From the CMRF point of view, a point in the lattice has influence of only its nearest neighbors. During the restoration process to be explained, an energy function depending on the value of a pixel and their neighbor values is defined. Then, by a certain dynamics, the energy is reduced toward equilibrium point of minimum energy configuration. The mapping of complex-valued pixels and the energy reduction process define the dynamic of the CMRF-LNN. Figure 4 illustrates the interaction between a pixel and its nearest neighbors for a second order CMRF- LNN. In contrast to a Hopfield neural network where all of neurons interact with each other, a neuron in the CMRF-LNN only interacts with its nearest neighbors.

During the restoration process, the complex-valued image is divided into small windows where stationary statistical condition is assumedly be satisfied. In this research, a window with size 64×64 pixels is chosen. In each window, SPs are detected and it as well as its neighbors are marked. The CMRF parameter is estimated by choosing the unmarked area whenever possible, or it can be used as it is anyway if the image is greatly corrupted. Then the restoration algorithm is applied. The diagram block of the system is depicted in Fig.5. In the diagram, filled circle indicates a SP position while the gray ones are its neighbors. These pixels form a mask where the image value should be updated. The size of the mask can be chosen adaptively. Here two filtering methods (the Adaptive Algorithm block) are proposed, i.e., the Monte Carlo Metropolis (MM) and the steepest descent methods.

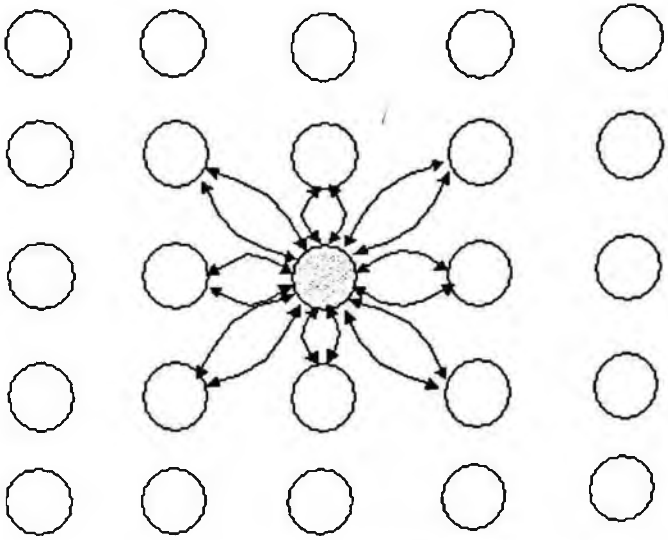


Figure 4 . Neighborhood interaction in CMRF-LNN.

4.2 CMRF Filtering by Monte Carlo Metropolis Algorithm

In this method, the estimation criteria is the MMSE evaluated to the entire complex-valued image in a *M*×*N* block, that is to say, it has to minimize (14). The restoration process is achieved by decreasing the energy function (14) by using the Monte Carlo Metropolis

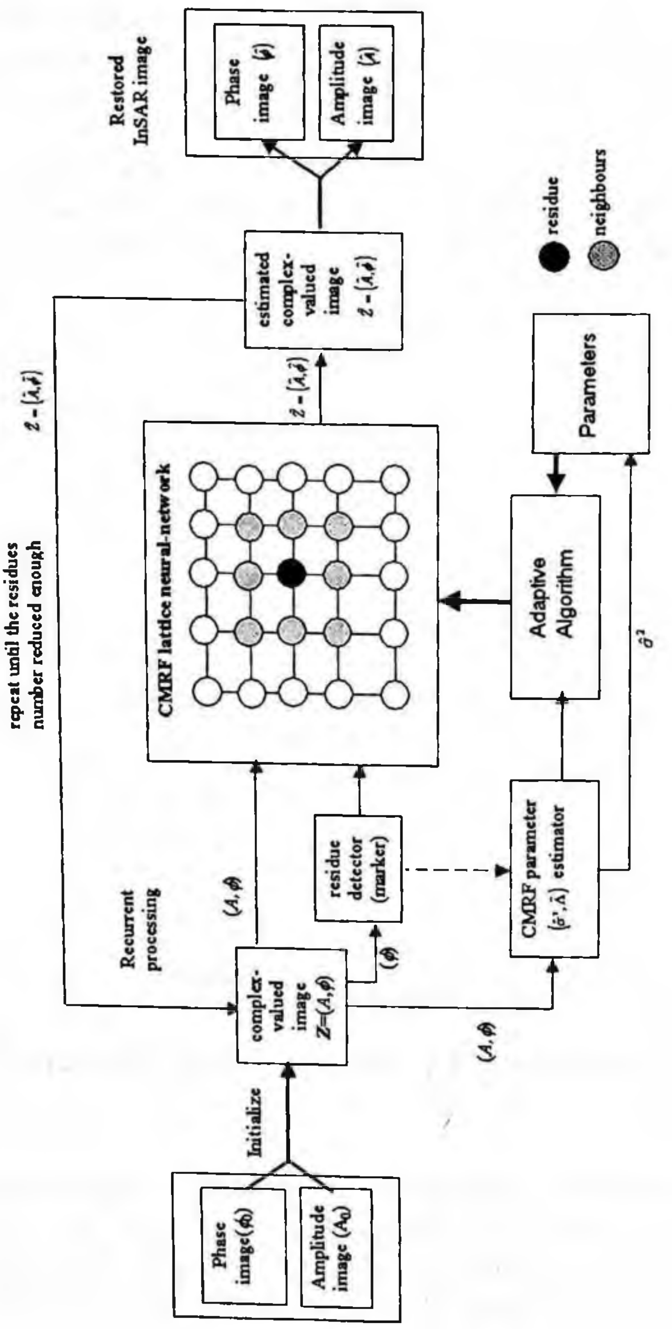

Phase image(ϕ_0)
Amplitude image (A_0)
Initialize
complex-valued image $Z=(A,\phi)$
Recurrent processing
(A,ϕ)
(ϕ)
(A,ϕ)
residue detector (marker)
CMRF parameter $(\hat{\sigma}^2,\hat{\lambda})$ estimator
repeat until the residues number reduced enough
CMRF lattice neural-network
Adaptive Algorithm
Parameters
$\hat{\sigma}^2$
$\hat{Z}=(\hat{\lambda},\hat{\phi})$
$\hat{Z}=(\hat{\lambda},\hat{\phi})$
estimated complex-valued image $\hat{Z}=(\hat{\lambda},\hat{\phi})$
Restored InSAR image
Phase image $(\hat{\phi})$
Amplitude image $(\hat{\lambda})$
residue
neighbours

Figure 5. Schematic diagram of adaptive restoration system.

```
1. Estimate $\hat{\Lambda}$ and $\hat{\sigma}^2(T)$ . Set the initial
   temperature $T=T_0$.
2. If convergent, go to Step 9.
3. Detect SPs and mark blocks around the SPs.
4. Choose a site $s$ in the marked block at
   random
5. Generate a small complex number $\Delta z$ at
   random, update the current site to

    $z_s(k+1) = z_s(k) + \Delta z$

      where $k$ denotes time step.

6. Calculate the energy $E_\varepsilon(z_s)$ before and after
   the update as:

    $E_\varepsilon(z_s;k) = \|z_s(k) - \hat{z}_s(k)\|^2$

    $E_\varepsilon(z_s;k+1) = \|z_s(k+1) - \hat{z}_s(k+1)\|^2$

      If $\Delta E_\varepsilon(z_s)$ ($\equiv E_\varepsilon(z_s;k+1) - E_\varepsilon(z_s;k)$) $<0$,
      accept the update and go to Step 8.

7. Calculate transition probability

    $P = e^{\frac{-\Delta E_\varepsilon(z_s)}{T}}$

   If a generated random value ([0,1])<$P$, then
   accept the update, otherwise reject it.
8. Decrease $T$ and go to Step 2.
9. Stop
```

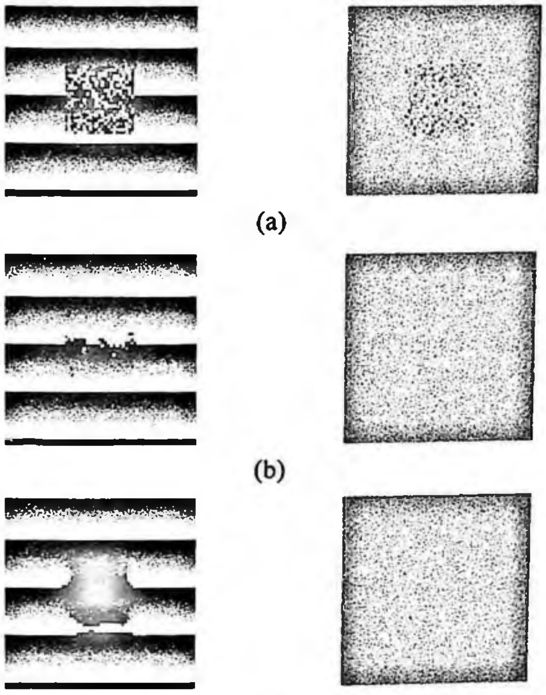
Figure 6. Detail of Monte Carlo Metropolis Algorithm for InSAR image restoration.

(MM) algorithm. In the process, we choose a corrupted pixel randomly and update it by adding (or subtracting) a small (complex) random value. If the update brings the system to a lower energy state, then we accept the update. But if the energy increases, we accept it with some probability. We iterate the update process until a certain convergence level is achieved. In the cycle of the restoration algorithm, some SPs are eliminated due to combination of –1 and +1 SPs or it disappears by itself. The detail of the algorithm is shown in Fig.6.

In the experiment, firstly we use a simulated phase image. The phase image represents a wrapped linear slope in vertical direction. We assume homogeneous amplitude (unity). An area in the complex image is then multiplied by zero mean complex Gaussian multiplicative noise (variance=0.5). The effect of complex multiplicative noise is then multiplicative in the amplitude, while additive in the phase inevitably.

Figure 7 (a) shows the simulated phase with noise (left part) and its corresponding SP distribution (right part). The number of detected SP is 115. We apply the MM algorithm where the initial chosen temperature T_0 = 0.26 and final temperature T_{final} = 0.00026. The detected current temperature T_{current}=0.11, is equal to the estimated variance ($\hat{\sigma}^2(T)$). The system is evolved for 7500 cycles. The restored phase image is displayed in Fig. 7 (b). For a comparison, a restoration/ filtering result using complex boxcar filter is provided in Fig 7 (c).

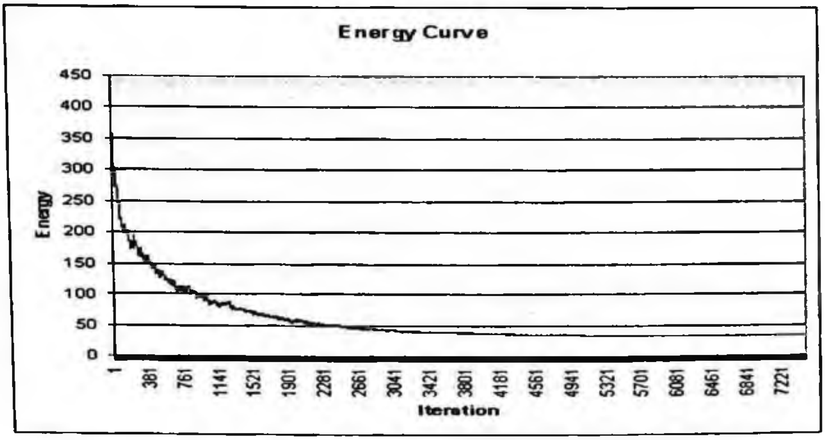
We observe that the result of the MM algorithm is better than the boxcar filter. Because of the averaging process in the boxcar filter, areas that contains noise is smeared out such that important fringe detail is lost as it is observed in Fig. 7. (c). On the other hand, our method restores the fringe detail as well as slopes in the corrupted areas (Fig.7 (b)). Additionally, while boxcar method reduces the SPs from 115 to 4, our method eliminates all the SPs.



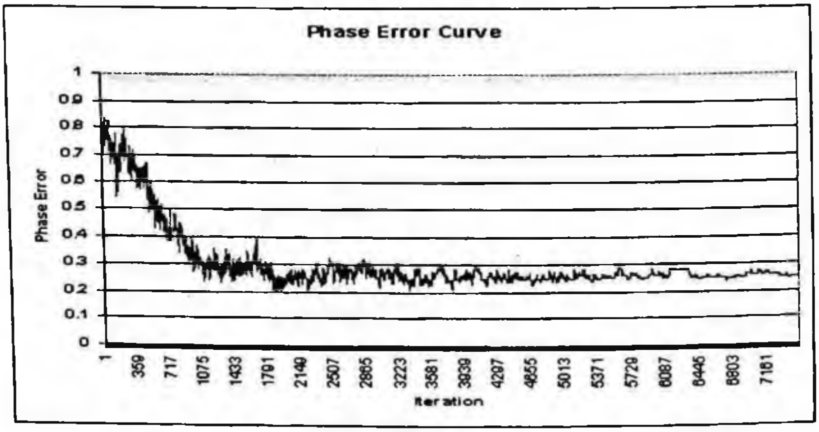
(c)

Figure 7. Simulated phase image (left part) and its corresponding SP distribution (right part): (a) original simulated phase has been degraded by noise (SP=115), (b) restoration by the proposed method (SP=0), and (c) restoration by complex boxcar method (SP=4).

Figure 8 displays the evolution of energy (a) and phase error (b). The phase error is calculated as the averaged absolute value of the phase difference between the estimated phase image and the original noiseless image. We observe that the energy as well as the phase error decreases monotonically. In both of the curves, we observe a high fluctuation in the initial iterations and it fades away afterwards. The fluctuation is due to configuration changes of the image in the corrupted area. This behavior is in concordance with the MM nature that allows transition to a higher energy in the early iterations (high *T*) and reduces such transition at latter iteration (low *T*).



(a)



(b)

Figure 8. Curves of (a) energy and (b) phase error as a function of iteration number.

4.3 CMRF Filtering by Steepest Descent Method

In the steepest descent method, after the parameters are estimated, the value of pixel with masks are updated as follows

$$z_s(k+1)=z_s(k)+\mu\Delta z_s(k) \tag{21}$$

$$\Delta z_s(k)=z_s(k)-\hat{\Lambda}^*(k)Q_s(k) \tag{22}$$

where μ is the learning constant ($0<\mu<1$). This algorithm decreases the energy exponentially. The process is repeated by feeding the estimation output of the network to the input recurrently, until a certain amount of SPs number is achieved.

In the experiment, we use an unfiltered phase difference image. We compare our result with those of the Lee filter's (Lee et.al. 1998) and the Goldstein-Werner (G-W) filter's (Goldstein and Werner 1998). Figure 9 (a) shows the unfiltered original image: (i) amplitude, (ii) phase and (iii) SP map. Before the processing, 540 SPs are detected. Firstly we apply the Lee filter to the original image. The result is depicted in Fig. 9 (b): (i) coherence map, (ii) filtered phase, and (iii) SP map. It is found that the SP number is reduced to 242. Secondly, we apply the G-W filter. The result is shown in Fig. 9 (c): (i) phase image and (ii) SP map. The SP is reduced to 157 after a strong filtering (by setting filter parameter α=1, (Goldstein and Werner 1998)). At last, we apply our proposed method. The result is depicted in Fig. 6(d): (i) amplitude, (ii) phase and (iii) SP map. It is found that the SP number has been reduced to 63.

It is observed that the proposed method not only reduce the SP more than others, but also give a better phase image result as follows. In the middle area of the Lee-filtered image, we observe a smearing effect that join fringes together and erase some detail of objects. The similar behaviour is also found in the G-W filtered: the smearing effect occurs in the left and right parts of the image. In contrast, the propose method gives a lower smearing effect to the fringe, so that the detail is more preserved.

