

University of New Hampshire

## University of New Hampshire Scholars' Repository

---

Doctoral Dissertations

Student Scholarship

---

Winter 2003

### A wavelet-based CMAC for enhanced multidimensional learning

Brian P. Kirk

*University of New Hampshire, Durham*

Follow this and additional works at: <https://scholars.unh.edu/dissertation>

---

#### Recommended Citation

Kirk, Brian P., "A wavelet-based CMAC for enhanced multidimensional learning" (2003). *Doctoral Dissertations*. 194.

<https://scholars.unh.edu/dissertation/194>

This Dissertation is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact [Scholarly.Communication@unh.edu](mailto:Scholarly.Communication@unh.edu).

**A WAVELET BASED CMAC FOR ENHANCED  
MULTIDIMENSIONAL LEARNING**

BY

Brian P. Kirk

B.S., University of Vermont, 1994  
M.S., University of New Hampshire, 1996

DISSERTATION

Submitted to the University of New Hampshire  
in partial fulfillment of  
the requirements for the degree of

Doctor of Philosophy

in

Engineering: Electrical

December 2003

UMI Number: 3111507

### INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

**UMI<sup>®</sup>**

---

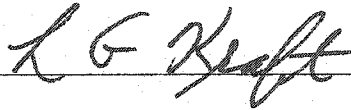
UMI Microform 3111507

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

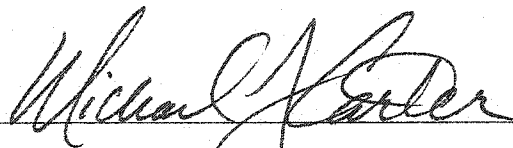
ProQuest Information and Learning Company  
300 North Zeeb Road  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

This dissertation has been examined and approved.



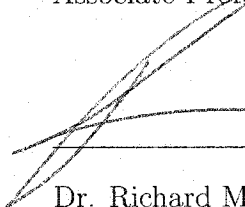
Dissertation Director, Dr. L. Gordon Kraft

Professor of Electrical Engineering



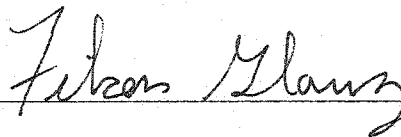
Dr. Michael J. Carter

Associate Professor of Electrical Engineering



Dr. Richard Messner

Associate Professor of Electrical Engineering



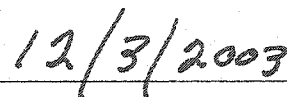
Dr. Filson Glanz

Professor Emeritus of Electrical Engineering



Dr. Kelly Black

Associate Professor of Mathematics



Date



# DEDICATION

To my wife, Kimberly.

# ACKNOWLEDGMENTS

This dissertation could not have been completed without the encouragement and assistance of many people. Most notable is Dr. Gordon Kraft, who not only supported me throughout the many years of my doctoral studies, but made the entire process one of the more enjoyable and rewarding experiences of my life. My deepest gratitude extends to the committee members (Dr. Michael Carter, Dr. Filson Glanz, Dr. Richard Messner and Dr. Kelly Black) for helping me through this process and for all of your insightful suggestions on completing this dissertation. To the entire ECE department, I not only thank you for the assistantships that helped make this degree possible, but for my entire academic experience at The University of New Hampshire. I would also like to thank my parents, who support and encourage me in all endeavors, and especially my father, who helped keep me focused on completing this degree. Finally, to my wife Kimberly, who sacrificed an exhaustive amount of time and energy in supporting me throughout this process, I could not have completed this dissertation without you.

# TABLE OF CONTENTS

Dedication . . . . .	iii
Acknowledgments . . . . .	iv
Abstract . . . . .	xvi
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>6</b>
2.1 Neural Networks . . . . .	6
2.2 A Brief History of Neural Networks . . . . .	7
2.3 CMAC . . . . .	9
2.3.1 Applications . . . . .	9
2.3.2 Extensions to the Original CMAC . . . . .	10
2.4 Convergence . . . . .	12
2.5 Wavelets, Filter Banks and CMAC . . . . .	14
<b>3 Overview of CMAC</b>	<b>15</b>
3.1 Introduction . . . . .	15
3.2 Conceptual Albus Model . . . . .	15
3.2.1 Structure . . . . .	15
3.2.2 Learning . . . . .	21
3.2.3 Receptive Fields . . . . .	21
3.2.4 Lattice Structure . . . . .	22
3.3 Computational Albus Model . . . . .	22

3.3.1	State-space to Weight-space Mapping . . . . .	22
3.3.2	Output Generation and Weight Training . . . . .	24
3.4	Extensions of the Albus CMAC Model . . . . .	25
3.4.1	Lattice Structures . . . . .	25
3.4.2	Receptive Field Shape . . . . .	26
3.4.3	Learning Algorithms . . . . .	27
3.4.4	Memory Hashing . . . . .	30
3.5	Stability Issues . . . . .	31
<b>4</b>	<b>Spectral Properties of CMAC</b>	<b>33</b>
4.1	Introduction . . . . .	33
4.2	One-Dimensional Spectral Model . . . . .	34
4.2.1	Albus Rectangular Receptive Field Model . . . . .	34
4.2.2	Linear Receptive Field Model . . . . .	39
4.2.3	Spline Receptive Field Model . . . . .	42
4.2.4	Quantization Effects in One-Dimension . . . . .	44
4.3	Higher-Dimension Problems . . . . .	49
4.3.1	Receptive Field Shape Effects . . . . .	49
4.3.2	Decimation Effects of the Lattice Structure . . . . .	51
4.4	Mapping Functions into CMAC . . . . .	67
4.5	Discussion . . . . .	69
<b>5</b>	<b>The Wavelet Based CMAC</b>	<b>70</b>
5.1	Introduction . . . . .	70
5.2	The Wavelet . . . . .	71
5.3	One-Dimensional Wavelet Based CMAC . . . . .	72
5.3.1	Quantization Effects in One-Dimension . . . . .	75
5.4	Multidimensional WCMAC . . . . .	80

5.4.1	Multidimensional Decimation and Lattice Structure . . . . .	87
5.5	WCMAC Mathematical Model . . . . .	95
5.6	Discussion . . . . .	97
<b>6</b>	<b>Extensions to the WCMAC</b>	<b>98</b>
6.1	Introduction . . . . .	98
6.2	Asymmetric WCMAC . . . . .	99
6.2.1	Input Quantization for WCMAC models . . . . .	104
6.3	Wide or Full Bandwidth WCMAC Models . . . . .	106
6.4	Discussion . . . . .	110
<b>7</b>	<b>Stability and Convergence Properties of WCMAC</b>	<b>112</b>
7.1	Introduction . . . . .	112
7.2	CMAC Stability . . . . .	113
7.2.1	WCMAC stability . . . . .	115
7.2.2	Reduced Computation WCMAC learning . . . . .	116
7.3	Convergence . . . . .	118
7.3.1	Reduced Computational Model vs. WCMAC Model . . . . .	118
7.3.2	Previous CMAC Models vs. WCMAC Model . . . . .	120
7.4	Discussion . . . . .	123
<b>8</b>	<b>Conclusions and Suggestions for Future Work</b>	<b>124</b>
8.1	Summary . . . . .	124
8.2	Conclusions . . . . .	127
8.3	Suggestions for Future Work . . . . .	128
<b>A</b>	<b>Hardware Development Paper</b>	<b>137</b>

# LIST OF FIGURES

2-1	McCulloch-Pitts Neuron . . . . .	7
3-1	Conceptual View of CMAC . . . . .	16
3-2	Two-Input CMAC Implementation . . . . .	17
3-3	Generalization Planes . . . . .	19
3-4	Single Layer Receptive Field Centers . . . . .	20
3-5	Receptive Field Centers . . . . .	20
4-1	Albus Receptive Field (1D) . . . . .	36
	(a) State Domain . . . . .	36
	(b) Frequency Domain . . . . .	36
4-2	CMAC Convergence for Different Harmonics (1D, Albus Receptive Field) .	38
	(a) Generalization of 4 . . . . .	38
	(b) Generalization of 10 . . . . .	38
4-3	Linear Receptive Field (1D) . . . . .	40
	(a) State Domain . . . . .	40
	(b) Frequency Domain . . . . .	40
4-4	CMAC Convergence for Different Harmonics (1D, Linear Receptive Field) .	41
	(a) Generalization of 4 . . . . .	41
	(b) Generalization of 10 . . . . .	41
4-5	Spline Receptive Field (1D) . . . . .	43
	(a) State Domain . . . . .	43
	(b) Frequency Domain . . . . .	43

4-6	CMAC Convergence for Different Harmonics (1D, Spline Receptive Field) .	43
(a)	Generalization of 4 . . . . .	43
(b)	Generalization of 10 . . . . .	43
4-7	Decimation without Aliasing . . . . .	45
(a)	Original Spectrum . . . . .	45
(b)	Spectrum After Decimation . . . . .	45
4-8	Decimation Causing Aliasing . . . . .	46
(a)	Original Spectrum . . . . .	46
(b)	Spectrum After Decimation . . . . .	46
4-9	Spectrum of Albus Receptive Field with Quantization . . . . .	47
4-10	The Effect of Input Quantization (1D, Albus Receptive Field) . . . . .	48
(a)	Quantized Simulation Data . . . . .	48
(b)	Simulation Data w/o Quantizing . . . . .	48
4-11	The Effect of Input Quantization (1D, Linear Receptive Field) . . . . .	49
(a)	Spectral Components . . . . .	49
(b)	Simulation Data . . . . .	49
4-12	Albus Receptive Field (2D, C=10) . . . . .	50
(a)	State Domain . . . . .	50
(b)	Frequency Domain . . . . .	50
4-13	Albus Receptive Field (2D, C=4) . . . . .	51
(a)	State Domain . . . . .	51
(b)	Frequency Domain . . . . .	51
4-14	Linear Receptive Field (2D, C=10) . . . . .	52
(a)	State Domain . . . . .	52
(b)	Frequency Domain . . . . .	52
4-15	Linear Receptive Field (2D, C=4) . . . . .	52
(a)	State Domain . . . . .	52

(b) Frequency Domain . . . . .	52
4-16 Lattice Structure (Receptive Field Placement) . . . . .	53
(a) Albus Lattice . . . . .	53
(b) An Lattice . . . . .	53
4-17 The Effect of weight-space Decimation (2D, C=4, Albus Receptive Field and An Lattice) . . . . .	56
(a) Base Spectrum . . . . .	56
(b) Decimated Spectrum . . . . .	56
(c) Aliasing Spectrum . . . . .	56
(d) Total Decimated Spectrum . . . . .	56
4-18 Simulation Data for Albus Receptive Field Shape (2D, C=4) . . . . .	58
(a) Albus Lattice . . . . .	58
(b) An Lattice . . . . .	58
4-19 The Effect of weight-space Decimation (2D, C=4, Linear Receptive Field and An Lattice) . . . . .	59
(a) Base Spectrum . . . . .	59
(b) Decimated Spectrum . . . . .	59
(c) Aliasing Spectrum . . . . .	59
(d) Total Decimated Spectrum . . . . .	59
4-20 Simulation Data for Linear Receptive Field Shape (2D, C=4) . . . . .	60
(a) Albus Lattice . . . . .	60
(b) An Lattice . . . . .	60
4-21 The Effect of weight-space Decimation (2D, C=10, Albus Receptive Field and Albus Lattice) . . . . .	61
(a) Base Spectrum . . . . .	61
(b) Decimated Spectrum . . . . .	61
(c) Aliasing Spectrum . . . . .	61



(d) Total Decimated Spectrum . . . . .	61
4-22 The Effect of weight-space Decimation (2D, C=10, Albus Receptive field and An Lattice) . . . . .	62
(a) Base Spectrum . . . . .	62
(b) Decimated Spectrum . . . . .	62
(c) Aliasing Spectrum . . . . .	62
(d) Total Decimated Spectrum . . . . .	62
4-23 Simulation Data for Albus Receptive Field Shape (2D, C=10) . . . . .	63
(a) Albus Lattice . . . . .	63
(b) An Lattice . . . . .	63
4-24 The Effect of weight-space Decimation (2D, C=10, Linear Receptive Field and Albus Lattice) . . . . .	64
(a) Base Spectrum . . . . .	64
(b) Decimated Spectrum . . . . .	64
(c) Aliasing Spectrum . . . . .	64
(d) Total Decimated Spectrum . . . . .	64
4-25 The Effect of weight-space Decimation (2D, C=10, Linear Receptive Field and An Lattice) . . . . .	65
(a) Base Spectrum . . . . .	65
(b) Decimated Spectrum . . . . .	65
(c) Aliasing Spectrum . . . . .	65
(d) Total Decimated Spectrum . . . . .	65
4-26 Simulation Data for Linear Receptive Field Shape (2D, C=10) . . . . .	66
(a) Albus Lattice . . . . .	66
(b) An Lattice . . . . .	66
4-27 Sine Wave . . . . .	67
(a) Time Domain . . . . .	67