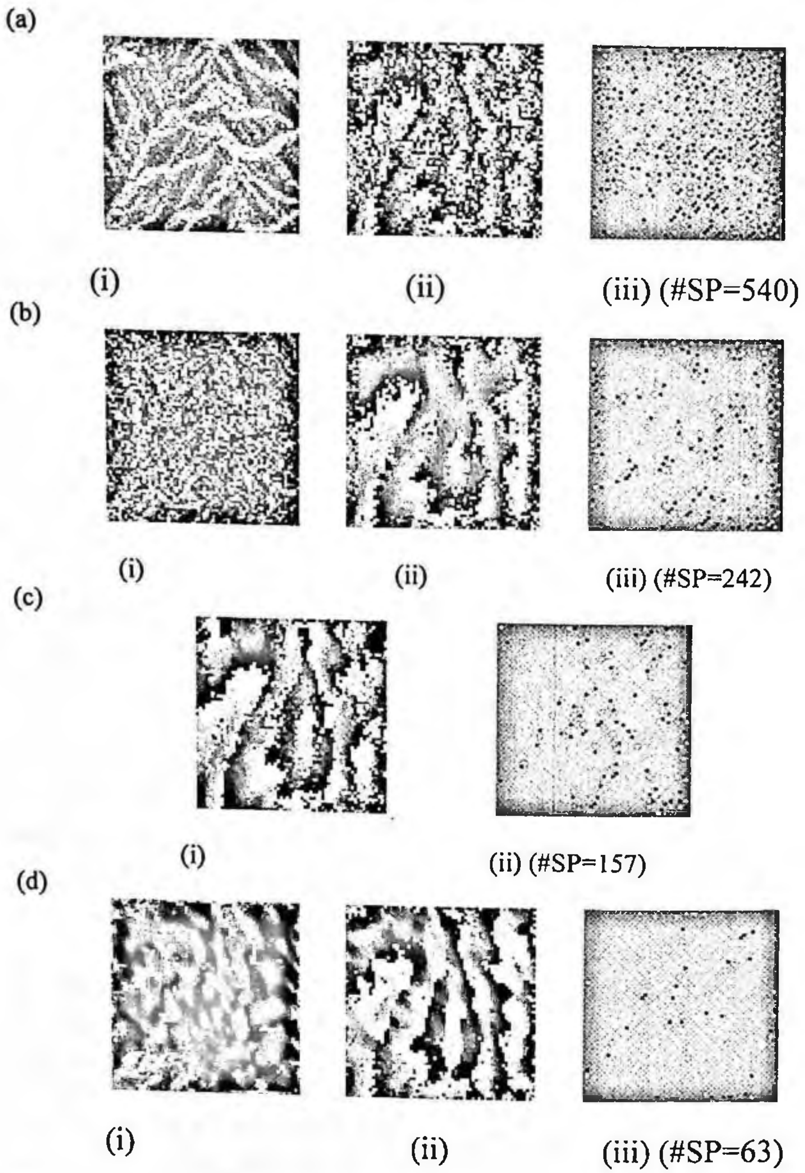


Figure 9. Performance comparison of the proposed method with the Lee and G-W filters: (a) unfiltered ((i) amplitude, (ii) phase, (iii) SP map), (b) Lee filtered ((i) coherence, (ii) phase, (iii) SP map), (c) G-W filtered ((i) phase, (ii) SP map), and (d) filtered by proposed method ((i) amplitude, (ii) phase, (iii) SP map).

4.4 Application to Phase Unwrapping

In this section we demonstrate the effectiveness of SP noise reduction method in improving PU result of a real InSAR image. We unwrap the phase by using MST (Minimum Shortest Path) branch cut method (Chen and Zebker 2000). Figure 10 (a1) shows phase images of an InSAR data captured at a region around the top of Mt. Fuji in Japan (Shimada 2000), each of them is 512×512 pixel size. Firstly the SPs are detected. We find 611 SPs in the original phase image. In the next step, we determine the MST that connects those SPs. Because there are so many numbers of SPs, we perform the MST block by block. Then these MSTs are merged together. The result is depicted in Fig.10 (a2). At last, unwrapping process is performed using flood fill process. The result is shown in Fig.10 (a3).

As a comparison, we firstly apply our restoration process before performing the phase unwrapping. By using the steepest descent method (the Monte Carlo Metropolis method will work in a similar fashion), we are able to reduce the number of SPs to 100 (around 83% reduction). The filtered phase is shown in Fig.10 (b1). The SP reduction gives a great advantage in the phase unwrapping process. Furthermore, the less number of SPs will theoretically reduce our computation complexity of unwrapping by about 8.5 times. The MST of the filtered phase is shown in Fig.10 (b2), while the unwrapping result is shown in (b3).

It is observed from Fig.10 (b2) that the filtered phase has fewer branches in the MST than the original one in (a2). This means that more areas can be unwrapped successfully. Observation in the unwrapped phase in (a3) and (b3) justifies our analysis. That is to say, the mountain top region of the raw phase data has dense SPs and, hence, the unwrapped phase in (a3) includes unnatural height variation. On the other hand, (b3) using our proposal shows a more detailed mountain shape. As a result, our proposal is found very useful in phase unwrapping.

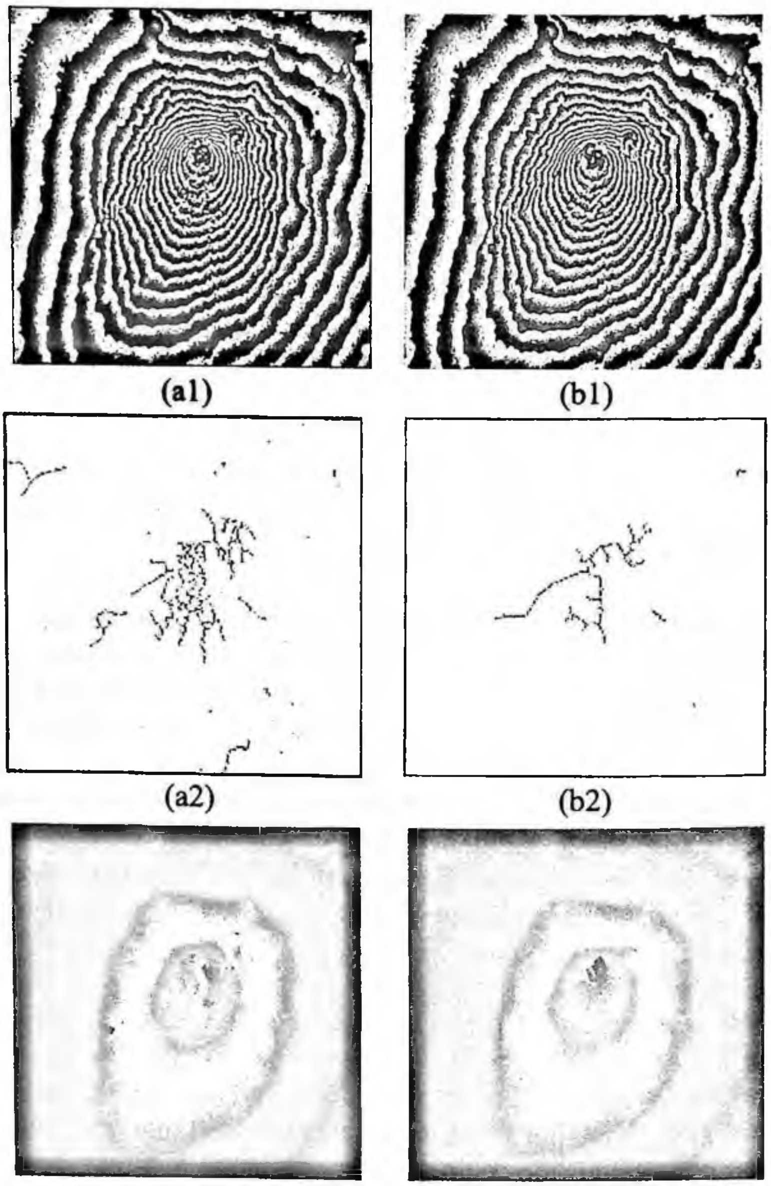


Figure 10. Applicability of the proposed method in improving the PU result: (a1) original InSAR phase image, (a2) cut-lines map of (a1), (a3) PU result without filtering, (b1) phase image after SP filtering, (b2) new cut-lines map and (b3) new PU result. (See also Cover Illustration of the book.)

5 Conclusion

New adaptive phase noise reduction/ image restoration methods of InSAR image have been described. The novelty lies in the treatment of the InSAR image as a complex-valued image, the usage of newly proposed CMRF model embedded in complex-valued neural network, and SP based adaptive mask. The proposed method employing the Monte Carlo Metropolis algorithm as well as the steepest descent method have successfully reduced the number of SPs, while maintained geometric information. We have also applied it to solve phase unwrapping problem. In combination with the MST branch cut phase unwrapping algorithm, it has been shown that the proposed method gives a better unwrapping result.

References

Chelappa; R., Manjunath; B.S., and Sinchony; T. (1991), "Texture segmentation with neural networks," in *Neural Network for Signal Processing*, Ed. B. Kosko, Prentice Hall, Englewood Cliffs, NJ, pp. 37-89.

Chen, C.W. and Zebker, H.A. (2000), " Network approaches to two-dimensional phase unwrapping: intractibility and two new algorithms," *J. Opt. Soc. America A*, vol. 17, pp. 401-414.

Cross, G.R. and Jain, A.K. (1983), " Markov random field texture models," *IEEE. Trans. Patt. Anal. & Machine Intel.*, vol. PAMI-5, pp. 25-39.

Geman, S. and Geman, D. (1984), "Stochastic relaxation, Gibbs distributions and Bayesian restoration of images," *IEEE. Trans. Pattern Anal. Machine Intel.*, vol. PAMI-6, pp. 721-741.

Ghiglia, D.C. and Romero, L.A. (1996), “Minimum L^p-norm two-dimensional phase unwrapping,” *J. Optical Soc. of America: A*, vol.13, pp. 1999-2013.

Goldstein, R.M. and Werner, C.L. (1998), “Radar interferogram filtering for geophysical applications,” *Geophysical Research Letters*, vol. 25, pp. 4035-4038.

Goldstein; R.M., Zebker; H.A., and Werner, C.L. (1988), ”Satellite radar interferometry: two-dimensional phase unwrapping,” *Radio Science*, vol. 23, pp. 713-720.

Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation,* New York, Macmillan.

Hirose, A. (1992), “Dynamics of fully complex-valued neural network,” *Electronics Letter*, vol. 28, pp. 1492-1593.

Hirose, A. (1999), “Coherent neural networks and their applications to control and signal processing,” in *Soft Computing in Systems and Control Technology*, S.G. Tzafestas, Ed., World Scientific, pp. 397-422.

Hirose, A. and Sugiyama, K. (1998), “A radar system with phase-sensitive milimetric wave circuitry and complex-amplitude neural processing,” *Proc. 8th ICANN 1998*, pp. 707-712.

Lee; J-S., Papathanassiou; K.P., Ainsworth; T.L., Grunes; M.R., and Reigber; A. (1998), “A new technique for phase noise filtering of SAR interferometric phase images,” *IEEE Trans. Geosci. and Remote Sensing*, vol.36, pp. 1456-1465.

Shimada, M. and Hirosawa, H. (2000), ” Slope corrections to normalized RCS using SAR interferometry,” *IEEE Trans. Geosci. and Remote Sensing*, vol. 38, pp. 1479-1484.

Suksmono, A.B. and Hirose, A. (2000), "Adaptive complex-amplitude texture classifier that deals with both height and reflectance for interferometric SAR images", *IEICE Trans. on Electronics,* vol. E83-C, pp. 1905-1911.

Suksmono, A.B. and Hirose, A. (2002), "Interferometric SAR image restoration using Monte-Carlo Metropolis method", *IEEE Trans. on Signal Processing*, vol. 50, pp. 290-298.

Suksmono, A.B. and Hirose, A. (2002), "Adaptive noise reduction of InSAR image based on Complex-MRF model and its application to phase unwrapping problem", *IEEE Trans. on Geoscience and Remote Sensing*, vol. 40, pp. 699-709.

Pritt, M. D. and Shipman, J. S. (1994), "Least-squares two-dimensional phase unwrapping using FFT's," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 32, pp. 706-708.

Takajo, H. and Takahashi, T. (1998), "Noniterative method for obtaining the exact solution for the normal equation in least-squares phase estimation from phase difference," *J. Optical Soc. of America: A*, vol. 5, pp. 1818-1827.

Zebker, H.A. and Lu, Y. (1998), "Phase unwrapping algorithms for radar interferometry: SP cut, least square, and synthesis algorithms," *J. Optical Soc. of America: A*, vol. 15, pp. 586-598.

Authors' address

Andriyan Bayu Suksmono: Dept. of Electrical Engineering, Institut Teknologi Bandung, Jl. Ganesha No.10, Bandung 40132, Indonesia.
Email: suksmono@yahoo.com, suksmono@radar.ee.itb.ac.id

Akira Hirose: Dept. Frontier Informatics, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan.
Email: ahirose@ee.t.u-tokyo.ac.jp

Chapter 14

Complex Neural Network Model with Analogy to Self-Oscillation Generated in an Optical Phase-Conjugate Resonator

Mitsuo Takeda and Takaaki Kishigami

This chapter reviews our early papers on a complex phase-conjugate neural network model with a Hopfield-like energy function. The complex neural network of the proposed model can change both the amplitude and the phase, and their dynamics has a close analogy to the dynamics of self-oscillation generated in an optical phase-conjugate resonator. It is shown that the optical gain medium should have a phase conjugate property in order for the generated complex optical fields to have an energy function that decreases monotonically with the time evolution of the fields. The results of experiments and computer simulations are presented that demonstrate the behaviors of the complex neural fields predicted by the theory.

1 Introduction

Optical implementation of neural networks attracted much interest in the years from late1980's to early 1990's (see, for example, Denz 1998). The purpose of this chapter is to review our early papers (Takeda and Kishigami 1992, 1993) on complex neural fields that have a Hopfield-like energy function and an analogy to optical fields generated in an optical phase-conjugate resonator. The work was motivated from the all-optical implementation of neural networks, in which the states of neurons are represented by coherent optical fields (Anderson 1986, Soffer *et al.* 1986, Yariv and Kwon 1986), rather than by the nonnegative intensities of the optical fields as in the incoherent optical implementation based on opto-electronic hybridization (Psaltis and Farhat 1985). In the coherent optical implementation, the states of neurons take complex values whose amplitude and phase correspond to those of the optical fields. The same is true with the synaptic weights that are represented by complex amplitude transmittance or reflectance (Psaltis *et al.* 1988). That both the states of neurons and the synaptic weights can take complex values is one of the most important features of coherent optical implementations that distinguish themselves from other implementations such as those based on VLSI technology. At that time, however, this fact did not seem to have attracted much attention of researchers working on optical implementations of neural networks, with only a few exceptions (see, for example, Little *et al.* 1990, Hirose 1992).

Noest proposed a model for complex neural networks, which he referred to as *phasor neural networks*, and studied their characteristics as an associative memory (Noest 1988). He has shown that a Hopfield-like energy function exists when synaptic weights of the network have the form of a Hermitian matrix $T_{mn} = T^*_{nm}$, where * denotes a complex conjugate. Noest's work was a very significant first step since it clarified the importance of complex neural networks, and pointed out their relevance to optical computing, such as the resonator memories demonstrated by Anderson and Erie (Anderson and Erie 1987). However, his model of phasor neurons is not suitable for coherent optical implementations, because it does not incorporate amplitude variation, which is one of the most fundamental physical characteristics of actual optical fields building up inside an optical cavity with a gain medium. The purpose of our paper was to propose a more general model that allows the change of both amplitude and phase, and whose dynamics has a closer analogy with that of self-oscillation generated by degenerate four-wave mixing. Whereas Noest dealt with a model of networks with Hermite conjugate synapses $T_{mn} = T_{nm}^{*}$, we proposed an alternative model of networks that have symmetric synapses $T_{mn} = T_{nm}$ and neurons with phase-conjugate gain. We first review the two models and point out that there exists a kind of dual relation between these two models. Next, we give some physical interpretations to our model in terms of optical physics, and show that the optical gain medium should have a phase conjugate property in order for the generated complex optical fields to have an energy function that decreases monotonically with the time evolution of the fields. We also show that, in the weak-field limit, the energy function can be physically interpreted as being proportional to the negative-signed time derivative of the optical power of the self-oscillating beam; this means that the energy function is a quantity *observable* by experiments. After presenting some examples of computer simulations that demonstrate typical behaviors of the complex neural fields predicted by the theory, we experimentally demonstrate the corresponding behaviors of the complex neural fields by using a $BaTiO_3$ crystal as a phase-conjugate medium. We also show by experiments that, in the weak-field regions, the energy function of the complex optical neural fields decreases as the oscillation builds up inside the phase-conjugate cavity. To our knowledge, this was the first prediction and experimental demonstration of the Hopfield-like energy function of the complex optical neural fields generated by phase-conjugate gain and feedback.

2 Energy Function for a Complex Neural Network

We propose two different models of complex neural networks that are composed of discrete neurons with feedback.

2.1 Phase-Preserving Neurons with Hermite Symmetric Synapses

Let us first consider a Noest-type complex neural network in which neurons change their states according to the following equations of dynamics:

$$\tau \frac{du_n}{dt} = -\alpha u_n + \sum_m T_{nm} V_m \ , \tag{1}$$

$$V_n = g(|u_n|)u_n/|u_n| \ , \tag{2}$$

where V_n and u_n denote, respectively, complex external and internal states of neurons, and T_{nm} is a complex synaptic weight for the complex signal flowing from neuron m to neuron n; τ and α are constant real parameters, and $g(\bullet)$ is a nondecreasing nonnegative real function. Equation (2) states that, while the amplitude u_n of the internal state is transformed by the nondecreasing function $g(\bullet)$ as in conventional real networks, the value of the phase is preserved; we term such neurons as *phase-preserving neurons* (PPNs). Note that the model includes the Noest's phasor neuron model (Noest 1988) as a special case; it can be obtained by putting $g(\bullet)=1$ and $\alpha = 0$. We can show that, when the synapses have Hermitian symmetry $T_{mn} = T^*_{nm}$, these phase-preserving neurons change their states in such a manner that the Noest-like energy function defined by

$$E = -\frac{1}{2}\sum_n \sum_m T_{nm} V_n^* V_m + \alpha \sum_n \int_0^{|V_n|} g^{-1}(s)ds \tag{3}$$

reduces its value monotonically with the time evolution of the system. In Eq.(3), $g^{-1}(\bullet)$ is an inverse function of $g(\bullet)$, and s is a real parameter for integration. Note that due to Hermitian symmetry the energy function E becomes a real function. The proof of the energy minimization characteristic of this phase-preserving neural network is given in the original paper (Takeda and Kishigami 1992).

2.2 Phase-Conjugate Neurons with Symmetric Synapses

Here we propose an alternative model for a complex neural network that has *phase-conjugate neurons* (PCNs) connected by symmetrical synapses $T_{mn} = T_{nm}$. Equations of dynamics for this model are given by

$$\tau \frac{du_n}{dt} = -\alpha u_n + \sum_m T_{nm} V_m \ , \tag{4}$$

$$V_n = g(|u_n|) u_n^* / |u_n| \ . \tag{5}$$

The only, but important, difference from the previous model is that internal states are transformed into external states with the sign of their phases reversed (or phase-conjugated) as seen in Eq.(5); we therefore term such neurons as *phase-conjugate neurons* (PCNs). If the complex synaptic weights have symmetry $T_{mn} = T_{nm}$, the network has a Hopfield-like energy function (Hopfield 1984)

$$E = -\frac{1}{2}\mathrm{Re}\left(\sum_n \sum_m T_{nm} V_n V_m\right) + \alpha \sum_n \int_0^{|V_n|} g^{-1}(s)ds \ , \tag{6}$$

which reduces its value monotonically with the time evolution of the system; Re denotes real part, and the proof is given in the original paper (Takeda and Kishigami 1992).

3 Analogy and Physical Interpretation

In this section, we point out that our PCN model for complex neural networks bears an interesting analogy with certain physical systems with phase conjugating gain and optical feedback. Let us consider an optical system as shown in Figure 1, where self-oscillation is

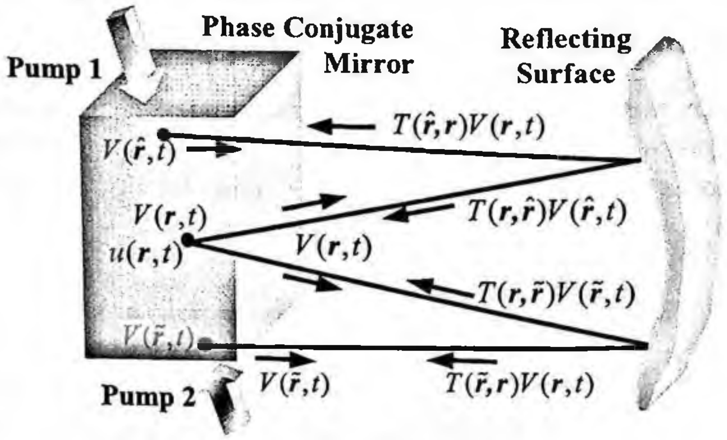


Figure 1. Analogy of the PCN model with the dynamics of optical field generated in a phase-conjugate resonator.

generated inside the cavity formed by a reflector and a phase conjugate mirror (PCM) with gain. Energy for the gain is provided by pump beams through degenerate four-wave mixing; self-oscillation in such a system was predicted by Yariv and Pepper (Yariv and Pepper 1977) and was experimentally demonstrated by Feinberg and Hellwarth (Feinberg and Hellwarth 1989). Actual dynamics of the optical fields in such a physical system may be too complex to be expressed by simple equations of motion. However, our aim here is to find an analogy between the PCN model and the physical system with phase conjugate gain and feedback, rather than to discuss the detailed physics of such a system. For this purpose, we first show that a somewhat fictitious yet plausible model for the dynamics of the optical fields inside the cavity can be derived from our PCN model. Based on this quasi-physical model for the dynamics of the complex optical fields, we then give some physical interpretations to the PCN model and clarify the role played by the phase conjugating neurons.

3.1 Dynamics of Complex Neural Fields

The equations of dynamics, Eqs.(4) and (5), for spatially discrete neurons can easily be modified to describe the dynamics of spatially continuous neural fields:

$$\tau \frac{\partial u(\boldsymbol{r},t)}{\partial t} = -\alpha u(\boldsymbol{r},t) + \int_{-\infty}^{\infty} T(\boldsymbol{r},\hat{\boldsymbol{r}}) V(\hat{\boldsymbol{r}},t) d\hat{\boldsymbol{r}} \ , \tag{7}$$

$$V(\boldsymbol{r},t) = g(|u(\boldsymbol{r},t)|) u^*(\boldsymbol{r},t) / |u(\boldsymbol{r},t)| \ . \tag{8}$$

Likewise, we can show that these complex neural fields change their states in such a manner that the energy function defined by

$$\begin{aligned} E = -\frac{1}{2} \mathrm{Re}\left(\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} T(\boldsymbol{r},\hat{\boldsymbol{r}}) V(\boldsymbol{r},t) V(\hat{\boldsymbol{r}}) d\boldsymbol{r} d\hat{\boldsymbol{r}} \right) \\ + \alpha \int_{-\infty}^{\infty} \int_{0}^{|V(\boldsymbol{r},t)|} g^{-1}(s) ds d\boldsymbol{r} \end{aligned} \tag{9}$$

reduces its value monotonically with the time evolution of the system. Referring to Figure 1, we interpret the state of a complex neuron $V(\boldsymbol{r},t)$ as the complex amplitude of the optical field emitted from a point at $\boldsymbol{r}$ in the crystal. Similarly, we interpret the internal state of the complex neuron $u(\boldsymbol{r},t)$ as the complex amplitude of the grating formed at $\boldsymbol{r}$ by the interference between one of the pump beams and the beams that originate from other sources $V(\hat{\boldsymbol{r}},t)$ at points $\hat{\boldsymbol{r}}$ in the crystal and that reach the point at $\boldsymbol{r}$ through reflections at the reflecting surface. We interpret the complex synaptic weight $T(\boldsymbol{r},\hat{\boldsymbol{r}})$ as a transmission function that describes the propagation of the beam from the point at $\hat{\boldsymbol{r}}$ to the point at $\boldsymbol{r}$. Based on these interpretations, Eq.(7) may be considered to represent a writing process of the grating where the complex grating amplitude $u(\boldsymbol{r},t)$ increases in proportion to the sum of the complex fields of the writing beams $\int_{-\infty}^{\infty} T(\boldsymbol{r},\hat{\boldsymbol{r}})V(\hat{\boldsymbol{r}},t)d\hat{\boldsymbol{r}}$. The decay term $-\alpha u(\boldsymbol{r},t)$ may be considered to express a rate of erasing that the grating undergoes during the writing process. Likewise, Eq.(8) may be considered to represent a read-out process where we read out a beam that is phase-conjugated to the writing beam and whose amplitude is transformed by a nondecreasing real function $g(\bullet)$. We may incorporate the effects of gain saturation and/or thresholding into this amplitude transform function. These two equations of dynamics, Eqs.(7) and (8), together form a set of simultaneous equations, so that they may reflect the fact that both the writing and reading processes occur simultaneously in the degenerate four-wave mixing.

3.2 Symmetric Synaptic Weights and Helmholtz's Reciprocity Theorem

Since the complex synaptic weight $T(\boldsymbol{r},\hat{\boldsymbol{r}})$ has been interpreted as a transmission function of the optical field, we may expect that the vast majority of optical feedback systems have symmetric synapses because of the reciprocity theorem of Helmholtz $T(\boldsymbol{r},\hat{\boldsymbol{r}})=T(\hat{\boldsymbol{r}},\boldsymbol{r})$ (Born and Wolf 1970). This in turn means that the PPN model, which includes Noest's model as a special case, is less suitable for optical implementation because its synaptic weights need to have

Hermitian symmetry, a condition that is generally not satisfied by physical law of wave propagation.

3.3 Role of Phase Conjugating Neurons

We are now ready to consider the role played by phase conjugating neurons in the PCN model. Suppose the phase conjugation mechanism were not necessary and that the phase-conjugate neurons could be replaced by ordinary phase-preserving neurons. Then any optical feedback systems with gain should have an energy function since we have already seen that the condition of symmetric synaptic weights is automatically fulfilled by the Helmohltz's reciprocity theorem. Because of their energy minimization characteristic, optical fields inside any such optical feedback systems with sufficiently large gain should always converge into some stably oscillating modes, irrespective of the shape of the reflecting surface. Obviously, this is not what we usually experience with laser oscillations, where stable oscillations are generated only when the reflectors form a stable cavity. Now the role played by the phase conjugating neurons in the PCN model has become clear. Their role can be physically interpreted as making the cavity stable in order to guarantee the convergence of the optical fields into a stable mode. The fact that a phase conjugating mirror can always form a stable cavity is well known (Yariv 1985), and Feinberg and Hellwarth's demonstration of a kitchen spatula laser (Feinberg and MacDonald 1989) is famous. To our knowledge, however, it has never been discussed from the viewpoint of its relation to the existence of the Hopfield-like energy function for the optical fields inside the cavity.

3.4 Energy Function and Total Intensity of Oscillating Fields

It is of interest to examine if we can observe the predicted energy function of the complex optical neural fields, Eq.(9), by experiments. It is generally not possible to specify the synaptic weights $T(\boldsymbol{r},\hat{\boldsymbol{r}})$ for all the possible optical ray paths between the distributed neurons. We, therefore, eliminate them by substituting Eqs.(7) and (8) into Eq.(9), and obtain (Takeda and Kishgami 1992)

$$E=-\frac{1}{2}\int_{-\infty}^{\infty}\Big\{|V(\boldsymbol{r},t)|\left[\tau\partial g^{-1}(|V(\boldsymbol{r},t)|)/\partial t+\alpha g^{-1}(|V(\boldsymbol{r},t)|)\right] \\ -2\alpha\int_{0}^{|V(\boldsymbol{r},t)|}g^{-1}(s)ds\Big\}d\boldsymbol{r} \qquad (10)$$

This relates the energy function to the modulus of the field amplitude $V(\boldsymbol{r},t)$. In most cases where the oscillation grows rather slowly, both the field amplitude and the grating amplitude remain small for some time period after the start of oscillation, so that $|V(\boldsymbol{r},t)|\ll 1$, $|u(\boldsymbol{r},t)|\ll 1$ for $0\leq t\leq t_w$. In such a weak-field limit, we may expect that the gain function can be approximated by a linear function $g(|V(\boldsymbol{r},t)|)\approx a|V(\boldsymbol{r},t)|$, and we have

$$g^{-1}(|V(\boldsymbol{r},t)|)\approx a^{-1}|V(\boldsymbol{r},t)| \ , \qquad (11)$$

where a is a positive constant. Substituting Eq.(11) into Eq.(10), we have

$$E=-\frac{\tau}{4a}\frac{dI}{dt} \ , \qquad (12)$$

where I is the total intensity or the power of the optical field defined by

$$I(t)=\int_{-\infty}^{\infty}|V(\boldsymbol{r},t)|^2 d\boldsymbol{r} \ . \qquad (13)$$

Thus we have shown that, in the weak-field limit, the energy function is proportional to the time derivative of the sign-reversed total intensity of the fields. In other words, the energy function (with its sign reversed) is proportional to the rate of the intensity growth of the oscillating beam. Since we have already shown that $dE/dt \le 0$, we have

$$\frac{d^2 I}{dt^2} = -\frac{4a}{\tau}\frac{dE}{dt} \ge 0 \quad , \tag{14}$$

which states that the total intensity $I(t)$ grows as a downward convex function of time. Finally we point out again that, since the total intensity of the oscillating beam $I(t)$ is a measurable quantity, the behavior of the energy function in the weak-field region can be observed by experiments.

3.5 Computer Simulations

We performed computer simulations to examine the behaviors of the complex neural fields and their energy function. A one-dimensional PCN model was adopted to save the time and memory for computation. Parameters are chosen so that the model bears an analogy to a physical system of an optical resonator formed by a concave parabolic mirror and a phase conjugate mirror (PCM) facing each other. We regard the optical fields leaving from the PCM toward the parabolic mirror as the complex neural fields $V(x,t)$, and make the distance between the two mirrors equal to the focal length f of the parabolic mirror. For such a system with the so-called f-f geometry, the synaptic weights take (aside from a constant factor) the form of a Fourier transform kernel $T(x,\hat{x}) = \exp(-2\pi i x\hat{x}/\lambda f)$ (with λ being the wavelength), so that the field emitted from the PCM as $V(\hat{x},t)$ and reflected by the parabolic mirror back to the PCM becomes the Fourier transform of the original field:

$$\int_{-\infty}^{\infty} T(x,\hat{x})V(\hat{x},t)d\hat{x} = \int_{-\infty}^{\infty} V(\hat{x},t)\exp\left(\frac{-2\pi i x\hat{x}}{\lambda f}\right)d\hat{x} \ . \qquad (15)$$

To incorporate the fact that the amplitude of the self-oscillating field saturates as it reaches full growth, we used a gain function of the form

$$g(x) = \tanh(x/T) \qquad (x \geq 0) \ , \qquad (16)$$

which is a positive part of a bipolar sigmoid function whose gain slope at $g(0) = 0$ is determined by the parameter T. Although our interest is in complex neural fields whose equations of dynamics are space-time continuous, we had to discretize the model to perform the simulations by digital computer. The one-dimensional neural fields $V(x,t)$ were represented by 501 complex discrete neurons arranged with equal separations to cover the effective PCM size of 0.72mm determined by the spot size of the pump beams. The gain parameter of these neurons was chosen as $T = 240$. The focal length of the parabolic mirror was $f = 500\text{mm}$, and the wavelength was $\lambda = 500\text{nm}$. The differential equations were approximated by difference equations with the parameters chosen as $\alpha = 1$ and $\tau = 2$. We changed the states of the neurons asynchronously in order to avoid an artifact that may manifest itself as the oscillation of the energy function when such a discrete-time approximation is made (Takeda and Goodman 1986); we later found that synchronous transition models also worked well, which allowed us to use an FFT algorithm for the calculation of Eq.(15). For the initial state, we used a weak random field whose intensity and phase distributions are shown in Figure 2(a). The neurons first change their states to have a smoother phase distribution as shown in Figure 2(b) and Figure 2(c). The graphs show the intensity distributions (left) and the phase distributions (right) after 1 time unit (b), and after 15 time units (c), where the time unit is defined as a period during which all the neurons renew their states. Physically, this

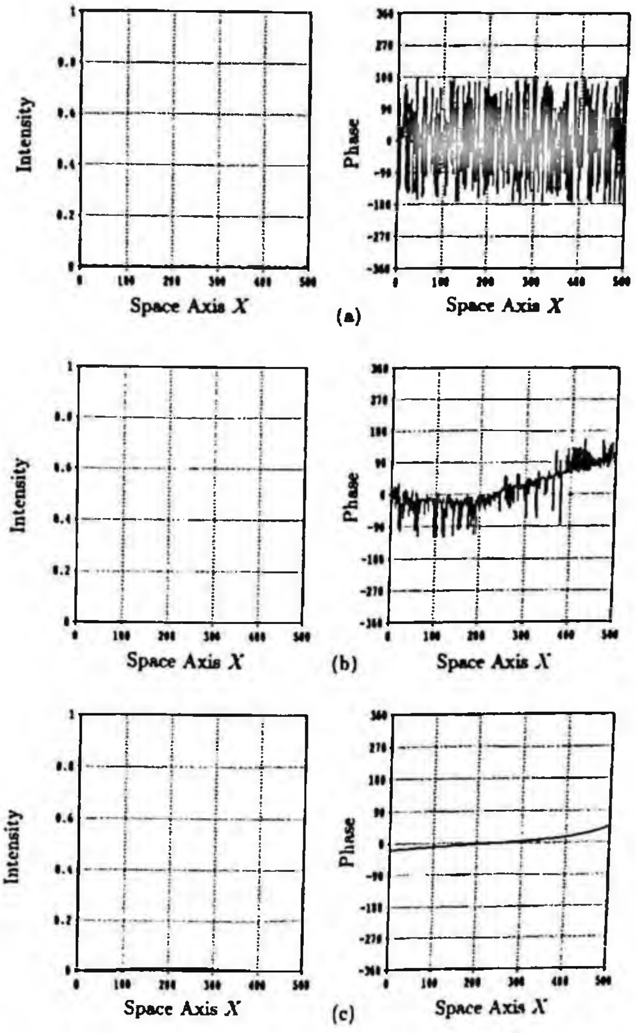


Figure 2. Examples of simulated complex neural fields that converge into a mode that looks like a fundamental Hermite-Gaussian mode. The graphs at left and right show the intensity distributions and the phase distributions, respectively, (a) at the initial state, (b) after 1 time unit, and (c) after 15 time units.