(b) Frequency Domain . . . . .	67
4-28 Time-Delay CMAC Model . . . . .	68
4-29 Time Delayed Sine Wave . . . . .	68
(a) State-Space Domain . . . . .	68
(b) Frequency Domain . . . . .	68
5-1 Wavelet Receptive Field in One-Dimension . . . . .	73
(a) State Domain . . . . .	73
(b) Frequency Domain . . . . .	73
5-2 WCMAC Convergence for Different Harmonics (1D, C=9) . . . . .	75
(a) Modulation of 1 . . . . .	75
(b) Modulation of 1.4 . . . . .	75
5-3 WCMAC Convergence for Different Harmonics (1D, C=20) . . . . .	76
(a) Modulation of 1.1 . . . . .	76
(b) Modulation of 1.4 . . . . .	76
5-4 Quantization Effects in One-Dimension of CMAC vs. WCMAC . . . . .	77
(a) No Quantization . . . . .	77
(b) CMAC with Quantization . . . . .	77
(c) WCMAC with Quantization . . . . .	77
5-5 WCMAC Convergence for Different Quantizations (1D, C=20, M=1.1) . . . . .	78
(a) Quantization of 1 . . . . .	78
(b) Quantization of 3 . . . . .	78
5-6 WCMAC Convergence for Different Quantizations (1D, C=20, M=1.9) . . . . .	79
(a) Quantization of 1 . . . . .	79
(b) Quantization of 5 . . . . .	79
5-7 WCMAC Convergence with Aliasing Problems (1D, C=10, M=1.4) . . . . .	80
(a) Quantization of 1 . . . . .	80

(b) Quantization of 2 . . . . .	80
5-8 WCMAC Convergence Across Entire Spectrum (1D, C=25, Q=5) . . . . .	81
5-9 Wavelet Receptive Field in Two-Dimensions (C=11, M=1.9) . . . . .	84
5-10 Wavelet Receptive Field Spectrum and Convergence (2D, C=11, M=1.9) . .	85
(a) Spectrum . . . . .	85
(b) Convergence . . . . .	85
5-11 Wavelet Receptive Field in Two-Dimensions (C=11, M=1) . . . . .	86
(a) State Domain . . . . .	86
(b) Frequency Domain . . . . .	86
5-12 Wavelet Receptive Field in Two-Dimensions (C=11, M=1.4) . . . . .	86
(a) State Domain . . . . .	86
(b) Frequency Domain . . . . .	86
5-13 WCMAC Convergence for Different Harmonics (2D, C=11) . . . . .	87
(a) Modulation of 1 . . . . .	87
(b) Modulation of 1.4 . . . . .	87
5-14 Quantization Effects in Two-dimension of CMAC . . . . .	88
(a) No Quantization . . . . .	88
(b) Quantization of 2 . . . . .	88
5-15 Quantization Effects in Two-dimension of WCMAC . . . . .	89
(a) No Quantization . . . . .	89
(b) Quantization of 2 . . . . .	89
5-16 The Effect of weight-space Decimation (2D, C=11, Q=4, M=1.0) . . . . .	91
(a) Base Spectrum . . . . .	91
(b) Decimated Spectrum . . . . .	91
(c) Aliasing Spectrum . . . . .	91
(d) Total Decimated Spectrum . . . . .	91
5-17 WCMAC Convergence for Different Quantization (2D, C=11, M=1) . . . .	92

(a)	Quantization of 1 . . . . .	92
(b)	Quantization of 4 . . . . .	92
5-18	The Effect of weight-space Decimation (2D, C=11, Q=2, M=1.9) . . . . .	93
(a)	Base Spectrum . . . . .	93
(b)	Decimated Spectrum . . . . .	93
(c)	Aliasing Spectrum . . . . .	93
(d)	Total Decimated Spectrum . . . . .	93
5-19	WCMAC Convergence for Different Quantization (2D, C=11, M=1.9) . . . . .	94
(a)	Quantization of 1 . . . . .	94
(b)	Quantization of 2 . . . . .	94
5-20	WCMAC Convergence Across the Entire Spectrum (2D, C=25, Q=5) . . . . .	95
6-1	Discrete Wavelet Transform Data Flow . . . . .	102
6-2	Decomposition in the Frequency Domain . . . . .	103
6-3	Symmetric WCAMC in the Frequency Domain (2D) . . . . .	103
6-4	Asymmetric WCAMC in the Frequency Domain (2D) . . . . .	104
6-5	Asymmetric WCMAC with Different Generalization Values (2D) . . . . .	105
6-6	Asymmetric WCMAC Weight Decimation (2D) . . . . .	106
6-7	WCMAC Convergence Across Entire Spectrum (1D, C=25, Q=5) . . . . .	107
6-8	Wide Band Model . . . . .	107
6-9	WCMAC Convergence with Full Bandwidth (1D, C=25, Q=5) . . . . .	109
6-10	CMAC Convergence with Linear Receptive Field (1D, C=25, Q=1) . . . . .	109
6-11	Spectrum of WCMAC with Combined Receptive Fields . . . . .	110
6-12	Convergence of WCMAC with Combined Receptive Fields . . . . .	111
7-1	Base System for CMAC Stability . . . . .	113
7-2	Full Computation WCMAC vs. Reduced Computational Model . . . . .	118
(a)	Full Computation . . . . .	118

(b) Reduced Computation . . . . .	118
7-3 Sample by Sample Convergence . . . . .	119
7-4 Albus Receptive Field Convergence . . . . .	120
7-5 Linear Tapered Receptive Field Convergence . . . . .	121
7-6 Wavelet Receptive Field Convergence . . . . .	122
7-7 Wavelet Receptive Field Convergence . . . . .	122

**ABSTRACT**  
**A WAVELET BASED CMAC FOR ENHANCED**  
**MULTIDIMENSIONAL LEARNING**

by

Brian P. Kirk  
University of New Hampshire, December, 2003

The CMAC (Cerebellar Model Articulation Controller) neural network has been successfully used in control systems and other applications for many years. The network structure is modular and associative, allowing for rapid learning convergence with an ease of implementation in either hardware or software. The rate of convergence of the network is determined largely by the choice of the receptive field shape and the generalization parameter. This research contains a rigorous analysis of the rate of convergence with the standard CMAC, as well as the rate of convergence of networks using other receptive field shape. The effects of decimation from state-space to weight space are examined in detail. This analysis shows CMAC to be an adaptive lowpass filter, where the filter dynamics are governed by the generalization parameter. A more general CMAC is derived using wavelet-based receptive fields and a controllable decimation scheme, that is capable of convergence at any frequency within the Nyquist limits. The flexible decimation structure facilitates the optimization of computation for complex multidimensional problems. The stability of the wavelet-based CMAC is also examined.

# CHAPTER 1

## INTRODUCTION

Artificial Neural Networks (ANN) have continued to increase in prevalence since their inception during the 1950s[1]. The areas of control systems, pattern recognition and other types of signal processing are three areas in which ANN are now ingrained. The connection to these areas can be related to how the human brain performs the same tasks. The brain is responsible for the coordinated movement of hundreds of muscles with real-time processing of feedback information from the inner ear and other sensory inputs. In parallel, the brain continually processes images through the eyes and sounds through the ears. The computational power of the brain is astonishing. The brain can perform complex perceptual recognition tasks (e.g., recognizing a familiar face in an unfamiliar scene) in a time period on the order of 100 milliseconds. Tasks of a much simpler complexity take days to complete on a huge conventional computer[2]. This human computation is performed with neurons that are only capable of responding in the millisecond range, while computers have transistors that respond in the picosecond range[3]. The speed disadvantage of neurons is minimized by the massively parallel, complex interconnections that link billions of these nonlinear processing elements together. It is no wonder the human brain is continually modeled for its computation and control abilities.

The ability to adaptively learn linear and nonlinear systems has given rise to ANN in control systems. There are two major components that limit classical control systems: nonlinear effects and the ability to correctly model the system to be controlled. Neural networks have the ability to overcome both of these limitations. A wide range of ANN has been

adopted for control applications. The focus of this research is one such network, the CMAC (Cerebellar Model Articulation Controller or Cerebellar Model Arithmetic Computer) neural network, which has been successfully implemented in a range of applications around the world with a concentration of work at the University of New Hampshire in robotics, vibration control and pattern recognition. The CMAC concept is a relatively simple and elegant structure that is comparatively easy to implement and has a low computational cost as contrasted with traditional multi-layer perceptron-based neural networks. In addition to the reduced computation, CMAC converges at a much higher rate. In fact, it typically converges orders of magnitude faster than a multi-layer perceptron trained with the back-propagation algorithm. These properties have made CMAC a popular choice for real-time applications in both control systems and pattern recognition.

In this dissertation, the rate of convergence of the CMAC neural network is rigorously examined for a broad class of functions. The initial research demonstrates the strong coupling between the choice of generalization parameter and the effective frequency range of the model. Due to this effect, the current CMAC implementations only allow low frequency information to be widely generalized and have severe limitations in the upper half band of the frequency spectrum. A new wavelet-based CMAC (WCMAC) is formulated that has the capability to learn bandlimited signals anywhere in the frequency range supported by the target function. A proof of stability and the convergence rates comparing the WCMAC to traditional CMAC models are also presented.

Neural Networks are typically divided into two categories: globally generalizing or locally generalizing. The CMAC concept is an associative neural network that only considers a small number of related weights to compute an output and learn a function. It is therefore a locally generalizing network. This local generalization helps CMAC converge faster and reduces the computational load. However, it also has a set of disadvantages. Since the network does not consider each training sample on a global basis, large regions of untrained state-space can exist. An input that excites this untrained region will produce an



uncorrelated or unpredictable output. Furthermore, the CMAC output is not necessarily a continuous function of the input, since the training sets may be sparsely spaced. This can produce abrupt changes of the output near the edges of training set clusters.

Another feature of CMAC, which enables the high rate of convergence and low computational load, is the static and sparse nature of the receptive field structure. The static nature allows very fast weight searching algorithms and removes the variability associated with adjustable receptive field centers. The sparse structure reduces the memory requirements on the weight-space and spreads the function approximation across a larger region of the input space. Again, this is offset by some disadvantages; since the weight-space is, in essence, a decimated state-space, there exists a set of functions that is orthogonal to the basis function defined by the receptive field function and weight-space. These functions can therefore never be learned by CMAC. As the generalization increases, the sparsity of the weights in the input-space increases and the number of unlearnable functions increases simultaneously. However, the sparse weight-space has another major advantage, besides the reduced memory requirements. It reduces the computational requirements and facilitates the processing of large multi-dimensional problems. The one-dimensional CMAC is a special case in that a weight exists for every input, or state value. Its learning ability is limited by the frequency response of the receptive field, rather than the decimation effects.

All current implementations use averaging techniques which effectively lowpass filter the output, making higher frequencies bands orthogonal and unable to be learned. The rate of convergence also has a dependency on frequency, in that the higher the frequency of the target function, the slower the convergence rate of the network. Another artifact of the structure is that sharpness—or filter order—is tied directly to the generalization parameter and is not up to the control of the user.

Although many of the filtering properties of CMAC are easily identifiable through simulation and are qualitatively recognized, there lacks a formality in documenting these effects. Furthermore, these filtering effects are highly coupled to the rate of convergence of the net-

work and these properties lack any documentation beyond the stability range of the learning coefficient, which is well documented as existing between 0 and 2. An entire chapter of this dissertation is dedicated to analyzing the rate of convergence of CMAC, considering multiple receptive field shapes and different weight structures.

Based on the convergence analysis represented here, and in keeping with the advantages of CMAC (low computational load and rapid convergence), a new CMAC structure called the Wavelet-based CMAC (WCMAC) is proposed that is capable of learning a larger range of frequencies, that are not limited to the lowpass frequency band. Furthermore, the generalization parameter only controls the coverage of the state-space, while the decimation is controlled by a separate quantization step. This allows the user to optimize the amount of computation versus the necessary weight storage. The receptive fields are based on wavelets, which allow the effective frequency band of the network to be adjusted. CMAC has been represented as analogous to a filter bank. Wavelet research arose from multi-resolution analysis and filter bank theory. Therefore, the wavelet and CMAC concepts merge naturally.

The second chapter presents pertinent literature that is essential to the formulation of the research throughout this dissertation. This includes the following: a historical perspective of neural networks and CMAC with an emphasis on convergence and learning capabilities, an assortment of applications of CMAC, some relevant papers on multiresolution analysis, filter banks and wavelets. The third chapter is a detailed discussion of the CMAC neural network and its significant properties. The advances in learning algorithms and design of receptive fields that have occurred since Albus's original work are also included in this chapter. Chapter 4 is a Fourier type analysis of CMAC for multidimensional problems employing different receptive field shapes and weight placements. This chapter focuses on the bandwidth of different CMAC implementations. From the basic properties illustrated in the previous chapter, the fifth chapter formulates the WCMAC and illustrates its advantages across a range of problems. An additional chapter is dedicated to extending the WCMAC model to include features associated with the Generalized CMAC model and

the coupling of multiple WCMAC models into a single system. The question of stability and some computational improvements are examined in the seventh chapter. The final chapter summarizes the results and presents a framework for future research.

# CHAPTER 2

## BACKGROUND

### 2.1 Neural Networks

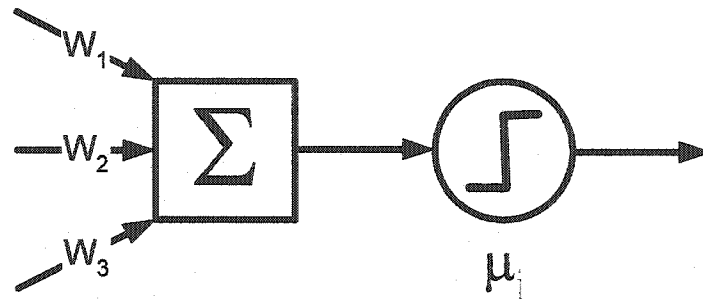
One definition for a neural network is the following[4]:

A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process.
2. Interneuron connection strengths known as synaptic weights are used to store the knowledge.