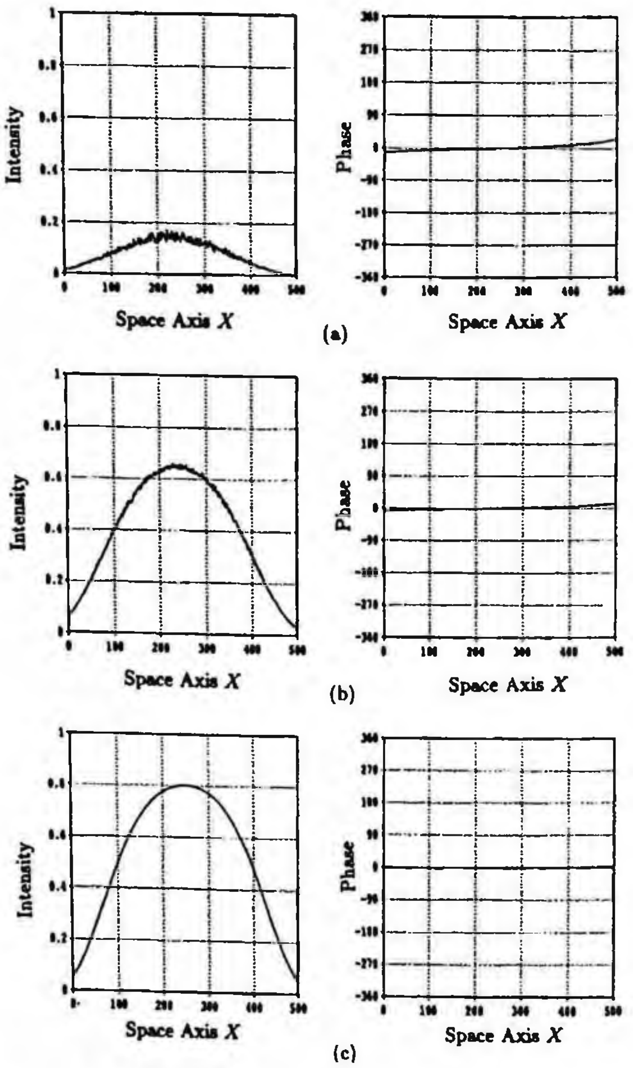


Figure 3. Examples of simulated complex neural fields that converge into a mode that looks like a fundamental Hermite-Gaussian mode. The graphs at left and right show the intensity distributions and the phase distributions, respectively, (a) after 20 time units,(b) after 26 time units, and (c) after 50 time units.

process of phase smoothing may be interpreted as being performed by spatial lowpass filtering. The Fourier transform kernel of the synaptic weights produces the spatial frequency spectrum distribution of the neural fields over the PCM from which they originated. Only very low spatial frequency components of these spectrum distribution will be returned as a phase conjugate beam because we have limited the effective size of the PCM. Since the spatial frequency spectrum distribution is most sensitive to the phase distribution of the original fields, we may consider it natural that the phase of the complex neural fields first tries to take a smoother spatial distribution. Once the smooth phase distribution is achieved, the fields become more and more concentrated onto the effective area of the PCM, and the intensity starts to grow.

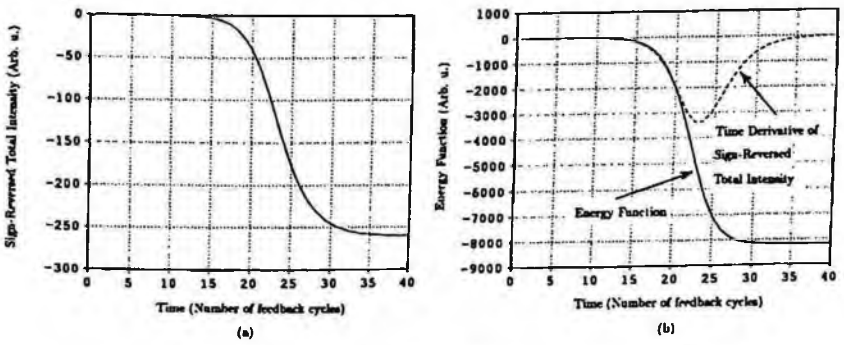


Figure 4. (a) Sign-reversed total intensity, (b) energy function (solid curve) and time derivative of the sign-reversed total intensity (dashed curve), exhibiting similar behavior in the weak signal region, which lasts until 20 time units.

The behavior of the simulated complex neural fields was found to be in agreement with that predicted from the physical picture described above; Figure 3 shows the intensity distributions (left) and the phase distributions (right) after 20 time units (a), after 26 time units (b), and after 50 time units (c). In this example, the fields converged into a mode that looks like a fundamental Hermite-Gaussian mode as shown in Figure 3(c). Illustrated in Figure 4(a) is

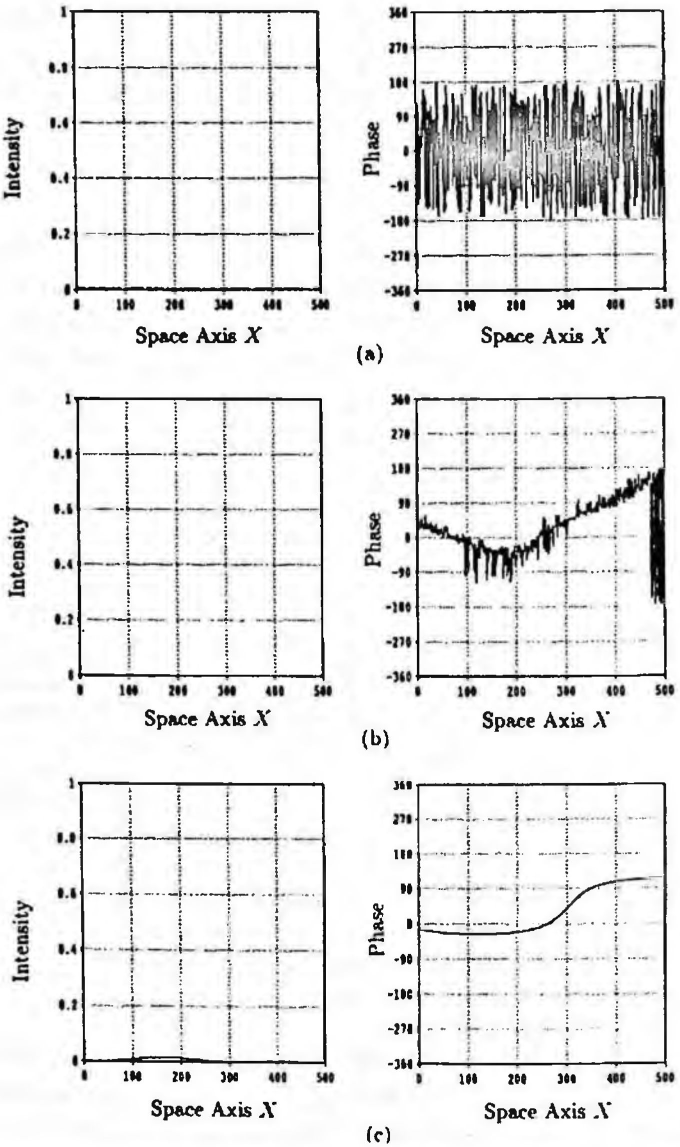


Figure 5. Examples of simulated complex neural fields that converge into a mode that looks like a higher-order Hermite-Gaussian mode. The graphs on the left and right show the intensity distributions and the phase distributions, respectively, (a) at the initial state, (b) after 1 time unit, and (c) after 15 time units.

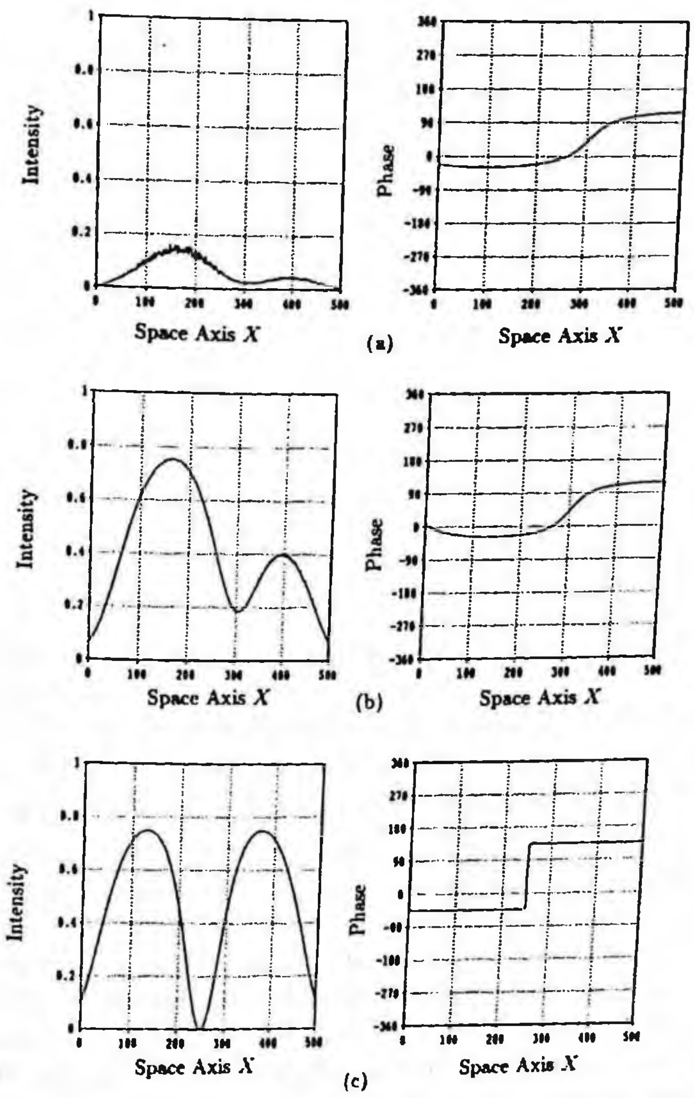


Figure 6. Examples of simulated complex neural fields that converge into a mode that looks like a higher-order Hermite-Gaussian mode. The graphs on the left and right show the intensity distributions and the phase distributions, respectively, (a) after 20 time units, (b) after 30 time units, and (c) after 500 time units.

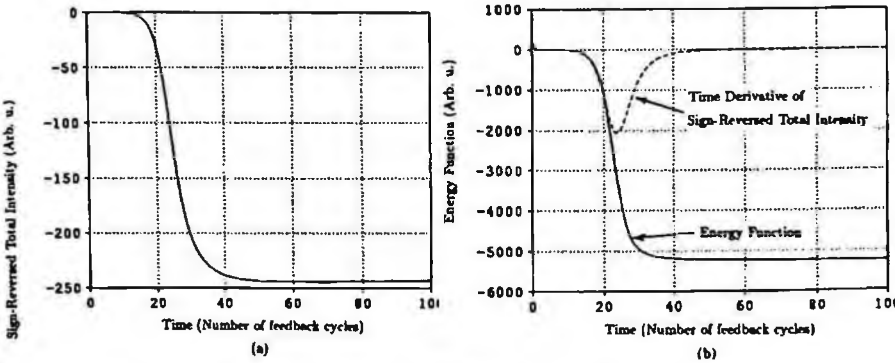


Figure 7. (a) Sign-reversed total intensity, (b) energy function (solid curve) and time derivative of the sign-reversed total intensity (dashed curve), exhibiting similar behavior in the weak signal region, which lasts until 20 time units.

the time variation of the (sign-reversed) total intensity, which starts to saturate at around 25 time units. In Figure 4(b), the solid line shows the energy function, which reduces its value monotonically as predicted by the theory. The broken line shows the time derivative of the sign-reversed total intensity $-dI(t)/dt$ scaled by a factor $\tau/4a$. Note that, as predicted by Eq.(12), it can well represent the energy function (apart from the constant scale factor) in the weak-field region that appears to last until 20 time units. This means that we can observe the behavior of the energy function in the weak-field region experimentally by detecting the total intensity of the fields and computing its sign-reversed time derivative. Figures 5 and 6 show another example of simulation where we increased the weak-field gain to $T=200$ and the effective size of the PCM to 0.90mm. In this example, the behavior of the phase distribution is of particular interest. First it tries to take a smoother distribution as in the previous example. Then it finally finds out a solution with a phase jump by π, and produces a higher-order mode with the sign-reversed twin peaks, as shown in Figure 6(c). As shown in Figure 7, the behaviors of the energy function and the total intensity look similar to the previous example. Thus our theoretical prediction again holds in this example where the neural fields converge into a higher-order mode.

4 Experiments

Based on the analogy that we have found between the complex neural fields of our PCN model and the optical fields generated in a phase conjugate resonator, we carried out experiments to observe the behavior of the energy function in a physical system as shown in Figure 8. The phase conjugate resonator is formed by combining a conventional mirror, M4, with a PCM that consists of a $BaTiO_3$ crystal and a pair of pump beams, pump1 and pump2, mutually counter propagating from mirrors M2 and M3. The optical power for the pump beams is supplied from an argon-ion laser operating at the wavelength of 514.5nm. Experiments of self-oscillation using such degenerate four-wave mixing have already been reported by many people. It should therefore be emphasized that our aim is not

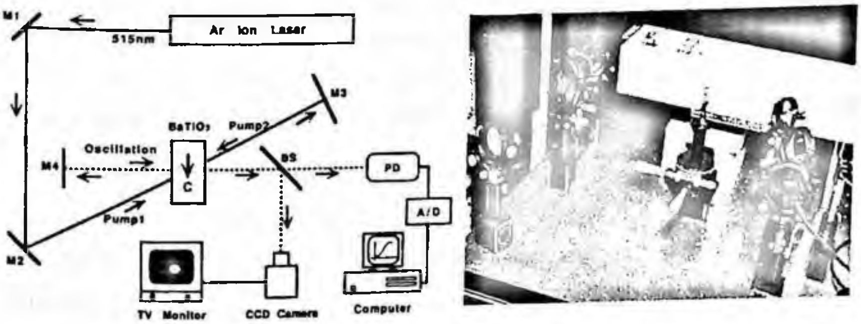


Figure 8. Experimental setup for observation of the Hopfield-like energy function of the optical fields generated in a phase-conjugate resonator: M1-M4, conventional mirrors; BS, beamsplitter; PD, photodetector; A/D, anlog-to-digital convertor.

to demonstrate the self-oscillation itself but to observe the predicted behaviors of the energy function that we have associated with the optical fields in the phase conjugate resonator. Likewise, studying the behavior of the total intensity of the oscillating beam may not be of interest by itself, since it may have also been done by many people. However, we consider that the significance of our experiments lies in that they are conducted with the understanding that the Hopfield-like energy function of the optical fields in the phase conjugate resonator can be observed through the measurement of the total intensity of the oscillating fields and the computation of its sign-reversed time derivative. Returning to Figure 8, we observe the field intensity distribution in the PCM by a CCD camera focused on the $BaTiO_3$ crystal, and take it into a frame memory for display and analysis. At the same time, we detect the total intensity of the oscillating field with a photo detector and take it into a personal computer to compute its sign-reversed time derivative. Strictly, what we are detecting in this experiment is not the total intensity of the neural fields that we defined by $I(t)=\int_{-\infty}^{\infty}|V(r,t)|^2dr$; remember that we have associated the neural fields with the optical fields leaving from the PCM toward the mirror M4. Instead of $I(t)$, we are detecting the total intensity of the fields propagating from the mirror M4 toward the PCM, which is given by $\hat{I}(t)=\int_{-\infty}^{\infty}\left|\int_{-\infty}^{\infty}T(r,\hat{r})V(\hat{r},t)d\hat{r}\right|^2dr$. Generally, the observed $\hat{I}(t)$ is not equal to $I(t)$, but in our experiment, where the transmission function $T(r,\hat{r})$ (representing the synaptic

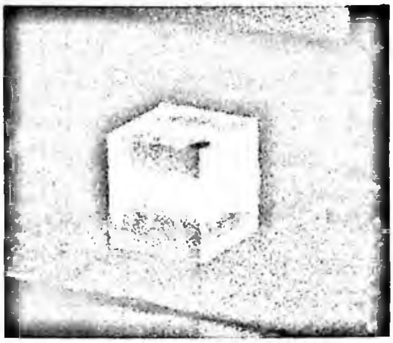


Figure 9. Complex neural fields generated in a $BaTiO_3$ crystal.

weights) has the form of the Fourier transform kernel, we can regard $\hat{I}(t)$ as being equal to $I(t)$ because of Parseval's theorem of the Fourier transform.

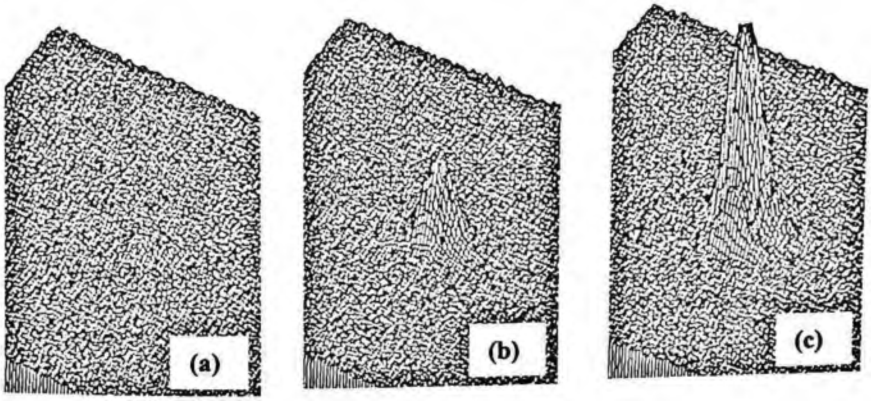


Figure 10. Intensity growth of optical fields in phase-conjugate resonator: (a) after 80s, (b) after 100s, and (c) after 150s.

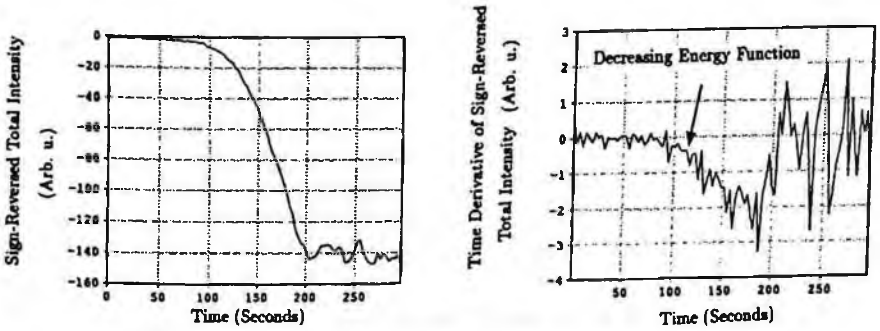


Figure 11. Sign-reversed total intensity of the optical fields (left) and its time derivative (right) representing the energy function that decreases monotonically in the weak-field region (ignore the noisy fluctuation enhanced by differentiation).

Figure 10 shows an example of the intensity growth of the optical fields observed by the CCD camera, which resembles the first example of the computer simulations depicted in Figures.2 and 3. In Figure 11, the left graph shows the sign-reversed total intensity of the optical fields, which resembles the result of simulations shown in Figure 4(a). Its time derivative is shown in the right graph in Figure 11. Ignoring the noisy fluctuations enhanced by differentiation, we can see that it decreases monotonically in the weak-field region which appears to last approximately until 150 seconds after the start of pumping. According to Eq.(12) and to the result of the computer simulation shown in Figure 4(b), we consider that the derivative of the sign-reversed total intensity shown in the right graph represents the Hopfield-like energy function of the optical fields in the region where the signal is small. Next, we increased the pump power to

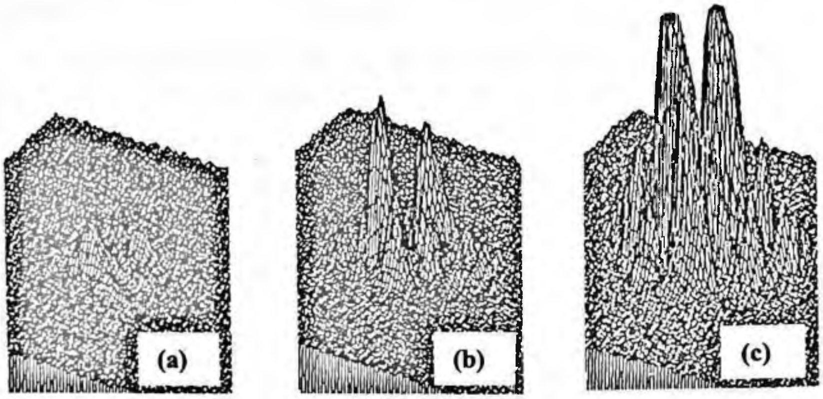


Figure 12. Intensity growth of optical fields in phase-conjugate resonator: (a) after 20s, (b) after 30s, and (c) after 45s.

realize the situation analogous to the second computer simulation where we increased the weak-field gain by choosing $T = 200$. Figure 12 shows an example of the intensity growth of the optical fields, which resembles the second example of the computer simulations depicted in Figures 5 and 6 though the mode of this experiment looks more complicated. The left graph in Figure 13 shows the sign-reversed total intensity of the optical fields which resembles the result of simulations shown in Figure 7(a). Its time derivative is shown in the right graph. Again, we can see that it decreases monotonically in the weak-field region. We consider that this derivative of the sign-reversed total intensity corresponds to that shown by the broken line in Figure 7(b) and represents the Hopfield-like energy function in the weak-field region which appears to last approximately until 40 seconds after the start of pumping.

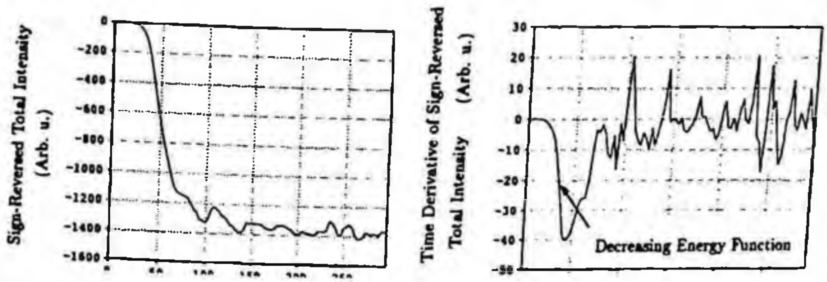


Figure 13. Sign-reversed total intensity of the optical fields (left) and its time derivative (right) representing the energy function that decreases monotonically in the weak-field region (ignore the noisy fluctuation enhanced by differentiation).

5 Conclusion

We have proposed a complex phase-conjugate neural network model that has a Hopfield-like energy function. We have pointed out that the dynamics of our complex PCN model has a close analogy with the dynamics of self-oscillation generated in a phase-conjugate resonator. From the physical interpretation of the model, we have found that the optical gain medium should have a phase conjugate property in order for the generated optical fields to have a

Hopfield-like energy function that decreases monotonically with the time evolution of the fields. We have shown that, in the weak-field limit, the energy function can be approximated by the time derivative of the sign-reversed total intensity of the fields, and is observable by experiments. We have conducted experiments and computer simulations, and demonstrated the behaviors of the complex neural fields predicted by the theory.

Finally, we should emphasize the importance of finding analogies between neural networks and physical systems. Historically, Hopfield proposed his neural network model and the concept of the energy function based on the analogy to the physics of spin glass (Hopfield 1982). And the analogy to statistical physics has motivated Kirkpatrick, Gelatt and Vecchi to think of simulated annealing (Kirkpatrick *et al.* 1983), and led Hinton and Sejnowski to the idea of the Boltzmann machine (Hinton and Sejnowski 1987). On the other hand, majority of researches in optical neuro-computing appear to have been focused on the technical issues of how to optically implement the existing neural network models and/or learning schemes. Our approach in this study is more like that of the people mentioned above, because our aim is to propose a new neural network model (the PCN model) based on the knowledge of (optical) physics, rather than to discuss the (optical) technology issues for the implementation of the existing neural network models. Possibilities and limitations of our complex PCN model in applications to information processing and/or storage are yet to be studied. Should we find some interesting applications, the optical implementation of the model would be quite easy and natural because the model was born from the analogy to optical physics.

References

Anderson, D. Z. (1986), "Coherent optical eigenstate memory," *Opt. Lett.* , vol.11, pp 56-58.

Anderson, D. Z. and Erie, M.C. (1987), "Resonator memories and novelty filters," *Opt. Eng.*, vol. 26, pp. 434-444.

Born, M. and Wolf, E. (1970), *Principle of Optics* , Pergamon Press, Oxford, p.381.

Denz, C. (1998), *Optical Neural Networks,* T. Tschudi Ed. Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesebaden.

Feinberg, J. and Hellwarth, R.W. (1989), "Phase-conjugating mirror with continuous-wave gain," *Opt. Lett.*, vol. 5, pp. 519-521.

Feinberg, J. and MacDonald, K. R. (1989), "Phase-conjugate mirrors and resonators with photorefractive materials," *Photorefractive Materials and Their Applications II*, Guenter, P. and Huignard, J.-P. Ed., Springer Berlin, p.154.

Hinton, G. E. and Sejnowski, T. J. (1987), "Learning and relearning in Boltzmann machines," *Parallel Distributed Processing,* vol. 1, Rumelhart, D. E. and McClelland, J. L. Ed., The MIT Press, Massachusetts, Chap.7.

Hirose, A. (1992), "Continuous complex-valued backpropagation learning," *Electron. Lett.*, vol.28, pp.1854-1855.

Hopfield, J.J. (1982), "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA,* vol. 79, pp. 2554-2558.

Hopfield, J.J. (1984), "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088-3092.

Kirkpatrick, S., Gelat, C.D., and Vecchi, M.P. (1983), "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680.

Little, G.R., Gustafson, S.C., and Senn, R.A., (1990), "Generalization of the backpropagation neural network learning algorithm to permit complex weights," *Appl. Opt.*, vol. 29, pp. 1591-1592.

Noest, A.J. (1988), "Phasor neural networks," *Neural Information Processing Systems*, Anderson, D.Z., Ed., American Institute of Physics, New York, pp. 584-591.

Noest, A.J. (1988), "Discrete-state phasor neural networks," *Phys. Rev. A*, vol. 38, pp. 2196-2199.

Noest, A.J. (1988), "Associative memory in sparse phasor neural neworks," *Europhys. Lett.,* vol. 6, pp. 469-474.

Psaltis, D. and Farhat, N. (1985), "Optical information processing based on an associative memory model of neural nets with thresholding and feedback," *Opt. Lett.*, vol. 10, pp. 98-100.

Psaltis, D., Brady, D., and Wagner, K. (1988), "Adaptive optical networks using photorefractive crystals," *Appl. Opt.,* vol. 27, pp. 1752-1759.

Soffer, B.H., Dunning, G.J., Owechko, Y., and Marom, E. (1986), "Associative holographic memory with feedback using phase-conjugate mirrors," Opt. Lett., vol. 11, pp. 118-120.

Takeda, M. and Goodman, J.W. (1986), "Neural networks for computation: number representations and programming complexity," *Appl. Opt.*, vol. 25, pp 3033-3046.

Takeda, M and Kishigami, T (1992), "Complex neural fields with a Hopfield-like energy function and an analogy to optical fields generated in phase-conjugate resonators," *Appl. Opt.*, vol. 9.

Takeda, M and Kishigami, T (1992), "Energy function of complex neural fields generated by optical gain and feedback," *Proceedings of ICO Topical Meeting on Optical Computing, Minsk Belarus*, Proc. SPIE, vol. 1806, pp.279-287.

Takeda, M and Kishigami, T (1993), "Dynamics of complex neural fields with an anlogy to optical fields generated in a phase-conjugate resonator: a review," *Proceedings of Conference on Chaos in Optics, San Diego USA*, Proc. SPIE, vol. 2039, pp. 314-322.

Yariv, A. and Pepper, D. M. (1977), "Amplified reflection, phase conjugation, and oscillation in degenerate four-wave mixing," *Opt. Lett.*, vol. 1, pp. 16-18.

Yariv, A. (1985), *Introduction to Optical Electronics*, CBS College Pub., New York, p.511.

Yariv A. and Kwon S. (1986), "Associative memories based on message-bearing optical modes in phase-conjugate resonators," *Opt. Lett.*, vol. 11, pp. 186-188.

Authors' address
Mitsuo Takeda: Dept. Information and Communication Engineering, The University of Electro-Communications, 1-5-1 Chofugaoka Chofu, Tokyo 182-8585, Japan.
Takaaki Kishigami: c/o M. Takeda Dept. Information and Communication Engineering, The University of Electro-Communications, 1-5-1 Chofugaoka Chofu, Tokyo 182-8585, Japan.

Chapter 15

Coherent Lightwave Neural Network Systems : Use of Frequency Domain

Sotaro Kawata and Akira Hirose

The spatial coherence of lightwave extends the ability of optical information-processing systems based on the spatial parallelism. Orthogonally to space, the temporal coherence also raises their potential by make good use of the vast optical frequency domain, i.e., the frequency domain multiplexing. The frequency domain attracts neural network systems because many of the neural specific dynamics originate from the distributed and parallel architecture. Furthermore, the optical carrier frequency can become the key information for modulation of the network behavior such as learning, self-organization and adaptive processing. In this chapter, we treat the complex-valued neural networks from the viewpoint of frequency-sensitive coherent lightwave information processing. We describe the theory and experimental results where the carrier frequency is found useful to control the learning and processing behavior of the neural networks.

1 Introduction

The neural networks have two characteristic features. First, the processing dynamics is embedded in simple synaptic weights. Second, simply operating neuron elements work together to yield an information processing dynamics. These facts are compatible with parallel processing architecture. Therefore it is important to investigate parallel hardware (Kolinummi *et al.* 1997).

However, in the conventional electronic hardware, we have a serious limitation on the operation speed. Because we need a cooling configuration, we cannot make the circuit smaller than a certain size. We also have the circuit capacitance and inductance. Then we are not free from a large signal-propagation delay. On the other hand, lightwave is expected to overcome the restriction.

The lightwave has a potential to realize a high-speed, flexibe and massively parallel three-dimensional interconnections. The only limitation is caused by the diffraction. A spatial lightwave modulator (SLM) modulates the lightwave patterns two-dimensionally in amplitude, phase or polarization. The SLM solves the connection delay problems in the electronic neural weights.

Though many ideas on optical neural networks have been proposed up to now (Psaltis *et al.* 1989, Ishikawa *et al.* 1989, Nitta *et al.* 1992), most of the systems use only the lightwave intensity or power. That is, they aim to utilize the spatial coherence, the straightness of propagation. The temporal coherence, i.e., the phase and frequency, are not in use. Besides, the lightwave has polarization. By using such a variety of lightwave variables, we will be able to realize an ultimately highly functional neural network hardware (Hirose and Eckmiller 1996). We call such systems the *coherent neural networks* where we utilize the wave and interference phenomena. Figure 1 illustrates the pioneering of the new dimensions.

In this Chapter, we describe the architecture, designs and experimental results of coherent lightwave neural network systems. They process the information by optical modulation, addition, coherent detection and nonlinear conversion, if needed, of lightwave. First we show the fundamental operation of a coherent associative memory system that embeds complex-valued signal vectors (Hirose and Kiuchi 2000). Then we expand the system into a frequency-domain

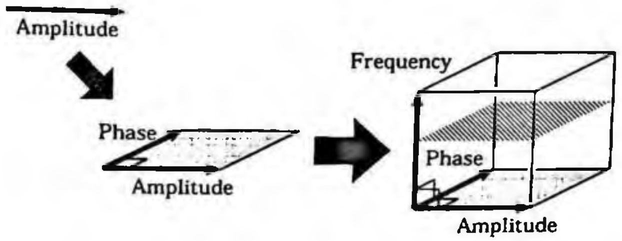


Figure 1. New dimensions of lightwave information space.

multiplexing network (Kawata and Hirose 2003) by which we show explicitly the use of the frequency domain as a new resource of the parallelism.

2 Coherent Neural Networks

2.1 The Amplitude-Phase Complex-Valued Neural Networks

In physiology we observe the membrane potential in the real-number domain. However, we can construct a complex-valued neural model as an extension of the real-valued one. The complex-valued expression is useful in many fields such as control and electrical circuits. In particular, when we deal with wave, the complex-valued treatment is essential to obtain a useful physical picture.

The research of the complex-valued neural networks can be traced back to the generalized threshold functions discussed in (Aizenberg *et al.* 1971). They extended the real binary threshold function to a multiple-valued threshold one that yields one of the discrete points on the unit circle in the complex plane. They suggested that such a machine is implemented by converting the phase information into pulse timing and by using a controllable time delay. In 1975, the complex-valued adaptive filter was proposed by Widrow's group (Widrow *et al.* 1975). They considered a linear system and described

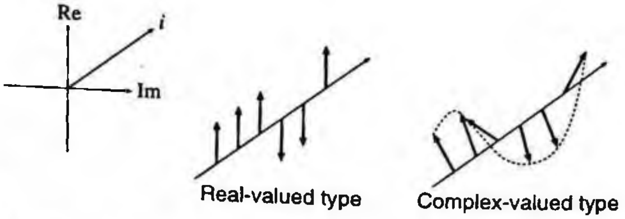


Figure 2. Difference of information space between the real-valued and the complex-valued networks where Re and Im stand for real and imaginary axes, respectively, and i is signal vector component index.

the complex-valued least mean square algorithm. In 1988, Noest proposed a phasor neural network and analyzed the capacity of the associative memory in which the phase value has information components with fixed amplitude (Noest 1988). On the other hand, several real-imaginary type complex-valued neural networks were proposed (Nitta and Furuya 1991, Nemoto and Kubono 1996, Birx and Pipenberg 1993) where the real and imaginary parts of the activation function are treated separately. Then the function is partially differentiable in each real or imaginary domain.

In 1992, the amplitude-phase type complex-valued neural network was proposed (Hirose 1992a, Georgiou and Koutsougeras 1992). This network has amplitude and phase as the basis vectors. Then the network dynamics is free from the artificial coordinate setup. That is, the expression does not depend on the real and imaginary axes. The fact is very important in particular when we deal with real physical phenomena which are intrinsically independent of coordinate. Though the amplitude-phase activation function is not analytical, the nonregularity is not a serious problem. The reason is that the neural learning and processing are performed realistically by a gradual and individual variation of signals and weights that are projected onto the two basis vectors on the two-dimensional complex plane.

This amplitude-phase framework is suitable for applications in many engineering fields such as the optical information processing sys-

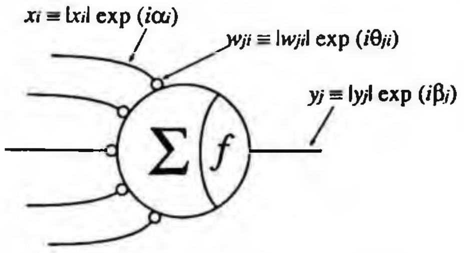

$x_i \equiv |x_i| \exp(i\alpha_i)$
$w_{ji} \equiv |w_{ji}| \exp(i\theta_{ji})$
$y_j \equiv |y_j| \exp(i\beta_j)$
Σ
f

Figure 3. Complex-valued Neuron.

tems, presented in this Chapter, and the amplitude modulation, phase modulation and frequency modulation in electromagnetic-wave communications and radars.

With the amplitude-phase type complex-valued neural-network theory, we can easily deal with a wave phenomenon where the phase value varies smoothly as shown in Fig. 2. When we design future neural networks with lightwave, electromagnetic wave, sound wave and quantum wave such as electron wave, the theory of this type will play a fundamental roll.