The human brain is the essential neural network and the driving force behind the discipline of artificial neural networks. Given the brain's ability to learn complex tasks, store that information for use at a later time, complete multiple tasks simultaneously and do all of this at speeds greater than the world's fastest supercomputers, it is no wonder researchers try to mimic its abilities.

There are at least two different disciplines studying the computational and decision making power of the brain: Artificial Intelligence and Neural Networks. Although there is no strict boundary between these areas, they can be classified with the following concepts. Artificial Intelligence (AI) is focused on the high-level decision making and reasoning capabilities of the human brain[5]. These high-level tasks include but are not limited to:



*Figure 2-1: McCulloch-Pitts Neuron*

perception, language, and problem solving[3]. The low-level tasks, such as muscle control and pattern recognition, are the concern of Artificial Neural Networks (ANN) researchers. Typically, ANN are built from simple elementary models, neurons, that can be assembled into a complex system, a neural network. Haykin describes artificial intelligence as algorithms and data representation in a top-down fashion, while ANN work as parallel distributed processors in a bottom-up fashion[3]. This dissertation is solely concerned with ANN and, in particular, the CMAC neural network.

## 2.2 A Brief History of Neural Networks

The foundation for the study of neural networks was laid by the work of McCulloch and Pitts in 1943[6]. They developed the first mathematical model of the biological process of the neuron, including inhibitory and excitatory connections, and threshold-based activation (figure 2-1). The next major step was the introduction of a learning rule by Hebb based on synaptic adjustments and has been deemed the Hebbian Learning Rule[7].

The study of neural networks gained great momentum with the introduction of the perceptron by Rosenblatt in 1958[8]. He followed up his initial publication with the "Prin-

ciples of Neurodynamics" in 1962[9]. This publication included the *perceptron convergence theorem*, which was the first well-developed theoretical model of a neuron and its linear separation capabilities. In the late 1950's, Widrow began his research into "Adaptive Sampled-Data Systems" which led to the introduction of the adaptive linear element, known as ADALINE[10]. In the early 1960's, Widrow and Hoff introduced a least squares gradient learning algorithm for the ADALINE model[11]. The natural extension to this single neuron was MADALINE, which had multiple interconnected models[12].

As the capabilities of neural networks expanded and the theoretical work appeared to back it up, the field looked to grow with unbounded interest. Unfortunately, in 1969, Minsky and Papert published a book on perceptrons that mathematically explained the fundamental limits of the single layer perceptron and its inability to learn certain functions[13]. Without an effective learning algorithm for multi-layer networks, neural networks seemed to have reached their limit. Consequently, research and interest died off quickly.

Although interest had waned, some important work progressed during the 1970's. The work of Kohonen on self-organizing maps was first published during this decade[14]. Unfortunately, some very important work was overlooked, including a learning algorithm for the multilayer perceptrons, which could overcome the limitations of the single-layer model. Werbos developed the learning algorithm in 1974[15], but it went unnoticed until 1986 when Rumelhart, Hinton and Williams published their work on the back-propagation algorithm[16].

In the 1980's, Grossberg formulated his adaptive resonance theory (ART) from his earlier work[17] and Hopfield published his work on recurrent neural networks, which are known now as *Hopfield Networks*[18]. The single layer neural network based on the radial basis function was proposed as an alternative to the multilayer perceptron[19].

## 2.3 CMAC

Another major development in neural networks that occurred in the 1970's and was overlooked for more than 10 years was the Cerebellar Model Arithmetic Computer (CMAC), also called the Cerebellar Model Articulation Controller. The CMAC idea was developed by Albus to model the functionality of the cerebellum, and he continued to publish on the subject from 1972 through 1981[20, 21, 22, 23, 24, 25, 26, 27]. The cerebellum controls neuromuscular and coordinated movements throughout the body. The CMAC concept was initially planned as a controller for artificial limbs and has been highly adopted in the areas of control and robotics. The CMAC idea uses associative memory structures that mimic the functionality of the cerebellum. The concept does not directly fit the connectionist model of weighted interconnects typical to ANN systems. It is conspicuously absent from many neural network texts, possibly for this reason. However, many texts also include radial basis function neural networks that represent a similar structure.

The CMAC model was adopted by Miller at the University of New Hampshire for adaptive control in robotic application[28]. This was followed by papers that outlined the different properties of CMAC including an evaluation of CMAC as an alternative to traditional backpropagated multilayer neural networks[29], comparisons to other control techniques[30, 31, 32], hardware implementations[33, 34, 35, 36], the effective fault-tolerance of the network[37], slow learning convergence at selected frequencies[38] and general overviews of the network[39]. With an increase in theory and understanding of the network, the applications of the network started to grow.

### 2.3.1 Applications

There are a few major properties that have led to widespread implementation of CMAC. They are the ability to compute the network response in real-time for many control system applications, the relative ease of implementation in terms of complexity, and the ability to

use standard digital systems (either low-level hardware or standard personal computers) for the implementation.

The following citations represent some of the implementations, research articles and applications of CMAC at the University of New Hampshire:

1. Robotics [40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]
2. Digital filtering [51, 52]
3. Shape recognition [39, 53]
4. Vibration control [54, 55, 56]

Obviously, there have been other significant applications of CMAC outside of the University of New Hampshire. A small subset of papers which are commonly referenced and some recent applications are listed below:

1. Robotics [57, 58]
2. Image processing [59, 60]
3. Motor control [61]
4. Fuel injection systems [62]
5. Wheel chair control [63]

This list represents a very small portion of the CMAC applications to date and represents areas that are of particular interest to the author of this dissertation.

### **2.3.2 Extensions to the Original CMAC**

As further understanding of the exact modeling properties of CMAC advanced and information was gained through actual implementations, some limitations became apparent. The



original CMAC used a binary weight activation function, which either includes or excludes a weight from the output formulation. This abrupt change caused a staircase-shaped output, which induced a host of limitations. Alternative weight activation functions (which are also referred to as receptive field shapes) have been explored [64, 65, 66, 67]. Techniques to improve the derivative of the output, which was typically lost by the binary activation function, have also been explored [5, 54, 68]. Another algorithm that has been used at the University of New Hampshire since the mid 1990's is weight normalization, effectively binding values of the weights to the function directly and eliminating their possible unbounded growth. This technique was published independently in 2002 under the topic of *improved generalization properties* [69]. Similar methods for controlling the growth of weights has been explored for other artificial neural networks under the context of *learning with weight constraints and regularization theory* [70].

A major feature of CMAC is the ability to reduce the total number possible inputs to a manageable number of storage elements or weights. The original algorithm placed the weights along the hyperdiagonal, which was not always the optimal structure. New static placement techniques for the weights were explored [65, 71, 72]. Recently, CMAC has also been coupled with genetic algorithms to produce an adaptive approach to the problem of receptive field placement [73], although this method is computationally intensive. Recent applications have adopted the method of using different degrees of generalization for each input coordinate [74]. This model is known as the Generalized CMAC (GCMAC).

Recent research on the structure of CMAC uses hierarchic models, i.e. multiple CMACs linked together, just as multiple perceptrons are linked together, in an attempt to solve higher-order problems with a simple base structure. Typically, a one dimensional CMAC [75, 76] is used as the basic building block. The concept of a hierarchic CMAC is effectively another approach to adaptive receptive field placement [77].

## 2.4 Convergence

Theoretical work on the proof of learning convergence started to be published in the late 1980s and continues through the time of publication for this dissertation. The convergence proofs are discussed in two different contexts: open-loop and closed-loop. Since CMAC is a supervised-learning system where the error in the output is fed back to the weight-space during the adjustment process, this essentially makes it a closed loop system. However, to comply with existing research the following definitions are used:

1. Open-loop – connecting and training the CMAC in parallel with the system model.
2. Closed-loop – CMAC is placed in the feedback loop. The input to CMAC is the system output and the control object is to drive the system response to zero.

The first open-loop paper was published by Ellison in 1988 that proved CMAC could find the optimal solution for a set of one dimensional linear equations[78]. Parks and Miltzer published their first CMAC convergence proof in 1989[79]. This paper proved two major results on CMAC learning convergence. First, if the weight space is large enough, the CMAC will converge to a unique weight vector and this is proven with a Lyapunov approach. The second case considered was the convergence to a limit cycle, if the physical memory size was effectively smaller than the number of weights that need to be stored. This proof examined the eigenvalues of the weight trajectory to see if they were either on or inside the unit circle, thus proving stability in the discrete time domain. The authors limited the CMAC learning coefficient, used in updating the weights, to unity in this initial paper. Parks and Miltzer followed up this paper with an investigation into the properties of five different learning algorithms[80]. Proofs of convergence were not developed.

Ellison extended his one dimensional proof to the multi-dimensional case in 1991[81]. Wong and Sideris formulated a matrix of the activated weights for corresponding inputs, thus creating a set of linear equations or a linear system[82]. A Gauss-Seidel iterative

scheme was used to prove convergence, and it was claimed the CMAC always converged to an arbitrarily small error when hashing is not present. They included comments about the effects of hashing, but gave no proof or simulation results. This detailed proof was only constructed for the one-dimensional CMAC. They claimed the results could be directly extrapolated to the multi-dimensional case, but this assertion was refuted by Brown and Harris[83][84]. Brown and Harris showed that the reduction of the weight-space from the address-space introduces a set of possible orthogonal functions that cannot map onto the weight-space and can never be learned[85][86]. In fact, it will be shown in chapter 4 of this document that the choice of receptive field shape also limits the number of unique functions that can be learned.

Wong continued his work in one dimension and identified the generalization parameter as the single most important factor in the rate of convergence[87]. As will be detailed later, the one-dimensional case has significantly different properties than the multi-dimensional cases. Similar results including multi-dimensional problems were demonstrated in simulation by An[65]. Serrano *et al* discussed the Fourier components and the Nyquist sampling effects when designing a Generalized CMAC (GCMAC)[88]. These spectral properties of CMAC are explored in depth in the fourth chapter of this document.

Campagna and Kraft[5] used an extension of the Parks and Miltzer analysis[79] to show that CMAC converges in a Lyapunov sense if the learning rate is between 0 and 2, as long as it can be postulated that a target set of weights exists. Lin and Chaing, in 1997, extended the original eigenvalue analysis to show that the learning is stable under the same condition[89].

Finally, Kraft and K. Liu proved the stability of CMAC in a closed-looped solution for vibration control with a variable time delay[90]. H. Liu proved the convergence of a closed-looped CMAC for a class of nonlinear dynamic systems[91].

## 2.5 Wavelets, Filter Banks and CMAC

In a recent publication, Horvath and Szabo[69] made the following assertion about CMAC:

The binary CMAC can be regarded as a complex piecewise linear filter, where the elementary filters are arranged in two layers. While the first layer contains one filter, the second layer consists of a filter bank.

Vetterli stated that there is a strong link between filter banks and wavelets[92]. He used the examples that filter banks can be used to generate wavelet bases[93], and filter banks can be used to calculate a wavelet series[94]. With CMAC essentially a bank of filters, one could postulate that CMAC and wavelets could be strongly linked. In fact, a new CMAC is derived from basic wavelet and filter bank theory that will extend the operational range of CMAC.

Wavelets have been used primarily for multi-resolution analysis, where the frequency spectrum at different times can be extracted. This methodology is widely taught in image processing to compress images and handle multiple focal points[95]. In chapter 4 it is shown that the CMAC network only has a limited bandwidth due to the structure and decimation from state-space to weight-space. All previous CMAC implementations only considered the low-pass component of the spectrum. Using basic wavelet structures defined in [96], a new CMAC is developed with far more flexibility in the effective spectrum of the model.

# CHAPTER 3

## OVERVIEW OF CMAC

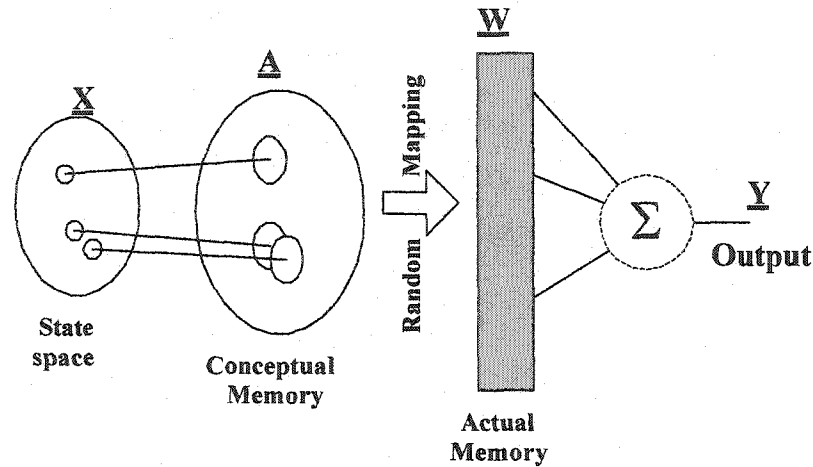
### 3.1 Introduction

The purpose of this chapter is to construct the principles of CMAC which are referred to in all future chapters. This chapter introduces CMAC from a conceptual and functional point of view. The mathematical or computational model that implements the original CMAC is then introduced. A large amount of research has already gone into improving and understanding CMAC under a variety of conditions. The relevant and important works that advanced the CMAC architecture are also included in this chapter.