In particular, the use of high-frequency and short-wavelength lightwave will realize highly dense parallelism in both the spatial and frequency domains. In this section, we describe the complex-valued neural network theory for the coherent neural networks.

2.2 Neural Dynamics on Complex Number Plane

Figure 3 shows a complex-valued neuron. The neuron operation is expressed in terms of input signal from i-th neuron $x_i \equiv |x_i| \exp(i\alpha_i)$, synaptic weight $w_{ji} \equiv |w_{ji}| \exp(i\theta_{ji})$ and output signal $y_j \equiv |y_j| \exp(i\beta_j)$ as

$$y_j = f\left(\frac{1}{N}\sum_{i=1}^{N} w_{ji}x_i\right) \tag{1}$$

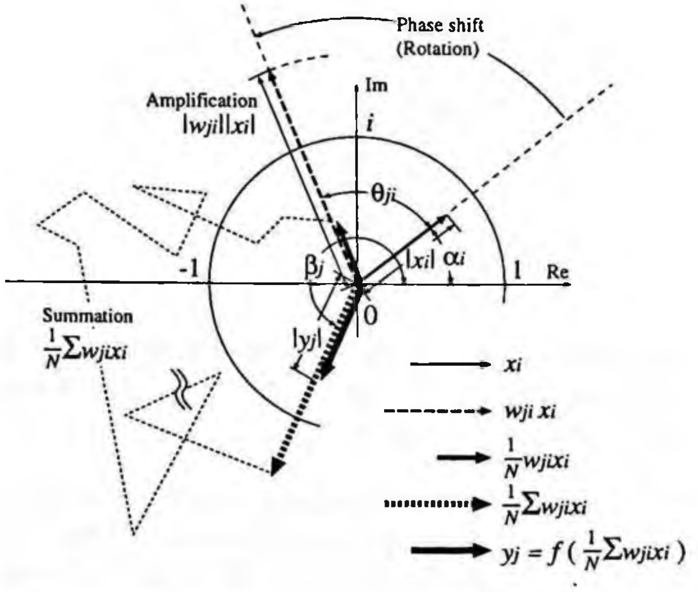


Figure 4. Weight multiplication, product summation and nonlinear conversion.

where $f(\cdot)$ expresses a certain nonlinear function and N is number of the inputs. Though several neuron nonlinear functions have been proposed, we choose the function so that the amplitude and the phase are converted separately as

$$f(z) \equiv A \tanh(g|z|) \exp(i \arg(z)) \tag{2}$$

where A ($\in$ Re) and g ($\in$ Re) denote output sigmoidal saturation amplitude and input amplitude gain. The function converts the amplitude nonlinearly while it leaves the phase component unchanged (Hirose 1992b). Figure 4, on the other hand, illustrates the weight multiplication and the neural summation of the network on the complex plane as well as the nonlinear conversion.

2.3 Coherent Neural Networks

A coherent associative memory dealing with phase information was proposed by Takeda and Kishigami (Takeda and Kishigami 1992).

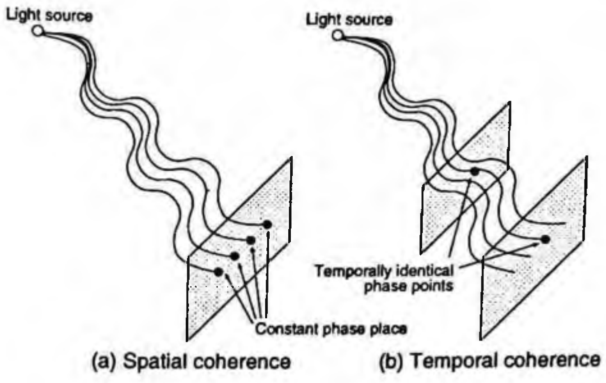


Figure 5. Two types of coherence.

The phase utilization leads to the multiplexing in the frequency domain. In particular, when a short-wavelength electromagnetic wave, i.e., lightwave, is introduced as the carrier, not only the spatial parallelism but also the vast frequency domain are available. The advantage originates from the coherence.

There are two types of coherence. One is the spatial coherence and the other is the temporal coherence. These concepts are very important when we design and construct a coherent system.

Figure 5(a) shows the spatial coherence. The coherent type neural network is a network that utilizes interference phenomena. Therefore, the constant phase surface should be well ordered on, for example, a plane or a sphere. A high directivity of the carrier lightwave realizes a flexible and massively parallel connections. The directivity, or the straightness of propagation, is based on Huygens' principle. Hence an effective control of the wavefront requires a well ordered phase surface in space. In other words, it needs a high spatial coherence.

On the other hand, Fig.5(b) shows the temporal coherence. The coherent system requires a reference lightwave to detect the phase of signals. Generally, the signal and reference lightbeams propagate

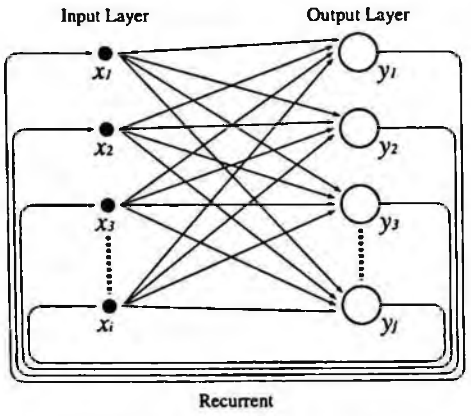


Figure 6. Associative memory network.

along optical paths different from each other. Two lightbeams yield a meaningful phase information by an interference only when they have a stably comparable phase values even if the optical path lengths are different. Therefore, the phase noise of the lightwave should be low enough. That is, the temporal coherence should be high.

We use a laser as a high spatial- and temporal- coherence light source. We choose a semiconductor laser diode in particular in the frequency-domain controllable system in which we can make good use of the frequency resource.

3 Coherent Lightwave Associative Memory System

3.1 Dynamics of Complex-Amplitude Associative Memories

Associative memories have a recurrent single-layer (fully connected) structure shown in Fig. 6. A connection weight w_{ji} from i-th neuron to j-th one is determined by complex-amplitude vectors to be embedded (memorized) $\boldsymbol{s}_\mu \equiv s_{i\mu}$ (index $\mu = 1, 2, ...$) as (Hirose 1994)

$$w_{ji} = \sum_{\mu} s_{j\mu}(s_{i\mu})^* \tag{3}$$

where $(\cdot)^*$ denotes Hermitian conjugate. This scheme is called the correlation learning. As to the neuron nonlinearity, the neuron activation function f employs saturant nonlinearity in the amplitude, whereas only a weak nonlinearity or simply a linear transfer function is used in the phase (Hirose 1992a). Input signals $\boldsymbol{x}(l) \equiv x_i(l)$ at l-th iteration are transformed by a single neural network process to yield output signals $\boldsymbol{x}(l+1)$ as

$$x_j(l+1) = f\left(\frac{1}{N}\sum_{i=1}^{N} w_{ji}x_i(l)\right) \tag{4}$$

where N denotes neuron number. In most associative memories, the saturation works only as an amplitude limiter. Therefore, precise properties of the activation function f are not significant in the neural dynamics.

Iterative neural processing by the recurrent signal flow performs a recall of a memorized vector. That is to say, a noisy initial input signal vector $\boldsymbol{x}(0)$ goes through the neural network iteratively to converge finally at one of the embedded vectors $\boldsymbol{s}_\mu$ which is most similar to the input $\boldsymbol{x}(0)$.

3.2 System Construction

Figure 7 is a schematic illustration of the optical part of a (forward-processing) coherent lightwave neural network (Hirose and Kiuchi 2000). A laser lightwave of angular frequency ω is used as the information carrier. The optical circuit forms totally parallel self-homodyne interferometers (Hirose and Eckmiller 1996).

Complex-amplitude input signals $\boldsymbol{x} = [x_i]$ is fed to spatial light modulators (SLM's). The SLM's modulate the amplitudes and the phases

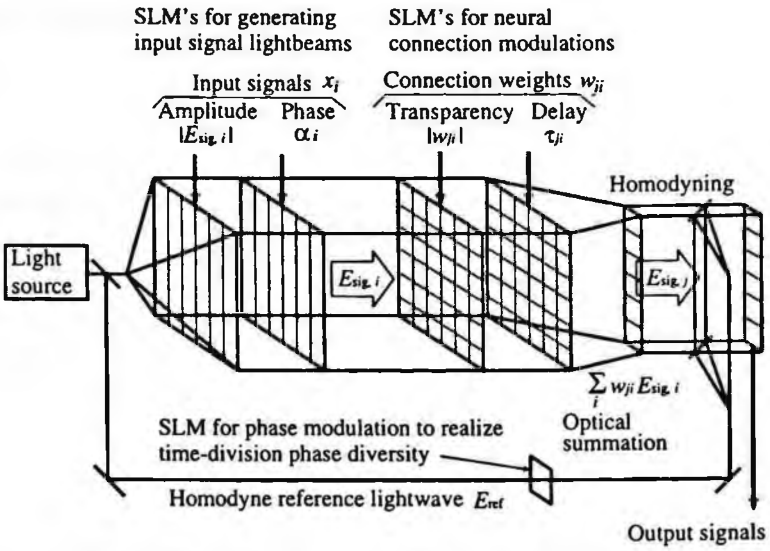

SLM's for generating input signal lightbeams
SLM's for neural connection modulations
Input signals x_i
Amplitude $|E_{sig,i}|$
Phase α_i
Connection weights w_{ji}
Transparency $|w_{ji}|$
Delay τ_{ji}
Homodyning
Light source
$E_{sig,i}$
$E_{sig,j}$
$\sum_i w_{ji} E_{sig,i}$
Optical summation
SLM for phase modulation to realize time-division phase diversity
Homodyne reference lightwave E_{ref}
Output signals

Figure 7. Schematic construction.

of the lightbeams to generate parallel signal lightwaves. The electric field of ith signal light $E_{\text{sig},i}$ can be expressed in amplitude and phase terms with time t as

$$\begin{aligned} E_{\text{sig},i} &\equiv |E_{\text{sig},i}| \exp(i(\omega t + \alpha_i)) \\ &= \exp(i\omega t) \cdot x_i \end{aligned} \tag{5}$$

where the phase angle α_i expresses delay or advance to the reference phase mentioned below. These input signal lightbeams propagate in parallel and are incident upon the neural connection SLM's. Here they are modulated according to the transparency $|w_{ji}|$ and the time delay $\tau_{ji} = \theta_{ji}/\omega$ of neural connections $\mathbf{W} = [w_{ji}]$ which have been given by (3) (Hirose and Eckmiller 1996). The modulated signals are written as

$$w_{ji} \cdot E_{\text{sig},i} = |w_{ji}|\, |E_{\text{sig},i}| \exp(i\,(\omega t + \omega\tau_{ji} + \alpha_i)) \tag{6}$$

Then they are summed optically to yield output signals expressed as

$$E_{\text{sig},j} \equiv |E_{\text{sig},j}| \exp(i\,(\omega t + \beta_j))$$

$$= \sum_i (w_{ji} \cdot E_{\text{sig},i}) \tag{7}$$

where the amplitude $|E_{\text{sig},j}|$ and the phase β_j are related to the input lightwaves $E_{\text{sig},i}$. The neuron activation function (4) can be implemented, for example, by using the saturation characteristics in optical amplifiers or optoelectronic transducers. The equation corresponding to (4) is written by using a saturation electric-field value E_0 as

$$\begin{aligned} f(E_{\text{sig},j}) &\equiv E_0 \tanh |E_{\text{sig},j}| \exp(i\,(\omega t + \beta_j)) \\ &= \exp(i\omega t) \cdot y_j \end{aligned} \tag{8}$$

This expression is mostly consistent with physical realizations. In most neural dynamics, however, precise profile of the nonlinearity has less importance. Therefore, in the experiment below, no explicit nonlinear processing is introduced for simplicity.

The summed signal lightbeams $E_{\text{sig},j}$ are mixed with a reference lightwave $E_{\text{ref}} \equiv |E_{\text{ref}}| \exp(i\omega t)$. The homodyne process generates electrical output currents $\boldsymbol{I} = I_j$ which are expressed with a response coefficient of the detectors R (detector sensitivity) as

$$\begin{aligned} I_j &= R\,|E_{\text{sig},j} + E_{\text{ref}}|^2 \\ &= R\,\left(|E_{\text{sig},j}|^2 + |E_{\text{ref}}|^2 + 2|E_{\text{sig},j}|\,|E_{\text{ref}}| \cos(i\beta_j)\right) \end{aligned} \tag{9}$$

Therefore, in the case that $|E_{\text{sig},j}| \ll |E_{\text{ref}}|$, or $|E_{\text{sig},j}|$ is almost constant, the output complex amplitude (except for the *dc* component) is proportional to $|E_{\text{sig},j}| \cos(i\beta_j)$.

To obtain both the in- and quadrature- phase components, we employ a time-division phase-diversity method in this experiment. The reference lightwave phase ϕ_{ref} (i.e., $E_{\text{ref}} = |E_{\text{ref}}| \exp(i\,(\omega t + \phi_{\text{ref}}))$) is also modulated as 0, $\pi/2$, π, and $3\pi/2$ cyclically in time. Then, the difference between the detected signals at 0 and π yields the in-phase value, whereas $\pi/2$ and $3\pi/2$ the quadrature value. The zero phase is chosen by the detected values without the signal modulation. Accordingly both the amplitude and phase information is extracted.

Thus the input complex-amplitude signal vector $\boldsymbol{x}$ yields an output vector $\boldsymbol{I}$ as a result of the neural processing. In the above explanation, a single-layer forward-processing system is taken as an example. However, more complicated neural circuits can also be constructed (Hirose 1994). In the following experiment, the output signals $\boldsymbol{I}$ obtained for $\boldsymbol{x}(l)$ at iteration number l generate electrically iterative input signals $\boldsymbol{x}(l+1)$.

3.3 Experiment

Figure 8(a) shows the experimental setup. A He-Ne laser lightwave (0.63μm) is used as the information carrier. After collimation, the lightbeam provides both the signal and reference lightwaves in parallel. The upper half of the beam is the signal light, and the lower one is the reference. They are modulated by an SLM, separated by a corner mirror, mixed to generate interference, and detected at a CCD camera. The SLM is a parallel-aligned nematic liquid crystal type (PAL-SLM, Hamamatsu X6345: 20mm$\times$20mm area, video input signal). It is originally for phase modulation. However, in this experiment, half of its SLM surface is combined with a polarizer to enable itself to modulate the amplitude.

The SLM surface is divided into eight divisions as shown in Fig.8(b). The upper four divisions are used for the following modulations: input amplitude generation $|E_{\mathrm{sig},i}|$, input phase generation α_i, neural-connection transparency modulation $|w_{ji}|$, and neural-connection time-delay modulation τ_{ji}. One of the lower divisions is used for reference phase modulation to realize the time-division phase diversity, whereas other three divisions are not in use (i.e., simple mirrors).

The input-signal generating divisions $|E_{\mathrm{sig},i}|$ and α_i have strips (subdivisions), each of which corresponds to an input signal (amplitude

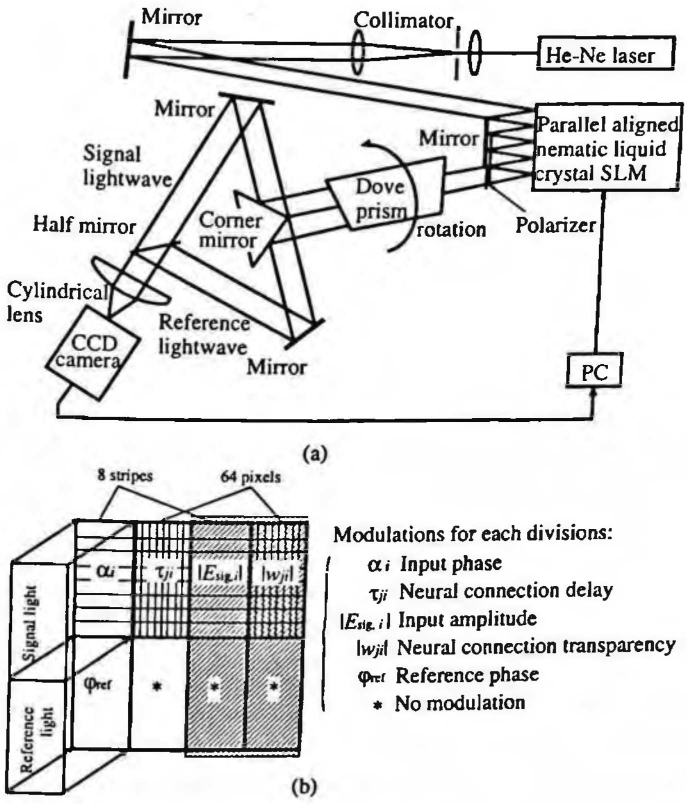


Figure 8. (a) Construction of the coherent optical associative memory and (b) SLM surface divisions.

or phase value). The neural connection divisions $|w_{ji}|$ and τ_{ji} have pixels (subdivisions), each of which corresponds to a neural connection weights (transparency or delay value). This mapping realizes the neural and homodyne processes in cooperation with the following optical circuit.