### 3.2 Conceptual Albus Model

#### 3.2.1 Structure

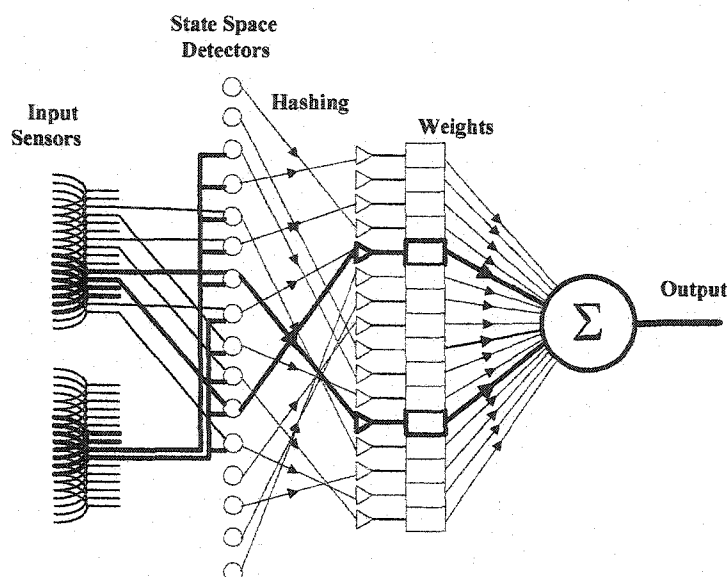
The CMAC network is an associative neural network that uses only a subset of the network's weights for the determination of any particular output. With only a small number of weights activated and used in the accumulation of the output, the network can very quickly formulate outputs, a clear advantage over many other networks. The training cycle is also extremely fast because only the same small subset of weights needs to be adjusted. The number of weights in the subset is always the same. These properties together result in a high-speed network that is also deterministic and ideal for control applications.



*Figure 3-1: Conceptual View of CMAC*

The associative properties of the network create local generalization—similar inputs give similar or correlated outputs; while distant inputs produce uncorrelated outputs. This local generalization can be seen in the conceptual view of CMAC, Figure 3-1. In general terms, the input to a CMAC is a point in a multidimensional space. This point is expanded upon, according to the generalization parameter, to force an overlapping in the conceptual memory. In the state-space region of figure 3-1, there exists two input states with values that are close but not equal. The conceptual memory for each of these points overlaps; consequently, these inputs share information and their outputs will have some level of correlation.

The inputs are mapped from the state-space to a region of the CMAC conceptual memory. Each region of conceptual memory mapped by the input contains a specific and constant number of weights that is equal to the generalization parameter. The weights are typically stored in a traditional memory structure and a pseudo-random code is used to translate the



**Figure 3-2:** Two-Input CMAC Implementation

conceptual memory address to the actual or physical memory. The weights associated with each input vector are accumulated to form the network output. The mapping structure (translation from conceptual memory to actual memory) and the generalization parameter are predetermined and held static. Adjusting the values within the weight vector produces the adaptive nature of the network.

Figure 3-2 shows how an actual CMAC can be implemented with two inputs. The first stage of the network quantizes the input values and then generalizes both inputs over a larger area of the state-space by activating the state-space detectors. The state-space detectors perform a logical AND function. Each state-space detector has a connection to each input dimension. When both of the connections are active, an associated weight is activated. The weight is consequently accumulated with other active weights to form the output value. The logical AND function forms the binary receptive field, since a weight is either included

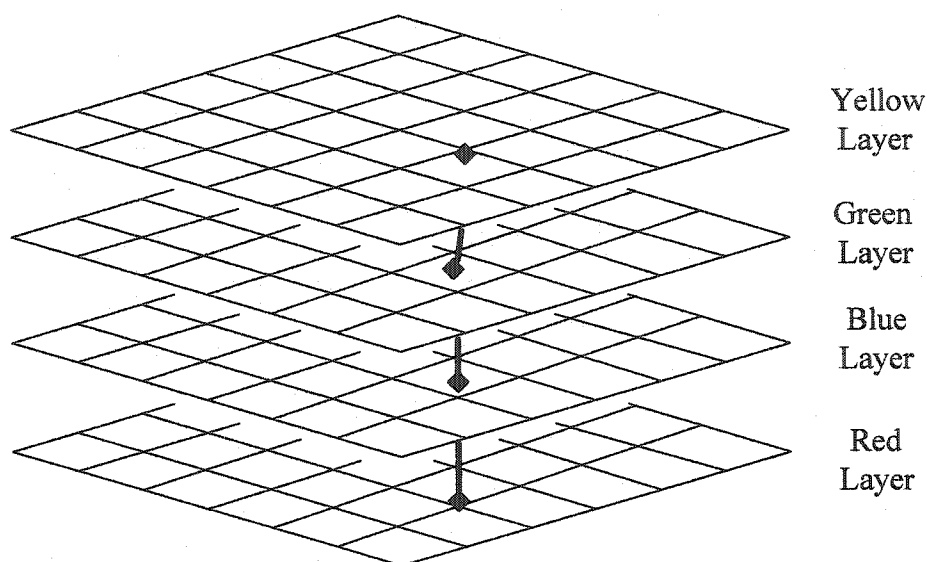
fully or excluded fully. It should be noted that figure 3-2 is an incomplete model, since four input states were activated and only two weights are shown accumulating. The actual implementation would accumulate four weights for this network. This is a simplification made to clarify the figure only.

There are a few other important properties of the translation from conceptual memory to actual memory. First, it has already been mentioned that the mapping from the state-space detectors to the weights is done in a pseudo-random fashion. The reason is that a multidimensional input can map to an extremely large space. For example, a three input system with twelve bit resolution will have 32 billion possible inputs. Most applications will only use a small subset of these possibilities. The random hashing allows us to map the extremely large input state-space to a much smaller memory structure. Since the memory system is smaller than the state-space, multiple conceptual memory addresses are mapped to the same physical location. This overlapping effect is known as a **collision**. A collision does not typically cause a catastrophic error in the output because the output is formed by the accumulation of multiple weights. The effect of the collision is dependent on the generalization parameter and the difference from the optimal weight for each unique solution.

The second important property in the translation from input space to the weight structure is the fact that only the number of weights equal to the generalization is activated. In figure 3-2, the generalization is four; therefore, each input is spread over four states and together they form an area of sixteen states, but only four of these map to weights. The placement of these receptive fields is determined by the generalization and a fixed lattice structure.

Figure 3-3 shows the organization of the generalization planes. Each distinct input is mapped exactly once to each generalization plane, activating one weight (receptive field center) per plane. Therefore, the number of generalization planes and weights is defined by the generalization parameter. Each of the activated weights is offset along the hyperdiag-

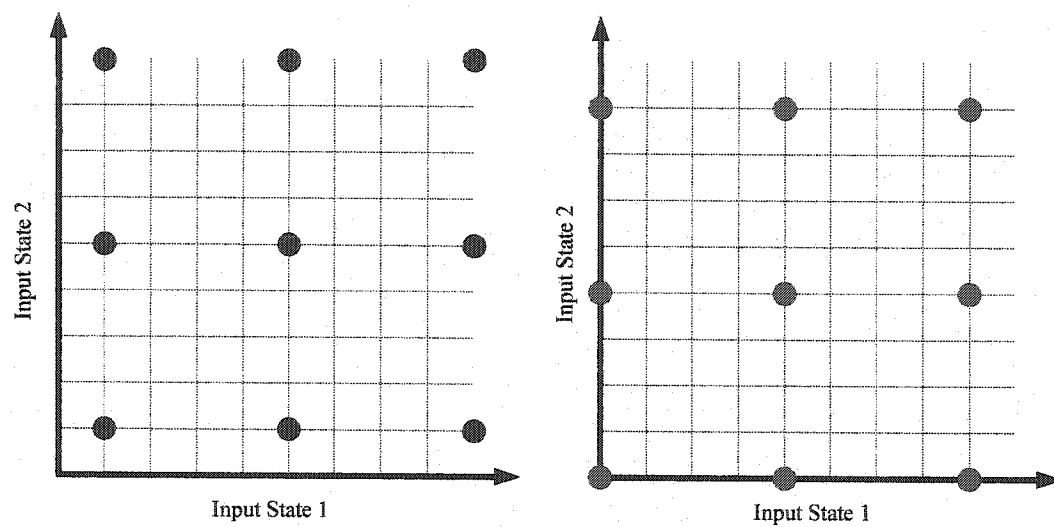




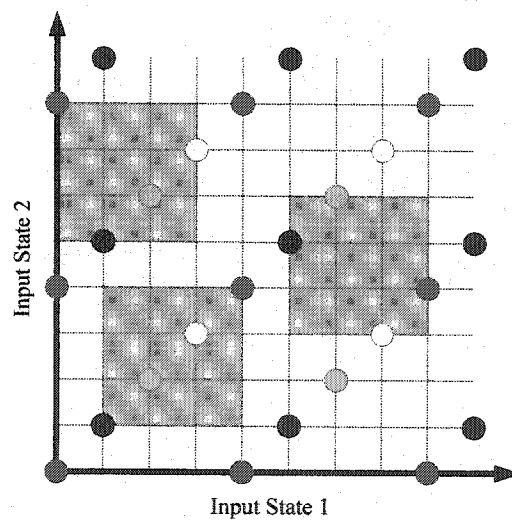
*Figure 3-3: Generalization Planes*

onals of the input space. A quantization level offsets each adjacent receptive field. Each layer in figure 3-3 is given a color notation to map to the receptive field centers in figure 3-4.

In simple cases, such as systems with two-dimensional inputs, it is possible to show all of the receptive field centers. Figure 3-5 shows all of the receptive field centers mapped onto the input state-space for a generalization of four. The shaded areas represent different inputs mapped onto these receptive fields. It can be seen that exactly four receptive fields are included in each shaded area. It is important to notice the reduction from the total number of possible inputs to the actual receptive field centers. This not only decreases the amount of memory needed, but the amount of computation to perform, since the number of weights for accumulation and adjustment has decreased. It also limits the number of solutions that can be uniquely modeled.



*Figure 3-4: Single Layer Receptive Field Centers*



*Figure 3-5: Receptive Field Centers*

In the Albus implementation of CMAC, each receptive field center is equally valued. It does not matter if the weight is a great distance from the center of the shaded generalization area. For networks with extremely large generalizations, like sixty-four, the weights at the fringes of the generalization region may have little correlation with the weights at the center of the region. In these cases, it may not be prudent to accumulate weights on an equal basis.

### 3.2.2 Learning

The standard CMAC network's ability to learn is done through an adjustment of the weight vector, which represents all of the receptive field centers. The state-space detectors control the weights to be adjusted on each training pair and also control the values to be used in the formulation of the output. As previously mentioned, the traditional Albus model uses only binary values for the field detectors. This means all weights are equally accumulated in the output and adjusted equally during the training cycle.

### 3.2.3 Receptive Fields

The initial Albus CMAC has a set of well-understood limitations. First, the binary weight activation scheme, as previously discussed, produces the piecewise or staircase functions. These discontinuous functions typically do a reasonable job of modeling the target function. However, the derivative of the function is poorly modeled due to the instantaneous changes in the output function where one weight is added and another is subtracted. Beyond modeling the derivative, the staircase function approximation introduces nonlinearities to the problem by generating new frequencies in the system. The result of introducing new frequencies in a control system may be unpredictable. An unwanted resonant frequency can severely disrupt a system's response.

Changing the receptive field shape from a rectangular function with binary values to a smooth, tapered function—or a Gaussian shape—can eliminate the staircase output. These functions are used to control the proportion of a weight's impact on the final output of

the network. Weights located close to the input vector are accumulated with the highest strength. The distance from the receptive field center can be calculated in variety of different manners. The three most common methods are the Euclidean method, the absolute minimum and the Manhattan. The traditional learning algorithm is also modified to adjust the weights according to their activation function.

### 3.2.4 Lattice Structure

As previously mentioned, the network has a reduced number of receptive field centers from the actual number of possible state-space values. This decimated receptive field structure is a static and defined lattice structure that was shown in figure 3-3. This particular lattice structure has limitations due to the uneven spacing of the receptive fields. Since the projection of the target function across the weight-space is typically not known, a symmetrically sampled weight-space is more desirable. The modular, static and repetitive nature of this lattice is important in the quick search of the activated weights, without having to analyze all of the weights in the system. Many other networks analyze all weights for each training pair; this significantly increases their computation time.

## 3.3 Computational Albus Model

A set of equations is set forth for each stage of the traditional CMAC. In this case, a vector notation is used, as it becomes more useful in the stability analysis.

### 3.3.1 State-space to Weight-space Mapping

The classic CMAC is activated by an  $N$ -dimensional state-space vector,

$$\mathbf{x}_s = \begin{bmatrix} x_{s1} & x_{s2} & \dots & x_{sn} \end{bmatrix} \quad (\text{Eqn 3.1})$$

where  $s$  is the specific target sample and the different values in the vector represent the different dimensions. The number of state-space dimensions is defined as  $n$ .

An important component of the CMAC method is reducing the solution space by quantization and generalization. There exists a quantization parameter for each axis of the state-space defined as  $\Delta_j$ . The quantization parameter reduces the solution space by reducing the number of discrete levels that access weights.

$$\mathbf{x}'_s = \left[ \text{int}\left(\frac{x_{s1}}{\Delta_1}\right) \quad \text{int}\left(\frac{x_{s2}}{\Delta_2}\right) \quad \dots \quad \text{int}\left(\frac{x_{sn}}{\Delta_n}\right) \right] \quad (\text{Eqn 3.2})$$

$$= \begin{bmatrix} x'_{s1} & x'_{s2} & \dots & x'_{sn} \end{bmatrix} \quad (\text{Eqn 3.3})$$

The generalization parameter is defined as the number of simultaneously excited receptive fields,  $C$ . The width of each receptive field is equal to  $C\Delta_j$  in the state-space and simply  $C$  in the quantized state-space. The hyperdiagonals which define the receptive field centers are also spaced by  $C$  along any axis of the state-space. This also significantly reduces the weight-space, by making the state-space to weight-space mapping even more sparse. The next step is to develop the weight-space indices for the excited state-space detectors. The equation for the address of the  $i^{\text{th}}$  weight is

$$\mathbf{A}_i = \begin{bmatrix} x'_{s1} - ((x'_{s1} - i)\%C) & x'_{s2} - ((x'_{s2} - i)\%C) & \dots & x'_{sn} - ((x'_{sn} - i)\%C) \end{bmatrix}, \quad (\text{Eqn 3.4})$$

$$\mathbf{A}_i = \begin{bmatrix} a_{i1} & a_{i2} & \dots & a_{in} \end{bmatrix} \quad (\text{Eqn 3.5})$$

where the index  $i$  references the  $C$  parallel layers of the receptive fields and  $\%$  represents the modulus operator.  $\mathbf{A}_i$  represents the location of a single receptive field in the normalized input space.

Each input vector excites exactly  $C$  receptive fields. Even after the reductions in weight-space by the generalization and quantization, there might still exist more weight addresses than are physically addressable or available in a computer's memory system. Therefore the hashing method,

$$\mathbf{A}'_i = h(\mathbf{A}_i), \quad (\text{Eqn 3.6})$$

is used to distribute the weights randomly throughout physically memory, where  $h(\cdot)$  represents any number of acceptable hashing techniques, like linear shift registers or psuedo-random lookup tables. Finally, the address are translated into the actual weight values

$$\mathbf{w}_s = \begin{bmatrix} W[A'_{s1}] & W[A'_{s2}] & \dots & W[A'_{sC}] \end{bmatrix} \quad (\text{Eqn 3.7})$$

where  $W[\cdot]$  represents the actual memory system or weight storage.

### 3.3.2 Output Generation and Weight Training

The output generation of the traditional CMAC is given as

$$\mathbf{y}_s = \frac{\mathbf{c}_s \mathbf{w}_s}{C}. \quad (\text{Eqn 3.8})$$

It is clearly evident that this equation is simply the accumulation of all the weights divided by the number of the weights, or the mean value of the weights. The vector  $\mathbf{c}_s$  is the activation function vector

$$\mathbf{c}_s = \begin{bmatrix} c_{s1} & c_{s2} & \dots & c_{sC} \end{bmatrix}, \quad (\text{Eqn 3.9})$$

$$\mathbf{c}_s = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}. \quad (\text{Eqn 3.10})$$

The Albus CMAC used only binary values, and this model is only accessing the activated weights through the address scheme given. Therefore,  $\mathbf{c}_s$  is simply a row vector of ones. The weight update algorithm for the binary CMAC is given as

$$\Delta \mathbf{w}_s = \frac{\alpha}{C} \mathbf{c}_s^T (\hat{y}_s - y_s), \quad (\text{Eqn 3.11})$$

$$= \frac{\alpha}{C} \mathbf{c}_s^T (\hat{y}_s - \mathbf{c}_s \mathbf{w}_s), \quad (\text{Eqn 3.12})$$

where  $\alpha$  is the learning rate, typically set between 0 and 1, and  $\hat{y}_s$  is the desired response of the network. This weight update method is considered a supervised learning algorithm.

The weight is simply updated as the original weight plus the updated weight:

$$\mathbf{w}_{s+1} = \mathbf{w}_s + \Delta \mathbf{w}_s. \quad (\text{Eqn 3.13})$$

### 3.4 Extensions of the Albus CMAC Model

With many years of research since the inception of CMAC, there are some very important enhancements to the basic CMAC structure, particularly in terms of receptive fields, lattice structures and memory implementations. The learning algorithms associated with each of these models has also evolved to accommodate the advances. Theoretical work on the stability of CMAC, although mostly focused on the traditional CMAC model, has also evolved.

For the purpose of this dissertation, the structure of the network is defined as the receptive field shape, the lattice structure that identifies the centers of the receptive field, and finally, the memory implementation that maps the virtual CMAC weight to the actual memory location in the implementation.

#### 3.4.1 Lattice Structures

The conventional CMAC structure aligned all the receptive fields along the hyperdiagonals of the input space, where each input falls within the same number of receptive fields. However, this concentration of receptive fields is not ideal, due to the inhomogeneous placement. Therefore, the receptive field placement was modified based on the assignment of displacement vector, which defines the location of the next receptive field center. For the original reference hypercube center at origin, the receptive field placement is calculated by the displacement vector,

$$\mathbf{d} = \begin{bmatrix} d_1 & d_2 & \dots & d_n \end{bmatrix}, \quad (\text{Eqn 3.14})$$

such that the corner of the  $i$ th receptive field in the reference hypercube was

$$\begin{bmatrix} (d_1 i) \% C & (d_2 i) \% C & \dots & (d_n i) \% C \end{bmatrix}. \quad (\text{Eqn 3.15})$$

The traditional CMAC would have a displacement of the following form,

$$\mathbf{d}_{Albus} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}. \quad (\text{Eqn 3.16})$$

Using this lattice structure implementation, a set of heuristic rules was developed by An [65] to form a more uniform placement strategy. First, a set of integers is chosen such that they fall between 1 and  $C/2$ . These integers cannot be a factor of  $C$  or integer products of  $C$ , although the value of 1 can be used. If there are more than  $n$  candidates to choose, where  $n$  is the number of input dimensions,  $n$  are selected. In the case of less than  $n$  candidates, the generalization can be increased, allowing for a larger set of initial integers. Otherwise, a suboptimal solution can be used by repeating one of the values. For example, a generalization of 8 has displacement vector candidates of 1 and 3. The addresses are given by

$$\mathbf{A}_s = \begin{bmatrix} x'_{s1} - ((x'_{s1} - d_1i)\%C) & x'_{s2} - ((x'_{s2} - d_2i)\%C) & \dots & x'_{sn} - ((x'_{sn} - id_n)\%C) \end{bmatrix} \quad (\text{Eqn 3.17})$$

$$\mathbf{A}_s = \begin{bmatrix} a_{s1} & a_{s2} & \dots & a_{sn} \end{bmatrix} \quad (\text{Eqn 3.18})$$

This equation is only valid for positive-referenced receptive field centers, due to the properties of the modulus operator. An equation for the negative-referenced receptive fields can easily be derived. It should also be noted that this methodology of building the lattice structure from a combination of the displacement vector and generalization parameter can be replaced with a sampling matrix, which is defined by the base hypercube. The sampling matrix method is used in the evaluation of CMAC in the next chapter.

### 3.4.2 Receptive Field Shape

Many people have examined the concept of a variable or tapered receptive field shape. For the general case, the output takes on the form

$$\mathbf{y}_s = \frac{\mathbf{c}_s \mathbf{w}_s}{\mathbf{c}_s \mathbf{u}_s}, \quad (\text{Eqn 3.19})$$



where  $\mathbf{u}_s$  is a column vector of ones and of equal length to the vector  $\mathbf{c}_s$ . The variable receptive field is accommodated as

$$\mathbf{c}_s = \begin{bmatrix} f(\delta_{s1}) & f(\delta_{s2}) & \dots & f(\delta_{sC}) \end{bmatrix}. \quad (\text{Eqn 3.20})$$

Notice that setting  $f(\delta_i)$  to a binary vector of width equal to the generalization factor, the equation reduces to the standard Albus CMAC output. The receptive field function,  $f()$ , can take on many forms including linear, Gaussian or a variety of splines. The  $\delta_i$  represents distance functions of the input state vector to the activated receptive field center. This parameter can be defined in a host of different ways including: Euclidean, Manhattan or the shortest projection along any input state to the nearest receptive field center. For example, the equation for the Linear receptive field as implemented in the UNH CMAC follows

$$f_i(\delta) = (C/2 - \delta) * M_s, \quad (\text{Eqn 3.21})$$

where  $\delta$  is the minimum distance to the nearest receptive field center and  $M_s$  is a scaling factor used for the look-up table implementation. The actual implementation has some interesting values for low generalization width receptive fields.

### 3.4.3 Learning Algorithms

The learning algorithms have also been modified to handle changes in receptive field shape. A few other learning algorithms have also been derived for handling special case problems that arise when CMAC is applied.

#### Steepest Descent Weight Updating

The most popular learning algorithm for the advanced CMAC implementation is the steepest descent learning algorithm. The following equation is the weight update value for this algorithm

$$\delta \mathbf{w}_s = \alpha \mathbf{c}_s^T (\hat{y}_s - y_s) \frac{\mathbf{c}_s \mathbf{u}_s}{\mathbf{c}_s \mathbf{c}_s^T} \quad (\text{Eqn 3.22})$$

As will be discussed and analyzed, most of these advanced learning algorithms have not been analyzed with respect to their stability boundaries.

### Simplified Update Model

A simplified version of the above equation can also be used in training the CMAC networks with higher-order receptive fields.

$$\delta \mathbf{w}_s = \alpha \mathbf{c}_s^T (\hat{y}_s - y_s) \frac{1}{A}, \quad (\text{Eqn 3.23})$$

It will be shown that this method has a stability boundary that is easily derived. From a comparison of the two previous equations, it is clearly evident that this method has less complexity and computation, although the weight update may not be optimal. The output generation also needs to take the form of

$$y_s = \frac{\mathbf{c}_s \mathbf{w}_s}{A} \quad (\text{Eqn 3.24})$$

where  $A$  is an approximation of  $\mathbf{c}_s \mathbf{c}_s^T$ . Since the learning rate is typically set less than one, it will be shown in chapter 7 that as long as  $A$  is a reasonable approximation the system will remain stable. This simplified method is helpful in direct hardware implementations and in reducing the number of computations to perform.

### Weight Normalization

The output formulations for CMAC depend on the weighted average of multiple weights, as seen in Eqn 3.8. Therefore, an infinite number of vectors can be used to formulate any single output value. Due to the finite precision of the implementation, this is actually not the case, but there are still a large number of patterns that exist. A problem arises that weights may continually drift apart, such that their average never changes. Over a period of time, these weights can cause a host of problem due to their extreme values. For example, a weight might actually reach a saturation boundary and change sign. Additionally, another

training sample might be added to the series and adjacent weights might not have the dynamic range to offset the large drifted values of the other associated weights.

Uneven or sporadic sampling—especially with noise—can also cause problems, since part of a weight's rate of convergence depends on how often the weight is accessed. The learning rate is typically very low when noise is present to average out the random process effects. Therefore, the weights can have dramatically different values, based on the number of access times.

To correct these problems the following learning algorithm is used,

$$\delta \mathbf{w}_{si} = \alpha_1 L_x + \alpha_2 (\hat{y} - w_{si}) \quad (\text{Eqn 3.25})$$

where  $\delta \mathbf{w}_{si}$  is the value used to update each weight accessed at sample  $s$ ,  $L_x$  represents any of the previous learning algorithms discussed,  $i$  is the range of one to the generalization number and  $w_{si}$  are the individual weights accessed during that sample set. As can be seen from the equation, each weight is individually targeted to learn the desired output. This avoids the drifting weight problem, by not simply relying on the accumulation of weights to learn the target function.

### Weight Smoothing

As noted many times, the original binary receptive field caused stair-case outputs which have no derivative and introduce a broad spectrum of noise into the output. Beyond this basic point, the uneven sampling and sporadic repetition of the samples also causes discontinuous regions of the output. Therefore, a method of binding weights at the edge of the generalization region to their current value, and dynamically changing the receptive field shape, was developed by Campagna [5]. This method used optimal control theory and minimization of a quadratic error function. The concept of a dynamically changing receptive field has broad implications in a decimated system; since the bandwidth of the model can change instantaneously, the weight structure had to support this concept also. Although

this dissertation does not focus directly on weight smoothing, it is used multiple times as a discussion point.

### 3.4.4 Memory Hashing

For systems that have high numbers of inputs with large dynamic ranges, the possible number of inputs can range from billions to trillions or more, and the amount of generalization and quantization can drastically reduce the number of states. However, it is possible to still have enough states that it is impractical or impossible to provide memory for each individual weight. Since many applications only use a limited number of the possible inputs, it is typically not necessary to map all of these states. The concept of virtual addressing for weights is used to map a larger weight-space to a smaller memory system. This is also known as memory hashing or hash coding.

A weight index along any input axis is found, it selects a pseudorandom value and is accumulated with the pseudorandom values generated by the other axes or input states. Finally, that large random number is truncated by the size of the memory array allocated, typically using the modulus operator as in

$$A'_{si} = \left( \sum_{j=1}^n T_j [a_{sij} \% R_j] \right) \% M \quad (\text{Eqn 3.26})$$

where  $\%$  represents the modulus operator,  $T_j$  is a pseudorandom vector,  $R_j$  is the size of the table  $T_j$  and  $M$  is the size of the physical memory the system can access.