As shown in Fig.8(b), the signal and reference lightbeams are incident in parallel only on the left two divisions slightly at the skew. Then the beams make four round trips between the SLM and a facing plain mirror with a constant displacement horizontally. These

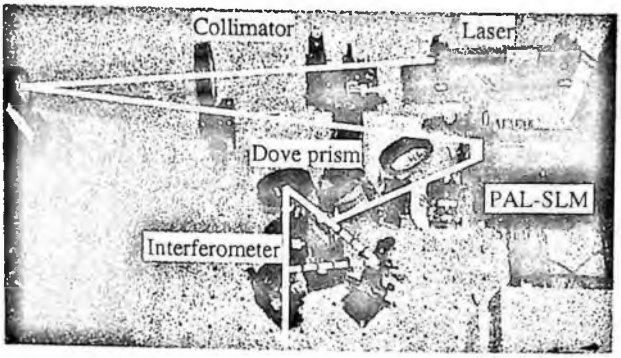


Figure 9. Experimental setup.

reflections realize the signal light generation and the neural weight multiplication at the upper divisions, as well as the reference phase modulation at the lower division for the time-division phase diversity.

The neural-processed signal and reference lightbeams are separated at the corner mirror, mixed, and summed up by a cylindrical lens to be detected at the CCD. The order of the mixing and the summation is also reversed from the explanation ((6) and (9)) only for experimental simplicity. The detected interference is fed to a personal computer where the CCD pixel data are averaged in each neuron region.

Figure 9 is a photograph of the experimental setup. Associative memories require a recurrent structure which feeds the output signals to the neural inputs. In the present system, the output signals obtained in the personal computer generate a recurrent input signal vector and controls the the SLM input-signal divisions. The phase modulation signal required for the time-division phase diversity is also generated by the personal computer.

The maximum neural connection number is about 64 in this setup. It is limited by the available SLM surface area and the lightwave

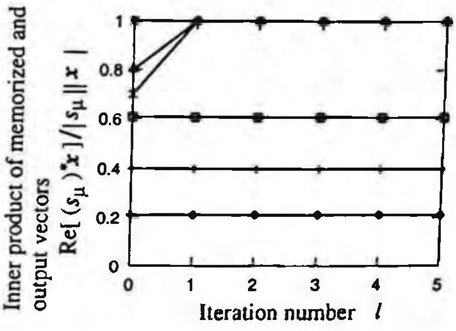


Figure 10. Time evolution of normalized inner products of memorized s_μ and output x vectors for various initial input signals versus iteration number l.

diffraction. The corresponding neuron number is $\sqrt{64} = 8$. The scale is slightly too small to employ a numerical analysis using the statistical neurodynamics. However, we take a general and empirical result of the statistical theory as a yardstick. The memory capacity of complex-valued associative memories is theoretically calculated as $\pi^2/8$ times as large as that of real-valued memories (Noest 1988). Non-sparse real memories have generally a capacity of 0.15 times the number of neurons. Then the capacity of the present system is estimated around 1.5. Therefore the number of the memorized vectors is chosen two, i.e., s_1 and its inverse $s_2 = -s_1$ which is automatically memorized in this case.

Figure 10 shows the recalling-process evolution versus iteration number l when various initial input vectors are fed to the associative memory. The normalized inner product of the signal vector $x(l)$ and the embedded vector s_1 is shown as the similarity. It should be unity when the recalling process is successfully completed. The attracting basin seems to be a region where the inner product is larger than ~0.7 in this case. Because of the scale smallness, a quantitative comparison with the mean field approximation theory is not available. However, a similar result is obtained by a programmed numerical simulation.

4 Extension to a Carrier-Frequency Controllable System

4.1 Basic Idea

Figure 11 presents the basic idea of the coherent associative memory that has a carrier-frequency dependent behavior (Kawata and Hirose 2003). The coherent network behavior can potentially be carrier-frequency sensitive. When the system has a carrier frequency $f_1 = \omega_1/2\pi$, it constructs a metric in the information space. If a signal vector $\boldsymbol{x}_1(\omega_1)$ at ω_1 is fed to the system, the system recalls the nearest attractor $\boldsymbol{s}_1(\omega_1)$. For $\boldsymbol{x}_2(\omega_1)$, it recalls $\boldsymbol{s}_2(\omega_1)$ in the same way. On the other hand, when the system carrier frequency is $f_2 = \omega_2/2\pi$, the metric is changed and the dynamics is varied. That is, for example, $\boldsymbol{x}_2(\omega_2)$ approaches $\boldsymbol{s}_1(\omega_2)$ because $\boldsymbol{x}_2(\omega_2)$ is near to $\boldsymbol{s}_1(\omega_2)$ in this metric, while $\boldsymbol{x}_1(\omega_2)$ may be far from all the attractors and trapped into a local minimum.

The above recalling story is only an example of the behavior of the frequency-dependent associative memory. The dynamics is determined by a learning process which is also frequency dependent. The details are presented in the follows.

4.2 Complex Hebbian Learning Rule

The weighting matrix value $w_{ji} = |w_{ji}| \exp(i\omega\tau_{ji})$ is determined by a Hebbian learning rule to construct an associative memory network. The conventional Hebbian rule changes the weight according to the product of signals presented (or obtained) at the input and output layers, $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. We extend the rule into a frequency-sensitive complex-valued version. The complex-valued Hebbian rule is expressed in terms of the amplitude $|w_{ji}|$ and the delay time τ_{ji} of

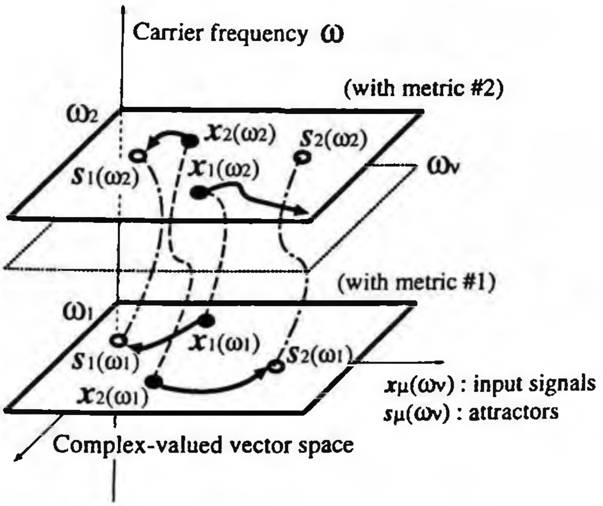


Figure 11. Conceptual illustration of the carrier-frequency dependent behavior of the associative memory.

the weight w_{ji} for an input signal $x_i = |x_i| \exp(i\alpha_i)$ and an output supervisor $y_j = |y_j| \exp(i\beta_j)$ as (Hirose *et al.* 2001)

$$\frac{d|w_{ji}|}{dt} = -|w_{ji}| + K|y_j||x_i| \cos(\beta_j - \alpha_i - \arg(w_{ji})) \quad (10)$$

$$\frac{d\tau_{ji}}{dt} = \frac{K}{\omega} \frac{|y_j||x_i|}{|w_{ji}|} \sin(\beta_j - \alpha_i - \omega\tau_{ji}) \quad (11)$$

where K denotes learning gain. The process (10) and (11) is repeated sequentially for various input-output signal sets.

4.3 Optical Implementation

Figure 12 shows the construction of the system. Frequency $f = \omega/2\pi$ of the light source (laser diode: LD #1) can be controlled by choosing the injection current appropriately. The light beam is divided into halves corresponding to signal and reference waves. The former is modulated by an SLM#1 yielding a signal vector $\boldsymbol{x}(l) = [x_i(l)]$.

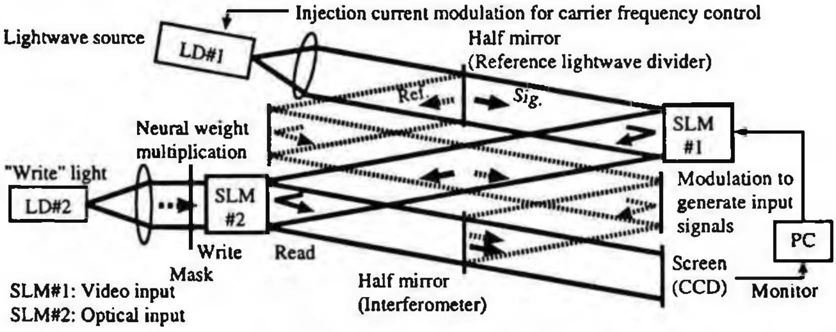

Injection current modulation for carrier frequency control
Lightwave source
LD#1
Half mirror
(Reference lightwave divider)
Ref.
Sig.
SLM #1
Neural weight multiplication
"Write" light
LD#2
SLM #2
Modulation to generate input signals
PC
Write
Read
Mask
Screen (CCD)
Monitor
Half mirror (Interferometer)
SLM#1: Video input
SLM#2: Optical input

Figure 12. Construction of the carrier-frequency-controllable coherent optical associative memory.

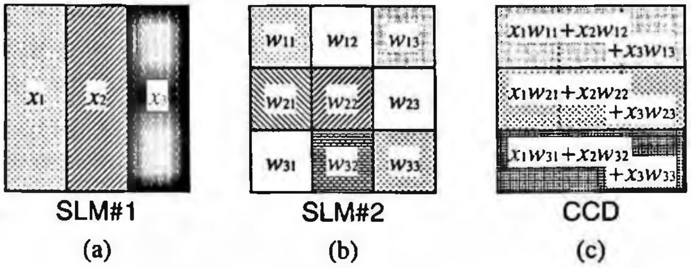


Figure 13. Signal assignment on SLMs and CCD surfaces in the case of 3 neurons and 9 synapses as an example: (a) SLM#1, (b) SLM#2 and (c) CCD.

The following SLM#2 multiplies the signal, which is incident on the *read* surface, by the neural weights $\mathbf{W} = [w_{ji}]$ optically. That is, the *write* surface of SLM#2 is illuminated by LD#2 with an optical weighting mask. At the both SLMs, only the phase (or, actually, time delay τ_{ji}) is modulated. The amplitude $|x_i|$ and $|w_{ji}|$ is always unity in this experiment for simplicity.

Figures 13 (a) and (b) show the use of the SLM modulation surfaces (when the numbers of the neurons and the synapses are 3 and $3^2 = 9$, respectively, as an example). Both of them modulate the lightwave phase pixelwise by changing their permittivity which is equivalent to delay time.

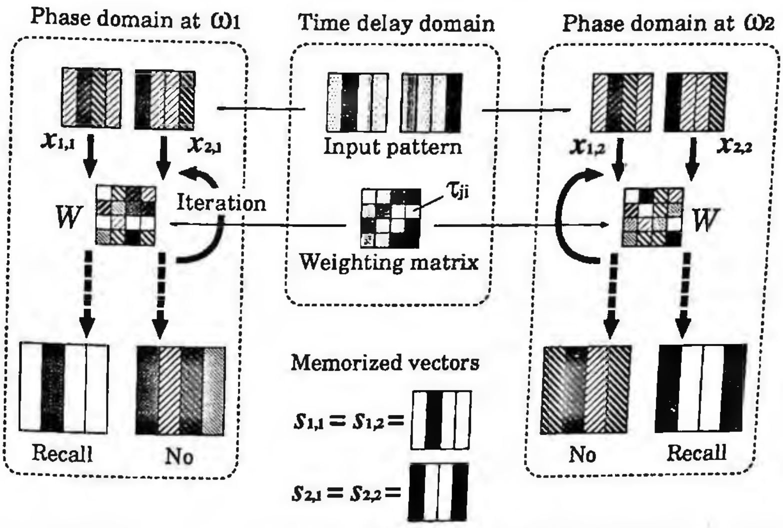


Figure 14. The frequency-dependent recall and the relation between delay time and phase.

Figure 14 illustrates the carrier-frequency-dependent recall process and the relation between delay time and phase. The weights expressed by the delay time τ_{ji} are determined at certain values by learning, the phase values $\omega\tau_{ji}$ are changed by the frequency modulation. Therefore, even for an identical input and the delay, the system recalls a different output pattern depending on the frequency. That is to say, the dynamics is modulated by the change of the carrier frequency value.

4.4 Frequency Sensitive Learning

In Fig. 15, the difference between the optical path lengths of the signal and the reference $\Delta L \equiv L_{\text{sig}} - L_{\text{ref}}$ determines the dependence of the homodyne interference on the lightwave carrier frequency ω. The fringe at the screen changes cyclically against the light carrier frequency ω with a frequency period of $c/\Delta L$ where c denotes the light speed. The initial value of the time delay τ_{ji0}^{Hebb} of a weighting

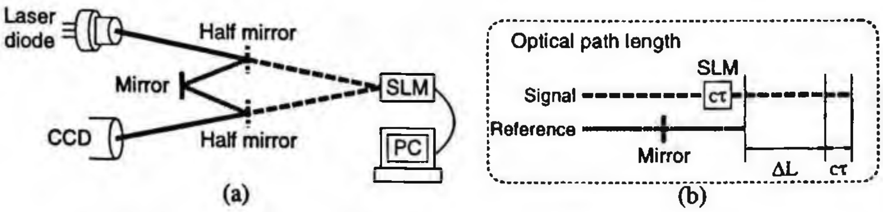


Figure 15. (a) Optical path difference between the signal path and the reference path and (b) an equivalent diagram.

matrix before the frequency-sensitive Hebbian learning is expressed with a basis arbitrary frequency (arbitrary value) ω_0 as

$$\tau_{ji0}^{\text{Hebb}} = \frac{\theta_0}{\omega_0} + \frac{\Delta L}{c} \tag{12}$$

where θ_0 denotes a initial phase value in the range of $[0, 2\pi)$ chosen at random. According to the theory (Hirose and Eckmiller 1996), a large time delay has a large influence on the phase change even for a small frequency variation. In our setup, $\Delta L/c$ works as the large time delay to realize an appropriate frequency sensitivity.

The weighting matrix is adjusted in the PC according to a set of desired attractors $\boldsymbol{s}_{\mu,\nu}$ where μ and ν are indices showing attractor number and intended (recalling) frequency number, respectively. As we provide both the input and output terminals of the memory with attractors $\boldsymbol{s}_{\mu,\nu}$ at a frequency ω_ν, the vector $\boldsymbol{s}_{\mu,\nu}$ is embedded in the weighting matrix associated with the frequency value.

4.5 SLM Modulation and Detected Phase Value

Following are the equations expressing the relation between delay time and phase of the weighting matrix when we take into consideration the difference of the optical path lengths. The synaptic phase modulation value $\theta_{ji}^{\text{SLM\#2}}(\omega_0)$ at SLM#2 is expressed as

$$\theta_{ji}^{\text{SLM\#2}}(\omega_0) \equiv \omega_0 \tau_{ji}^{\text{SLM\#2}} = \omega_0 (\tau_{ji}^{\text{Hebb}} - \frac{\Delta L}{c}) \tag{13}$$

where $\tau_{ji}^{\mathrm{SLM\#2}}$ and $\tau_{ji}^{\mathrm{Hebb}}$ denote the SLM delay time corresponding to $\theta_{ji}^{\mathrm{SLM\#2}}(\omega_0)$ and the total synaptic delay time of the weight generated by the learning process (11), respectively. The equivalent phase of the weights $\phi_{ji}(f)$ is expressed as

$$\begin{aligned}\phi_{ji}(\omega) &= \omega(\tau_{ji}^{\mathrm{SLM\#2}} + \frac{\Delta L}{c}) \\ &= \frac{\omega}{\omega_0}\theta_{ji}^{\mathrm{SLM\#2}}(\omega_0) + \omega\frac{\Delta L}{c} \end{aligned} \tag{14}$$

where ω is the carrier frequency used for the phase detection. When the frequency deviates from the intended recalling frequency ω_ν, i.e., $\omega = \omega_\nu + \Delta\omega$, the equivalent synaptic phase $\phi_{ji}(\omega_\nu + \Delta\omega)$ is calculated as

$$\begin{aligned}\phi_{ji}(\omega_\nu + \Delta\omega) &= \frac{\omega_\nu + \Delta\omega}{\omega_0}\theta_{ji}^{\mathrm{SLM\#2}}(\omega_0) + (\omega_\nu + \Delta\omega)\frac{\Delta L}{c} \\ &= \phi_{ji}(\omega_\nu) + \frac{\Delta\omega}{\omega_0}\theta_{ji}^{\mathrm{SLM\#2}}(\omega_0) + \Delta\omega\frac{\Delta L}{c} \end{aligned} \tag{15}$$

When $\Delta\omega \ll \omega_0$ in (15), the second term on the right-hand side is almost zero because $\Delta\omega/\omega_0 \cong 0$. However, the last term $\Delta\omega\frac{\Delta L}{c}$ influences the phase shift in proportion to $\Delta\omega$ and ΔL, resulting in modulation of the recalling behavior. Consequently, the recalling process is successful only when the frequency deviation $\Delta\omega$ is related to the optical path-length difference ΔL as

$$\Delta\omega\frac{\Delta L}{c} = 2n\pi \tag{16}$$

where n stands for integer.