Occasionally, the truncation of the random value from one input state maps to the same truncated value of another input state. This is known as a **collision** which is defined as:

$$A'_n = A'_m \quad \text{for} \quad A_n \neq A_m. \quad (\text{Eqn 3.27})$$

The probability that a collision will not occur is dependent on the amount of memory already used,

$$P_{no} = \left( 1 - \frac{M_u}{M} \right)^n, \quad (\text{Eqn 3.28})$$

where  $M_u$  is the amount of memory already used and  $n$  is the number of input dimensions. By implementing a hash tag that is generated separately from a different pseudorandom table, the probability of a collision occurring can be significantly reduced,

$$P_{no} = \left(1 - \frac{M_u}{kM}\right)^n, \quad (\text{Eqn 3.29})$$

where  $k$  is the dynamic range of the hash tag. The system finds the initial virtual address as done in the first method. It then compares the hash tag for a match; if there is no match, the next sequential hash tag is compared until it is either blank or there is a match. If the system cycles through the entire memory allocation without finding a match or an open location, the memory is considered saturated. The system simply takes the next location in this case and thus, causes a collision. There are a couple of things to note about the collision-free approach. If a system is very sensitive to changes in the deterministic control loop, this method can be problematic. The constantly varying number of accesses to memory in order to match the hash tag makes the system nondeterministic. Furthermore, if the system uses a large amount of the memory over time and continually requires the addition of new weights, the memory system will eventually saturate and each weight access will search the entire memory system for an open location before it causes the collision. This would severely limit the throughput of the system. However, it must also be noted that the memory systems have continually grown in an exponential manner and the memory concerns of ten years ago are far less prevalent today. Over time memory allocation problems should only effect a smaller and smaller number of implementations.

### 3.5 Stability Issues

There has been some amount of work done in proving the theoretical stability of CMAC[79, 81, 87, 5]. A common thread in all of this research is that the proofs are based on the rate of change of the weights. There have been two common approaches to bounding the stability of the traditional CMAC: the Lypanov approach and eigenvalue decomposition.

The eigenvalue approach is not as straight forward in bounding the CMAC stability regions, although it does provide the same final solution. It is, however, more useful in examining the trajectory and rate of convergence of the weights for a variety of different learning algorithms. The Lyapunov methods have been used in a variety of different approaches by different authors, but never extended beyond the conventional CMAC implementation[79, 5]. Again, to minimize the repetition of large amounts of information in this dissertation, the original method and the some extensions are presented in chapter 6.

# CHAPTER 4

## SPECTRAL PROPERTIES OF CMAC

### 4.1 Introduction

The bandwidth, or sensitivity, of CMAC, in the Fourier sense, can be determined by analyzing CMAC's ability to learn a series of sine waves. The concept of training CMAC to learn sine waves was previously used in understanding the effects of using the higher-order linear and spline based receptive field shapes[65, 38]. This methodology is quite useful in a single dimension. However, the frequency response of the network is also sensitive to quantization effects and the decimation that takes place in weight-space mapping. This chapter systematically examines the spectral sensitivity of CMAC due to different receptive field shapes, quantization effects and higher dimension decimation effects. The three major receptive field shapes (Albus, linear and spline) and the lattice structures proposed by Albus and An are examined in this chapter[20, 64, 65, 66, 67]. This analysis of the spectral properties of CMAC was initially used as a method to clarify simulations results from a hardware based CMAC model. The hardware development project is described in appendix A.

## 4.2 One-Dimensional Spectral Model

As discussed in the previous chapter, the single input CMAC is a special case such that there exists a weight for every input. This direct mapping is less complex than the multidimensional weight mapping that is decimation dependent on a fixed lattice structure. Therefore, frequency response is first developed in one dimension and correlated to actual CMAC data. From this base model, the multidimensional problems are derived with the additional necessary detail, particularly around the decimation effects.

### 4.2.1 Albus Rectangular Receptive Field Model

The logical place to begin research on the learning convergence of the CMAC techniques is the original Albus model in one dimension. There has already been significant research in this area, particularly around the bandwidth and stability[78, 38]. The bandwidth of one-dimensional CMAC is re-examined here, as a natural procession to understanding the multidimensional and higher-order receptive fields problems. The current functionality and limitations of CMAC are explicitly demonstrated using Fourier analysis. The same Fourier approach is used in the later chapters to demonstrate the expanded functionality. Therefore, this chapter serves as foundation for the entire dissertation.

It is well known that the receptive field shape is a major contributor to learning capabilities of CMAC and the speed at which it converges. The receptive field is fundamentally a lowpass filter in all current CMAC implementations. The output formulation of the CMAC is the sum of the locally activated weight-space,

$$y_s = \frac{\mathbf{c}_s \mathbf{w}_s}{C} \quad (\text{Eqn 4.1})$$

where  $C$  is a scalar generalization parameter,  $\mathbf{c}_s$  is the vector of the weight activation function, and  $\mathbf{w}_s$  is the weight vector. Both vectors are dependent on the sample time,  $s$ .



This function can be rewritten into the following form,

$$y[x] = \sum_{k=0}^{C-1} \mathbf{c}[k] \mathbf{w}_s[x - k] \quad (\text{Eqn 4.2})$$

where  $x$  is the input or state value. The weight activation vector is zero by definition outside the generalization region,  $[0 \dots C - 1]$ , so the output formulation can be written as

$$y[x] = \sum_{k=0}^{C-1} \mathbf{c}[k] \mathbf{w}_s[x - k] = \sum_{k=-\infty}^{\infty} \mathbf{c}[k] \mathbf{w}_s[x - k]. \quad (\text{Eqn 4.3})$$

It can be seen from the above output formulation that CMAC is simply performing the discrete-time convolution sum, which is expressed in the traditional notation below:

$$y[x] = \mathbf{c}[x] * \mathbf{w}_s[x]. \quad (\text{Eqn 4.4})$$

If the weight structure is held static, i.e. no weight update is applied, and the network is excited sequentially across the entire input domain, the output receptive field function is essentially filtering the weight-space. It is important to remember the common identities of the Discrete Fourier Transform (DFT) that convolution in the time domain is equivalent to multiplication in the frequency domain,

$$w[x] * c[x] \stackrel{DFT}{\leftrightarrow} W[\omega] C[\omega]. \quad (\text{Eqn 4.5})$$

A weight-space with unity magnitude for all frequencies or

$$W_\delta[\omega] = 1 \quad \forall \quad \omega \quad (\text{Eqn 4.6})$$

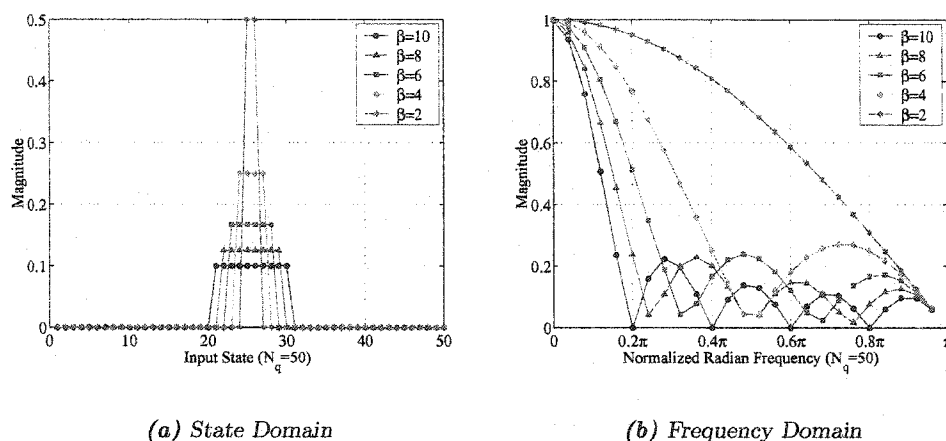
is represented in the state-space as

$$w_\delta[x] = \delta[x], \quad (\text{Eqn 4.7})$$

where  $\delta$  is the impulse or delta function occurring at  $x$  equal to zero. Therefore, by placing a delta function in the weight-space, the impulse response of the network is found.

$$w_\delta[x] * c[x] \stackrel{DFT}{\leftrightarrow} C[\omega]. \quad (\text{Eqn 4.8})$$

By taking the Fast Fourier Transform (FFT) of this impulse response, the bandwidth of the network is examined and compared to actual training data. This essentially is the inverse of the experiment by An [65], where CMAC attempted to learn the impulse response. Using this simple technique, the spectrum of the following receptive fields is generated for a one dimensional CMAC.



**Figure 4-1: Albus Receptive Field (1D)**

The traditional Albus receptive field, or to which it is also referred as the rectangular receptive field, is displayed in figure 4-1(a). The generalization is represented as  $\beta$  in the legend of the figure. The generalization value of  $\beta$  and  $C$  are in fact identical. The sum of the overlapping receptive fields must be one and thus the scaling of magnitude, by the generalization width, can be seen in the plot. The magnitude of corresponding FFT of the output stage of the traditional one-dimensional CMAC is shown for the respective generalization widths in figure 4-1(b). The infinite spectrum and narrow passband are due to the abrupt change in the edge of the receptive field and is clearly identifiable. These components led to the development of many other receptive field models. The different bands of the filter are clearly visible and the effective bandwidth, using standard filter

nomenclature of -3dB, is approximately  $0.1\pi$  and  $0.25\pi$  for generalizations of 4 and 10, respectively. Previous literature has referred to a critical frequency at which the network convergence becomes slow. Since this is a rather qualitative value depending on the definition of "slow", it is unclear where this value would be. Therefore, the author will refer to the effective bandwidth as defined by the -3dB point.

A set of simulations with full CMAC implementations were devised to show that the frequency response of the output stage is the dominant effect in the full CMAC implementation. The CMAC is targeted to learn a series of sine waves. In fact, it is the same series of sine waves used as the basis set for the FFT. This allows the direct comparison of the nulls, bandwidth and other significant spectral components in the CMAC simulation to the FFT analysis of the receptive field shape.

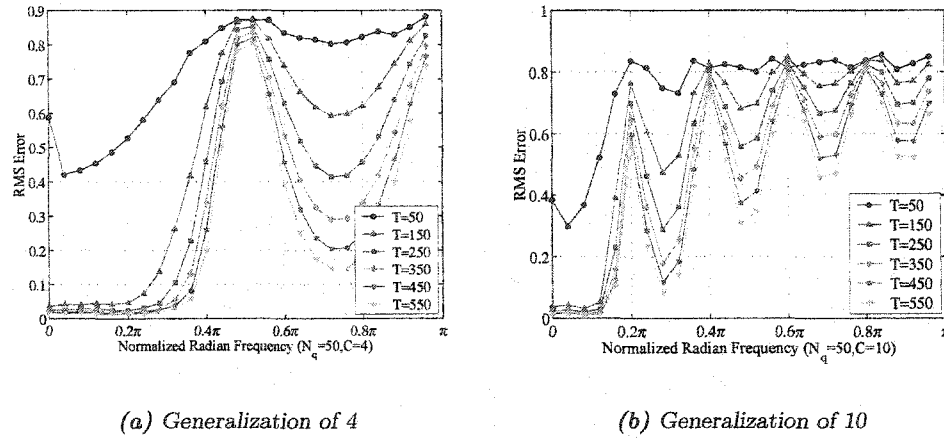
The exact equation for the target function is given as

$$y = \sin\left(\frac{2\pi F}{50}x\right) \quad (\text{Eqn 4.9})$$

where  $F$  is the integer value of the harmonic. The normalized frequency used in the plots is defined as  $2\pi F/50$ . The value of  $F$  is swept from 0 to 24. The input state is  $x$  and it contains 50 possible levels. With 50 input levels for the state-space and setting  $F$  to 24, the number of input levels per cycle of the target function is approximately 2. This meets the Nyquist sampling criteria of 2 samples per cycle. The input states and target responses are presented in a random order. By sweeping the frequency of the target function from 0 to the Nyquist limit and holding the number of input states to the same values used during the FFT analysis of the receptive field, this simulation is effectively performing a Fourier analysis on the complete CMAC model. The process of training for a series of sine waves was also used by An [65].

The CMAC implementations throughout this research use the traditional integer implementations for all calculations. The experimental simulations scale all target functions by a factor of 1024 and then truncate to the nearest signed integer value. The Root Mean

Squared (RMS) error is used to determine how closely the target functions are being approximated. The scaling factor of 1024 is removed from the RMS error value before displaying the results. The RMS output is displayed on a scale of 0 to 1 for the previous 50 samples that cover the full input space.



**Figure 4-2:** CMAC Convergence for Different Harmonics (1D, Albus Receptive Field)

Actual CMAC simulations were performed for the one-dimensional Albus CMAC and are shown in figures 4-2(a) and 4-2(b) for generalizations of 4 and 10 respectively. The  $T$  in the legend represents the number of training cycles, and the learning rate,  $\alpha$ , which in all of the preceding graphs, was set to 0.5.

Figure 4-2(b) shows the progression of the CMAC from 50 to 550 training samples for each harmonic. The rate of convergence for each harmonic can be directly inferred from the graph by comparing the reduction in RMS error as the number of samples progresses. Figure 4-2(b), the actual simulation of CMAC with a generalization of 10, can be directly compared to the FFT spectrum results in figure 4-1(b) for the same generalization width, which is represented as  $\beta = 10$  in the legend.

In figure 4-2(b), it is quite evident that the rate of convergence at low frequencies is

significantly higher than the convergence rate at the opposite end of the spectrum. This is proportional to the spectrum of the receptive field, shown in figure 4-1(b). For example, if the frequency value of  $0.1\pi$  and  $0.9\pi$  are chosen for examination between figures 4-2(b) and 4-1(b), it is clear that the receptive field spectrum at  $0.1\pi$  has a large magnitude. This translates into a small RMS error in the actual CMAC simulation. The simulation also approaches this small RMS value rapidly and has almost completely converged by 150 training samples.