4.6 Experiment

Figure 16 shows the experimental setup. In Fig. 12, the frequency of the light source LD#1 (SHARP LT051PS, wavelength 635[nm]

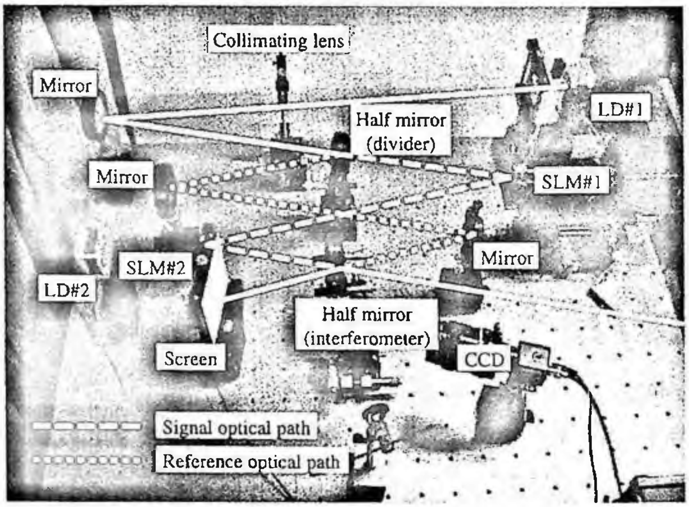


Figure 16. Experimental setup of the carrier-frequency dependent system.

$f_0 \cong 472$[THz]) is changed by injection current control. The frequency sensitivity is 10.5[GHz/mA]. The temperature is stabilized electrically. The frequency controllability (resolution) is better than 0.016[GHz]. SLM#1 (Hamamatsu Photonics X6345) is addressed with a video input, while SLM#2 (X7665) is with a spatially parallel optical input.

The optical path-length difference is about ΔL=6.8[mm], resulting in the frequency period of $c/\Delta L = 44.1$[GHz]. Numbers of neurons and synaptic connections are 9 and 81, respectively. We select four vectors $\boldsymbol{s}_{1,1}$, $\boldsymbol{s}_{1,2}$, $\boldsymbol{s}_{2,1}$ and $\boldsymbol{s}_{2,2}$ (where $\boldsymbol{s}_{1,1} = \boldsymbol{s}_{1,2}$, $\boldsymbol{s}_{2,1} = \boldsymbol{s}_{2,2}$ and $\boldsymbol{s}_{\mu,\nu} \equiv \boldsymbol{s}_{\mu}(\omega_\nu)$) to be memorized for two carrier frequencies ω_ν(ν=1,2). We intend that $\boldsymbol{s}_{1,1}$ be recalled at ω_1 and $\boldsymbol{s}_{2,2}$ at ω_2, respectively. The amplitudes of all the elements of both of them are unity. The frequencies are chosen as $\omega_1 = \omega_0$ and $\omega_2 = \omega_0 + (\pi c)/(2\,\Delta L)$ so that the network behavior becomes independent.

We generate input vectors $\boldsymbol{x}_{1,1}(= \boldsymbol{x}_{1,2})$ and $\boldsymbol{x}_{2,1}(= \boldsymbol{x}_{2,2})$ by adding phase noise whose distribution is homogeneous within 30 % range

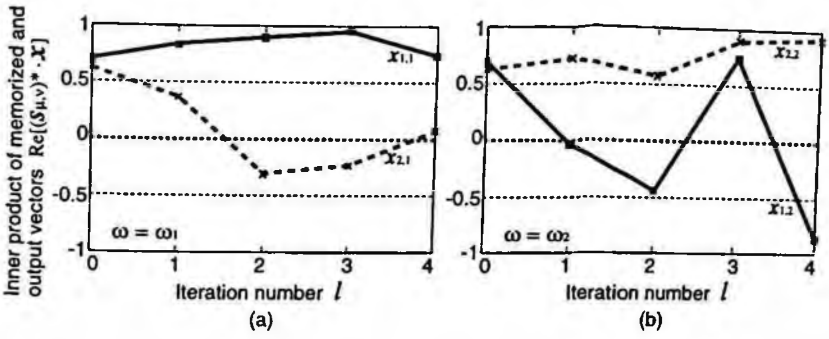


Figure 17. Experimental results of recalling process showing evolutions of the inner products of embedded- and signal-vectors $\mathrm{Re}[(s_{\mu,\nu})^* \cdot x]$ versus iteration number for different carrier frequencies.

of $\pm\pi$. The elements' amplitudes of both the input vectors are unity again. The learning process generates the weights w_{ji} on a PC. First, we initialize the delays τ_{ji} by choosing the initial phase θ_0 in (12) at random. Then the learning process based on (11) is iterated 1000 times, with which the weights settle at a sufficiently steady state. The learning gain K is 0.5.

Figure 17 shows the recalling result for noisy input vectors $x_{1,1}$ (near to $s_{1,1}$) and $x_{2,2}$ (near to $s_{2,2}$) when one of the carrier frequencies ω_1 or ω_2 is chosen. In the case of ω_1 shown in Fig.17(a), the inner product $\mathrm{Re}[(s_{1,1})^* \cdot x_{1,1}]$ converges almost at unity, which means that the system recalls the corresponding vector $s_{1,1}$. On the other hand, the inner product $\mathrm{Re}[(s_{2,1})^* \cdot x_{2,1}]$ shows oscillatory behavior, resulting in a failure in the recalling process of $s_{2,1}$. In Fig.17(b), contrarily, the system recalls $s_{2,2}$, while it does not $s_{1,2}$. We actually carried out experiments many times for various input vectors. When the inputs are somewhat near to one of the attractors, we typically observe similar results. On the other hand, when they are far from the attractors, then the overlaps converge at around zero. Accordingly, it is found that the experiments demonstrate a frequency-dependent behavior.

5 Conclusion

We have demonstrated the coherent optical associative memory whose behavior can be changed by the control of carrier frequency. The system has a homodyne construction where the signal and the reference optical path lengths are slightly different from each other. We have introduced a carrier-frequency dependent complex-valued Hebbian rule to realize the learning process. The result leads to a future frequency-domain parallelism in the optical neural networks.

References

Aizenberg, N.N., Ivaskiv, Y.L. and Pospelov, D.A. (1971), "A certain generalization of threshold functions," *Doklady Akademii Nauk SSSR*, vol. 196, pp. 1287-1290. English translation: *Soviet Physics – Doklady*, vol. 16 (1971), pp. 84-86.

Birx, D.L. and Pipenberg, S.J. (1993), "A complex mapping network for phase sensitive classification," *IEEE Trans. Neural Networks*, vol. 4, pp. 127-135.

Georgiou, G.M. and Koutsougeras, C. (1992), "Complex-domain backpropagation," *IEEE Trans. Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 39, pp. 330-334.

Hirose, A. (1992a), "Dynamics of fully complex-valued neural networks," *Electron. Lett.*, vol. 28, pp. 1492-1494.

Hirose, A. (1992b), "Continuous complex-valued back-propagation learning," *Electron. Lett.*, vol. 28, pp. 1854-1855.

Hirose, A. (1994), "Applications of complex-valued neural networks to coherent optical computing using phase-sensitive detection scheme," *Inform. Sci. - Applications*, vol. 2, pp. 103-117.

Hirose, A. and Eckmiller, R. (1996), "Coherent optical neural networks that have optical-frequency-controlled behavior and generalization ability in the frequency domain," *Appl. Opt.*, vol. 35, pp. 836-843.

Hirose, A. and Kiuchi, M. (2000), "Coherent optical associative memory system that processes complex-amplitude information," *IEEE Photon. Tech. Lett.*, vol. 12, pp. 564-566(2000).

Hirose, A., Tabata, C. and Ishimaru, S. (2001), "Coherent neural network architecture realizing a self-organizing activeness mechanism," *International Conference on Knowledge-Based Intelligent Information Engineering System & Allied Technologies (KES 2001 Osaka)*, pp. 576-580.

Ishikawa, M., Mukohzaka, N., Toyoda, H. and Suzuki, Y. (1989), "Optical Associatron - A Simple Model for Optical Associative Memory -," *Appl. Opt.*, vol. 28, pp. 291-301.

Kawata, S. and Hirose, A. (in press), "Coherent Lightwave Associative Memory System that Possesses a Carrier-Frequency-Controlled Behavior," *Opt. Eng.*, (in press).

Kolinummi, P., Hamalainen, T. and Kaski, K. (1997), "Improving ANN processing with a dedicated hardware system," *IEEE circuits and Devices*, vol. 13, pp. 19-27.

Nemoto, I. and Kubono, M. (1996), "Complex associative memory," *Neural Networks*, vol. 9, pp. 253-261.

Nitta, T. and Furuya, T. (1991), "A complex back-propagation learning," *J. Information Processing Society of Japan*, vol. 32, pp. 1319-1329 (in Japanese).

Nitta, Y., Ohta, J., Takahashi, M., Tai, S. and Kyuma, K. (1992), "Optical neurochip with learn capability," *IEEE Photon. Technol. Lett.*, vol. 4, pp. 35.

Noest, A.J. (1988), "Associative memory in sparse phasor neural networks," *Europhys. Lett.*, vol. 6, pp. 469-474.

Psaltis, D., Lin, S., Yamamura, A., Gu, X., Hsu, K., and Brady, D. (1989), "Optoelectronic implementations of neural networks," *IEEE Commun. Mag.*, vol. 27, pp. 37-40.

Takeda, M. and Kishigami, T. (1992), "Complex neural fields with a Hopfield-like energy function and an analogy to optical fields generated in phase-conjugate resonators," *J. Opt. Soc. Am. A*, vol. 9, pp. 2182-2191.

Widrow, B., McCool, J. and Ball, M. (1975), "The complex LMS algorithm," *Proc. IEEE.*, vol. 63, pp. 719-720.

Authors' address
Sotaro Kawata and Akira Hirose: Department of Frontier Informatics, Graduate School of Frontier Sciences, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
kawata@eis.t.u-tokyo.ac.jp
ahirose@ee.t.u-tokyo.ac.jp

Index

Contributors

Hiroyuki Aoki
Dept. Electronic Engineering, Tokyo National College ofTechnology, 1220-2 Kunugida-machi, Hachioji-shi, Tokyo 193-0997, Japan

George M. Georgiou
Computer Science Department, California State University, San Bernardino, CA 92407-2397, U.S.A.

Su Lee Goh
Communication & Signal Processing Group, Department of Electrical and Electronic, Imperial College of Science, Technology and Medicine, London SW7 2AZ, United Kingdom

Masafumi Hagiwara
Department of Information and Computer Science, Faculty of Science and Technology, Keio University, 3-14-1 Hiyoshi, Yokohama, Kanagawa 223-8522, Japan.

Andrew I. Hanna
Signal & Imaging Informatics Group, Royal Society Wolfson Bioinformatics Lab, The University of East Anglia, Norwich, NR4 7TJ, United Kingdom

Akira Hirose
Department of Electrical and Electronic Engineering, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Sotaro Kawata
Department of Frontier Informatics, Graduate School of Frontier Sciences, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

Subhash Kak
Department of Electrical & Computer Engineering, Louisiana State University, Baton Rouge, LA 70803, U.S.A.

Makoto Kinouchi
Department of Bio-System Engineering, Faculty of Engineering, Yamagata University, 4-3-16 Jounan, Yonezawa, Yamagata 992-8510, Japan.

Takaaki Kishigami
Dept. Information and Communication Engineering, The University of Electro-Communications, 1-5-1 Chofugaoka Chofu, Tokyo 182-8585, Japan

Yasuaki Kuroe
Department of Electronics and Information Science, Kyoto Institute of Technology, Matsugasaki, Skyo-ku, Kyoto 606-8585, Japan

Donq-Liang Lee
Dept. Computer Science and Information Engineering, Van Nung Institute of Technology, Chung Li 32056, Taiwan, R.O.C.

Danilo P. Mandic
Communication & Signal Processing Group, Department of Electrical and Electronic, Imperial College of Science, Technology and Medicine, London SW7 2AZ, United Kingdom

Teruyuki Miyajima
Department of Systems Engineering, Faculty of Engineering, Ibaraki University, 4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan.

Iku Nemoto
Dept. Information Environment Design and Integration, Tokyo Denki University, Muzai-gakuendai, 2-1200, Inzai, Chiba, 270-1382, Japan.

Tohru Nitta
Mathematical Neuroinformatics Group, Neuroscience Research Institute, AIST, Tsukuba Central 2, 1-1-1 Umezono, Tsukuba-shi, Ibaraki, 305-8568 Japan

Justin Pearson
Uppsala University, Department of Information Technology, Box 337, SE-751 05 Uppsala, Sweden.

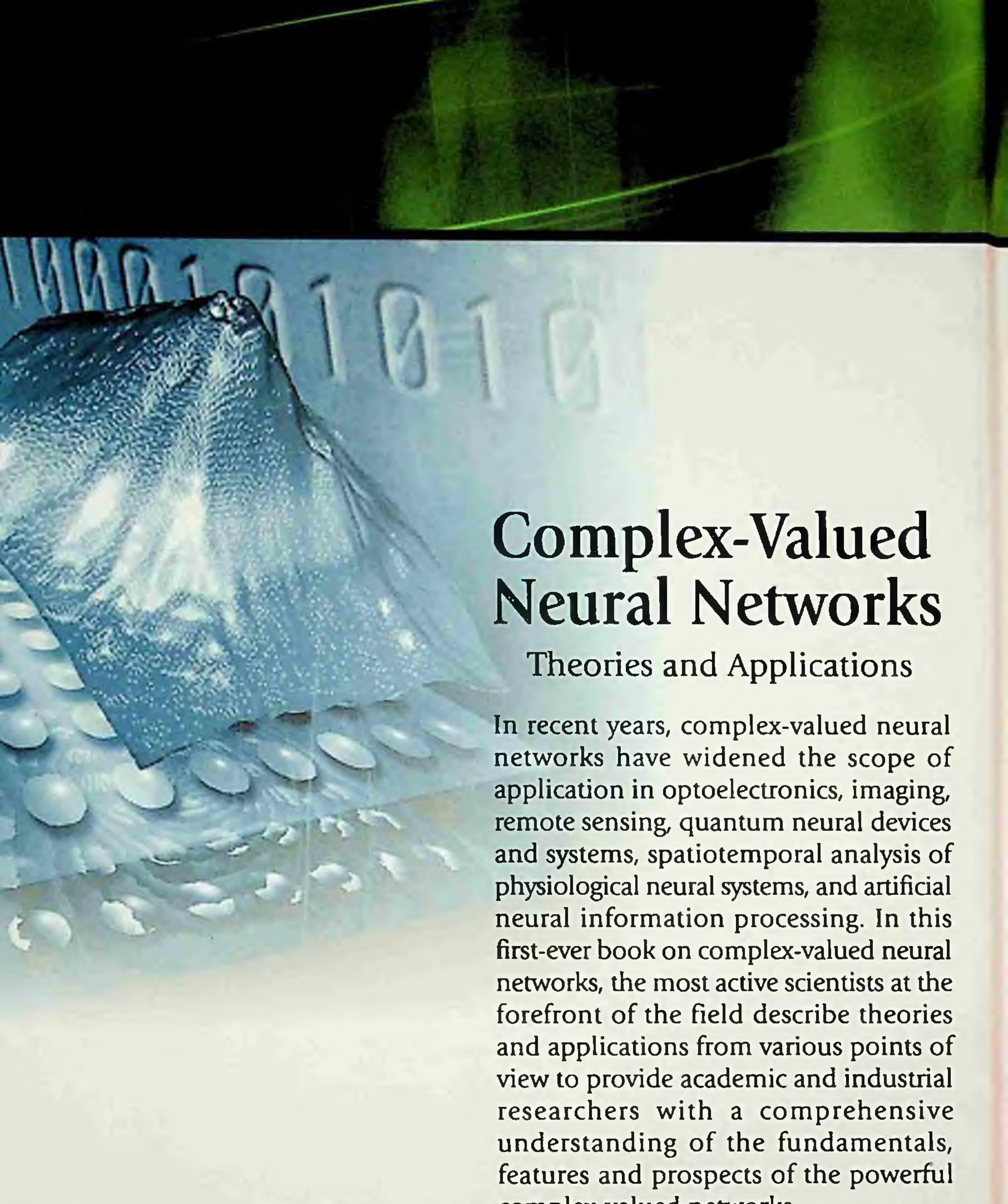
Pritam Rajagopal
Department of Electrical & Computer Engineering, Louisiana State University, Baton Rouge, LA 70803, U.S.A.

Andriyan Bayu Suksmono
Dept. of Electrical Engineering, Institut Teknologi Bandung, Jl. Ganesha No. 10, Bandung 40132, Indonesia.

Mitsuo Takeda
Dept. Information and Communication Engineering, The University of Electro-Communications, 1-5-1 Chofugaoka Chofu, Tokyo 182-8585, Japan

Kazuo Yamanaka
Department of Systems Engineering, Faculty of Engineering, Ibaraki University, 4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan.

Yanwu Zhang
c/o Aware Inc., 40 Middlesex Turnpike, Bedford, MA 01730, U.S.A.



Complex-Valued Neural Networks

Theories and Applications

In recent years, complex-valued neural networks have widened the scope of application in optoelectronics, imaging, remote sensing, quantum neural devices and systems, spatiotemporal analysis of physiological neural systems, and artificial neural information processing. In this first-ever book on complex-valued neural networks, the most active scientists at the forefront of the field describe theories and applications from various points of view to provide academic and industrial researchers with a comprehensive understanding of the fundamentals, features and prospects of the powerful complex-valued networks.

ISBN 981-238-464-2

9 789812 384645

World Scientific
www.worldscientific.com
5345 hc