The exact opposite set of properties is seen at the frequency value of  $0.9\pi$ . The receptive field shows significantly less magnitude at this frequency. This translates into a high RMS error and very slow convergence. At 150 training samples, the frequency value of  $0.9\pi$  has barely reduced and the RMS error at the  $0.1\pi$  is almost completely converged after 105 training cycles. Therefore, it can be stated that the rate of convergence for any particular frequency is proportional to the frequency response of the output stage as defined by the receptive field shape.

Since there is a unique weight for every input, the system can slowly converge to any of the harmonics except the nulls where multiples of the wave length are exactly equal to generalization width. Rectangular filters have relatively high-magnitude side lobes, which can actually converge on the function, albeit very slowly, if it is not limited by the bit-precision of the discrete system.

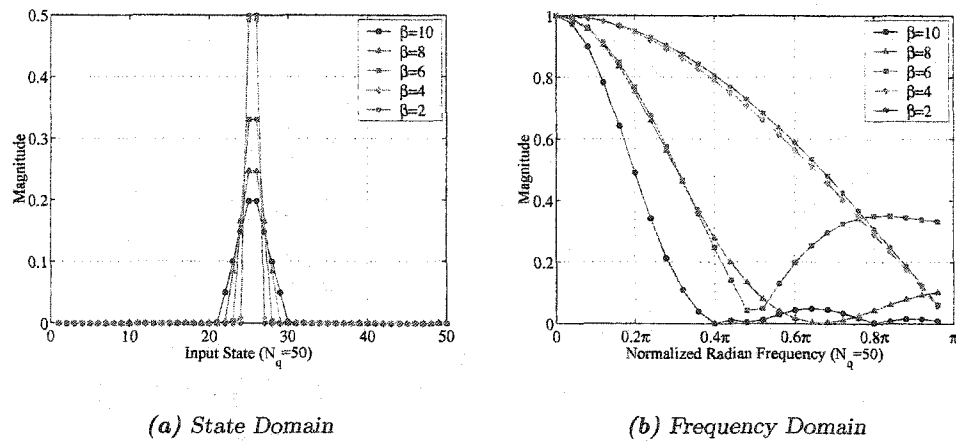
#### 4.2.2 Linear Receptive Field Model

The rectangular receptive field has some clearly undesirable effects. It has a very narrow main lobe and large out-of-band energy. There have been a large number of filters derived in digital signal processing to counteract these limitations. Analogously, there have been a variety of different receptive field models to perform the same function. The first, and one

of the most logical, is the linear receptive field defined by the following equation,

$$c_l[x] = (C/2 - x) \quad (\text{Eqn 4.10})$$

where  $x$  is the distance of input to the weight and this distance can be calculated in a variety of manners, as discussed in the previous chapter. The receptive field value is typically scaled by a larger integer value to support integer based CMAC implementations. The representation of the linear receptive field and the Fourier response of the network are shown in figures 4-3(a) and 4-3(b). The effective bandwidth of the filter is now  $0.17\pi$  and  $0.5\pi$  for generalizations of 10 and 4, respectively. It can also be seen that the non-zero side lobes are suppressed and more spread out than the rectangular receptive field models, particularly in the case of the generalization of 10.

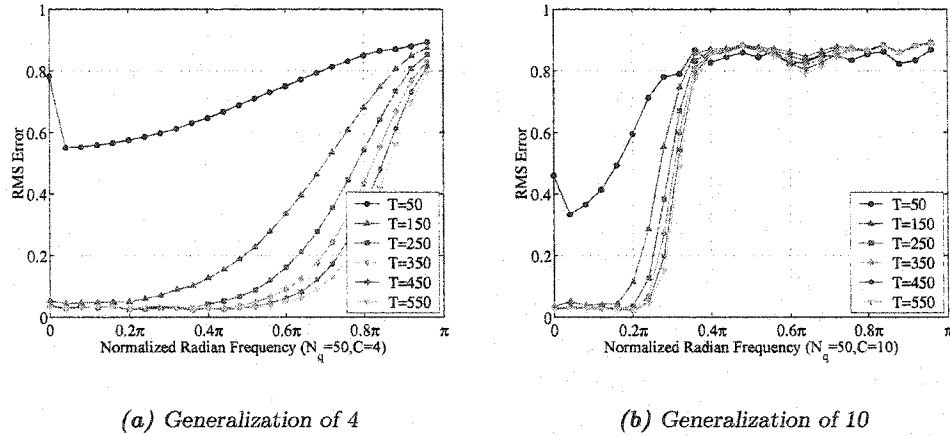


**Figure 4-3:** Linear Receptive Field (1D)

The actual simulations of CMAC with the linear receptive field once again confirms the initial frequency domain model; see figures 4-4(a) and 4-4(b). Some of the side bands actually show no ability to converge on the solutions. This can be seen in figure 4-4(b) in the frequency range above  $0.4\pi$ . It is important to point out that the frequency domain solutions

were computed from floating point math, while the CMAC uses integer-based operations. The result is that sidebands, that have very small values, are actually truncated to zero, due to the lack of precision in the integer math.

It can also be seen that the spectrum for generalizations of 2 and 4 have very similar results in figure 4-3(b). This is due to the linear receptive field model implementation in the UNH CMAC code which always assigns a 1 to the outer edges of the receptive field then indexes a lookup table which is filled from 1 to 128. For a linear receptive field model with a generalization of 2, the UNH CMAC calculates the set of receptive field coefficients as  $\{1, 1\}$ . The set of coefficients for a generalization of 4 are  $\{1, 33, 33, 1\}$ . Both of these functions spread the majority of the response across only 2 states and, essentially, have the same frequency response when used in the CMAC.



**Figure 4-4:** CMAC Convergence for Different Harmonics (1D, Linear Receptive Field)

A comparison of simulation results with the generalization of 10, in figures 4-4(b) and 4-3(b), shows the correlation of the receptive field bandwidth to the actual CMAC bandwidth. Both plots show that the primary passband, or the sensitivity region of the network, is between 0 and  $0.4\pi$ . The magnitude of the receptive field frequency response is greatest as

it approaches the 0 frequency point. If the simulation data is examined at the 50 training set line, this rate of convergence to 0 RMS error follows the exact same relationship, with the exception of the 0 frequency value. This is due to the fact that the RMS value for a DC level is 1, while the RMS value of the sine waves is actually 0.707. Clearly, the dominant component to this point with respect to the bandwidth of the network is the receptive field shape.

### 4.2.3 Spline Receptive Field Model

The spline models for receptive fields were developed to further increase the bandwidth and suppress the side bands. The higher generalization splines have derivatives that can be used to develop hierarchical CMAC models, using techniques similar to the backpropagation methods used in multi-layer perceptrons. The equation for the spline basis function is

$$c_s[\hat{x}] = \frac{(3(C/2)\hat{x}^2 - 2\hat{x}^3)}{(C/2)^2} \quad (\text{Eqn 4.11})$$

where, again,  $C$  is the generalization width and

$$\hat{x} = \frac{C}{2} - x \quad (\text{Eqn 4.12})$$

where  $x$  is the distance of input to the weight.

The receptive field models and the corresponding frequency domain response are given in figures 4-5(a) and 4-5(b), respectively. The spline-based receptive fields for generalizations of 2 and 4 are actually identical to the linear receptive fields of the same generalization. The higher generalizations have a slightly broader frequency response and more suppression of the side lobes. Once again, the corresponding bandwidth of the CMAC used in the simulations has the same bandwidth as the Fourier analysis done for the receptive field shape at any give generalization width; see figures 4-5(a) and 4-5(b).



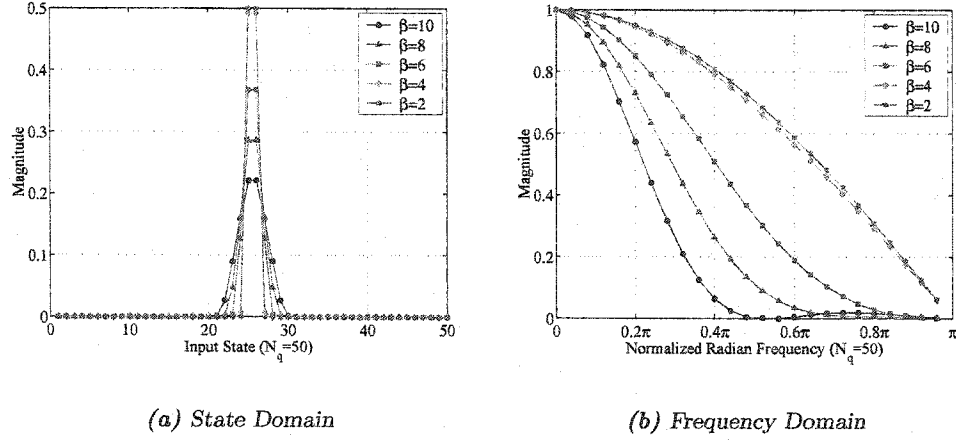


Figure 4-5: Spline Receptive Field (1D)

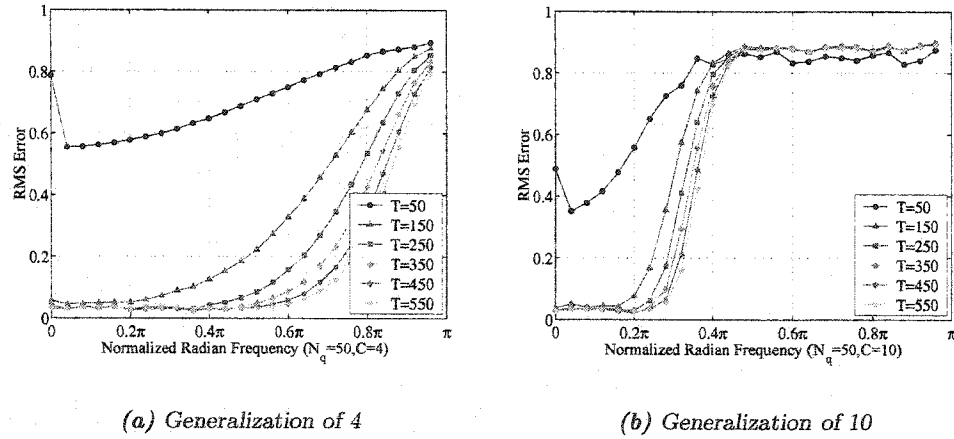


Figure 4-6: CMAC Convergence for Different Harmonics (1D, Spline Receptive Field)

#### 4.2.4 Quantization Effects in One-Dimension

Another feature of the traditional CMAC is the quantization levels that can be used to limit the number of input states; refer to section 3.3.1 for more detail. The quantization spreads the generalization over  $N$  multiple distance of the input space, where  $N$  is the quantization value. For example, if a one-dimensional CMAC with a generalization of 10 and an input quantization value of 2 is used, the CMAC algorithm maps each two consecutive input states to one weight location and the network still generalizing over 10 weights. Since each weight now represents two input-states, this CMAC is generalizing over 20 input-states with 10 weights. Notice, the number of weights is reduced, or decimated, over the input-states by a factor of 2.

This process of weight reduction is somewhat analogous to decimation in signal processing terms. For systems with a bandlimited spectrum, the decimation process is an effective means of optimizing information storage and minimizing computation. However, if the system's spectrum is not bandlimited, such as in the case of a CMAC with rectangular receptive fields, the network now suffers from aliasing problems, or a loss of information.

Using a simple set of formulas from digital signal processing, the effects of weight decimation, during input quantization, can be analyzed. In a digitally sampled system, replication of the Fourier spectrum occurs at periods of  $2\pi$ . In the case of a tradition digital filtering, the sampled function

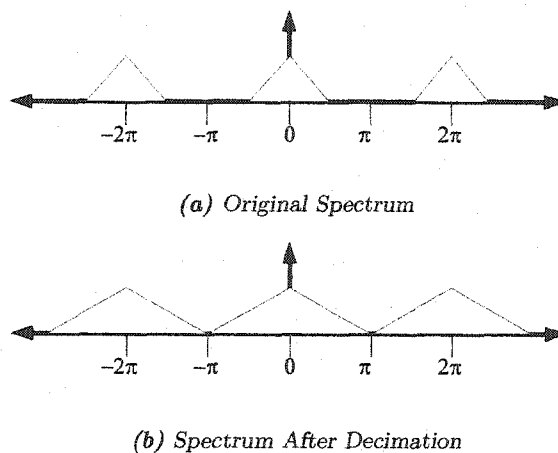
$$s[n] = x[MTn] \quad (\text{Eqn 4.13})$$

is decimated by the integer factor  $M$  and the sampling period is defined as  $T$ . The Fourier spectrum of the sampled function is

$$S(\omega) = \sum_{k=0}^{M-1} X\left(\frac{\omega - 2\pi k}{M}\right) \quad (\text{Eqn 4.14})$$

where  $X$  is the original Fourier spectrum of  $x$  before the decimation. Figure 4-7(a) represents the spectrum of a sampled bandlimited signal. The base spectrum, the region center around 0, is contained within the frequencies  $-0.5\pi$  and  $+0.5\pi$ . The replica spectrum, which exist

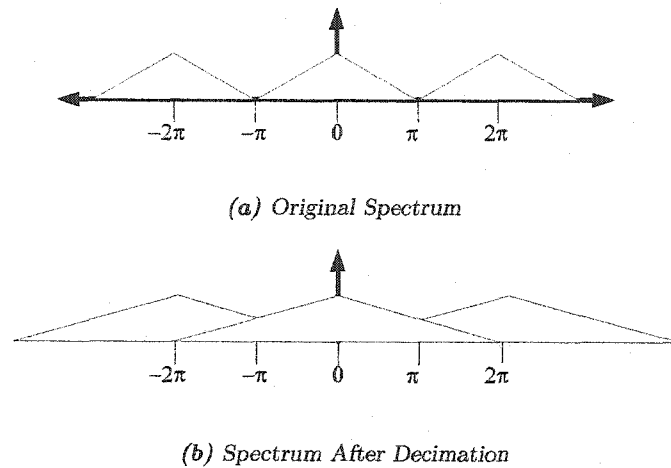
in all sampled systems, are at every interval of  $2\pi$ . The spectrum shown in figure 4-7(a)



**Figure 4-7:** Decimation without Aliasing

only occupies half of the total available spectrum. The regions from  $0.5\pi$  to  $1.5\pi$  and  $-0.5\pi$  to  $-1.5\pi$  contain no spectral energy. Therefore, this system can be decimated by a factor of 2 without loss of information. Figure 4-7(b) represents the spectrum after decimating by a factor of 2. The entire available spectrum is now occupied, but there are no overlapping regions of support between the base spectrum and the replica spectrum. Therefore, all of the information in this signal can be recovered, by interpolating between the samples and filtering.

Figure 4-8(a) represent a signal that is considered critically sampled. A critically sampled signal is a bandlimited signal that has been sampled at the lowest possible frequency, such that no spectral information is lost during sampling. By decimating a sampled signal, the effective sampling frequency is divided down and in the case of a critically sampled signal, it will result in a loss of information. Figure 4-8(b) shows the spectrum of figure 4-8(a) after decimation by a factor of 2. This results in the overlapping regions of support between the base spectrum and the replica spectra. The overlapping of spectral information



**Figure 4-8:** Decimation Causing Aliasing

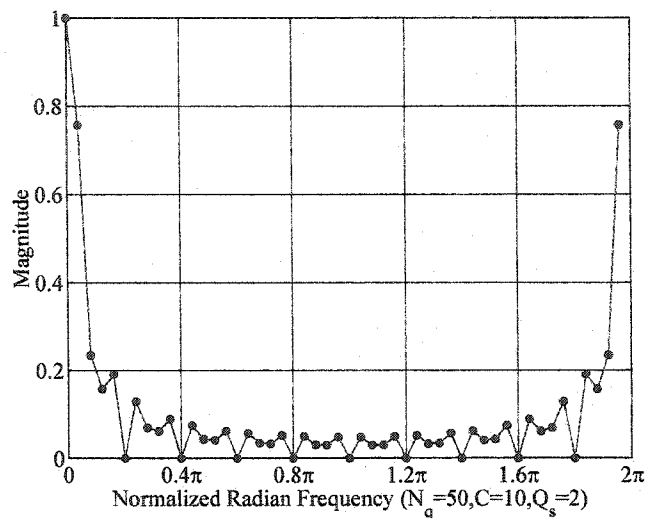
is the aliasing effect. Since figure 4-8(b) has some amount of overlapping across the entire base spectrum,  $-\pi$  to  $\pi$ , some amount of distortion and loss of information will occur at all frequencies. When the amount of aliasing energy equals the amount of base spectral energy at the same frequency, there will be a total loss of information.

It is clearly evident from the previous CMAC simulations that the receptive field bandwidth limits the frequency response of the network and as previously mentioned, the input quantization is effectively a decimation process. Therefore, CMAC using input quantization can be investigated for aliasing problems, by examining the bandwidth of the network in conjunction with the amount of weight decimation.

To reiterate, the quantization in CMAC also spreads the generalization by a factor of  $M$  by reducing the number of weights, therefore, the base spectrum is also changed. The equation above still applies but the base spectrum is found by convolving the receptive field over a weight-space that has been reduced by a factor of  $M$ . This narrows the bandwidth of the filter by generalizing over a greater effective width.

As previously shown, the rectangular receptive field has a very broad spectrum with a narrow main band. The spectrum for a one-dimensional CMAC with a generalization width

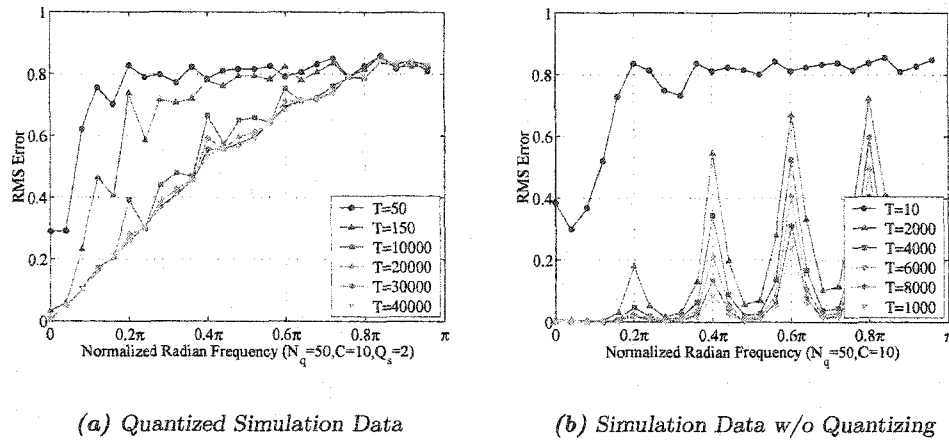
of 10 with an input quantization value equal to 2 is shown in figure 4-9.



*Figure 4-9: Spectrum of Albus Receptive Field with Quantization*

It should be noted that there is significant spectral energy at all frequencies. If the spectrum were divided in half around the frequency of  $\pi$  and the two halves were expanded as done in figures 4-8(a) and 4-8(b), it is clear that severe aliasing will occur. The magnitude of the aliasing spectrum to the base spectrum increases as the frequency approaches  $\pi$ , where the aliasing and base spectrum have equal magnitude. The amount of aliasing energy at any frequency correlates well to the achievable learning accuracy, figure 4-10(a). At the frequency of 0, no aliasing exists and the network can fully converge. The base spectrum and aliasing spectrum have equal magnitude at  $\pi$  when the weights are decimated. Therefore, the network cannot converge at this frequency,  $\pi$ . This result is as also seen in figure 4-10(a).

The CMAC with quantization can only achieve a low RMS error for the extremely low frequencies and does not converge at all for the higher frequencies. The CMAC without quantization can converge, albeit slowly, at all frequencies as seen in figure 4-10(b).



**Figure 4-10:** The Effect of Input Quantization (1D, Albus Receptive Field)

It is important to recognize that these simulations were run for 4000 cycles with only 25 possible discrete input states and there is clearly no further convergence that is going to happen. Therefore, it is quite evident that a broad spectrum of the rectangular receptive field is quite detrimental when the input state is quantized.

The spectrum linear receptive field, which has a broader main lobe, or effective bandwidth, also has a well-contained spectrum. From figure 4-11(a), the spectrum of the linear receptive field has minimal energy between  $0.4\pi$  and  $1.6\pi$ . Therefore, the quantization of the input is going to still limit its bandwidth, since the generalization is expanded to cover more input-states during quantization. However, the containment of the spectrum and its minimal sidebands limits the aliasing. This allows the network to converge without error due to aliasing over the bandlimit which is defined by the receptive field shape. Figure 4-11(b) shows no ill-effects from aliasing. The rectangular receptive field has a variable amount of error at all frequencies above zero.

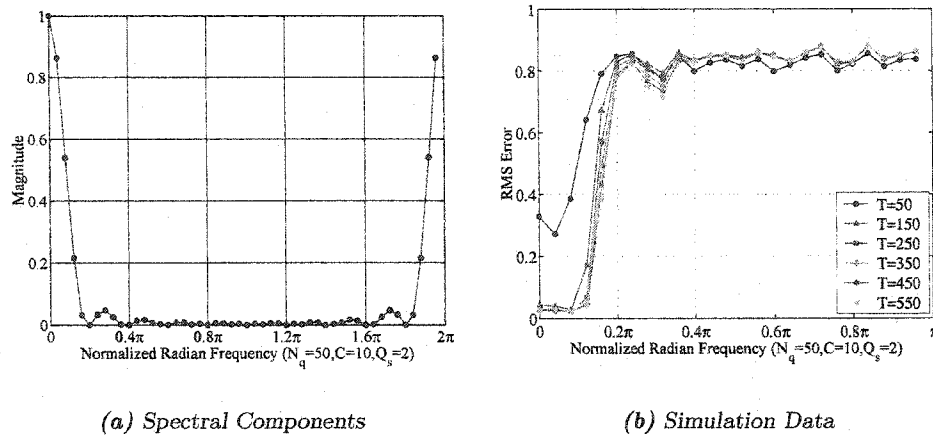


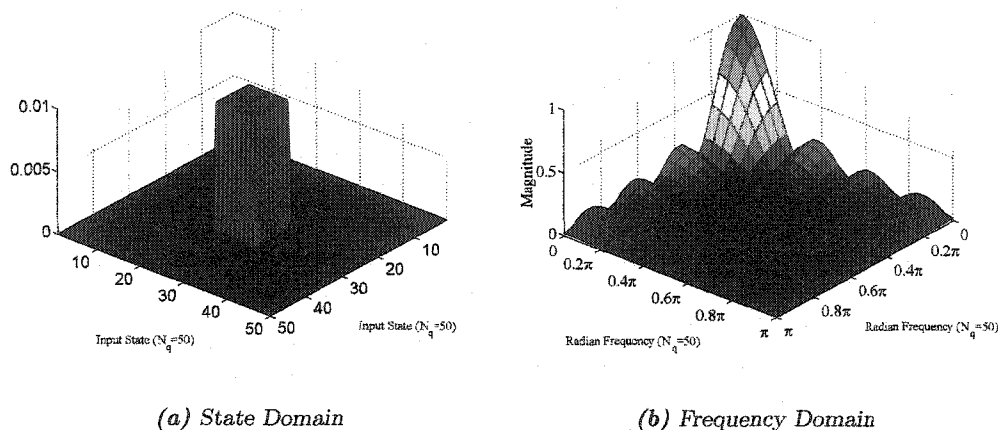
Figure 4-11: The Effect of Input Quantization (1D, Linear Receptive Field)

### 4.3 Higher-Dimension Problems

As mentioned in previous sections, the multidimensional CMAC has the added complication of decimation from the state-space to the weight-space. This decimation has the advantage of reducing the weight storage required for the network, which is especially valuable in very high-dimensional problems, and decreasing computation by only computing a subset of the weights. Contrary to the benefits, the reduction in storage has an adverse effect on the bandwidth of the network. The decimation in the weight-space has some similarities to the quantization effects, in particular aliasing, but the generalization is not expanded over a larger region of the input space. These effects are systematically demonstrated in the following sections. The results in the following section are limited to two-dimensions for visual purposes only. The concepts extend to even higher dimensions.

#### 4.3.1 Receptive Field Shape Effects

This section begins, in the same manner as the previous section on one dimensional CMAC models, by examining the basic frequency response expected for a given receptive field shape and generalization parameter. The process to determine the spectrum of the receptive field



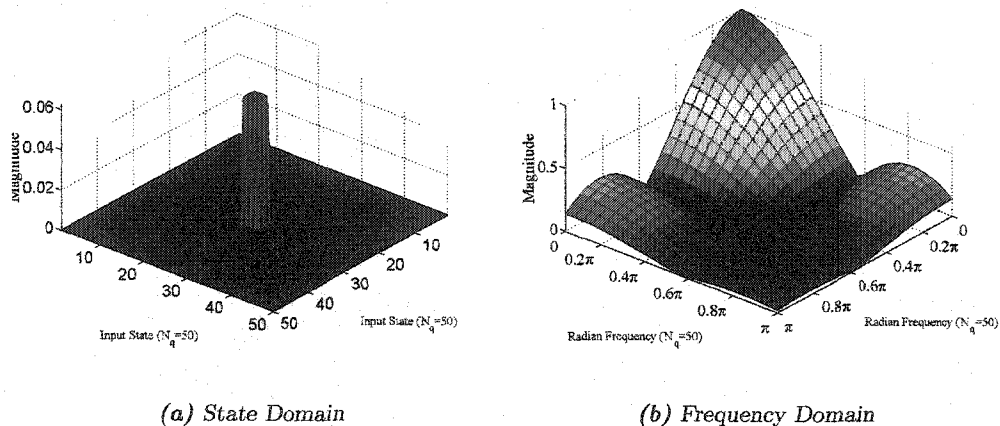
**Figure 4-12: Albus Receptive Field (2D, C=10)**

is identical to previous sections except the convolution sum and FFT are extend to two dimensions. Figures 4-12(a) and 4-12(b) represent the basic shape of a two-dimensional Albus CMAC and the frequency response of the associated receptive field, neglecting the decimation effects. The frequency domain plot represents only one quadrant,  $[0 \dots \pi, 0 \dots \pi]$ , in the full frequency range,  $[-\pi \dots \pi, -\pi \dots \pi]$ . It can clearly be seen that the generalization of 10 plot has a limited main lobe with side lobes rippling out along the axis. The continuous spectrum causes problems during decimation, as will be shown in a later section.

To limit the number of cases to study, the generalizations were limited to 4 and 10 for this analysis. The Albus CMAC of generalization 4 is shown in figures 4-13(a) and 4-13(b). As expected, the smaller generalization has a broader main passband, but major sidebands that are not suppressed.

The Linear receptive field for the generalization of 10 is shown in figures 4-14(a) and 4-14(b). Although the linear receptive field seems somewhat jagged, it was modeled from the UNH CMAC implementation that divided up a table of 128 entries, ranging from 1 to 128, into half as many entries as the generalization number. The value of one is always included as the outer edge of the receptive field, since it is an integer based system. The





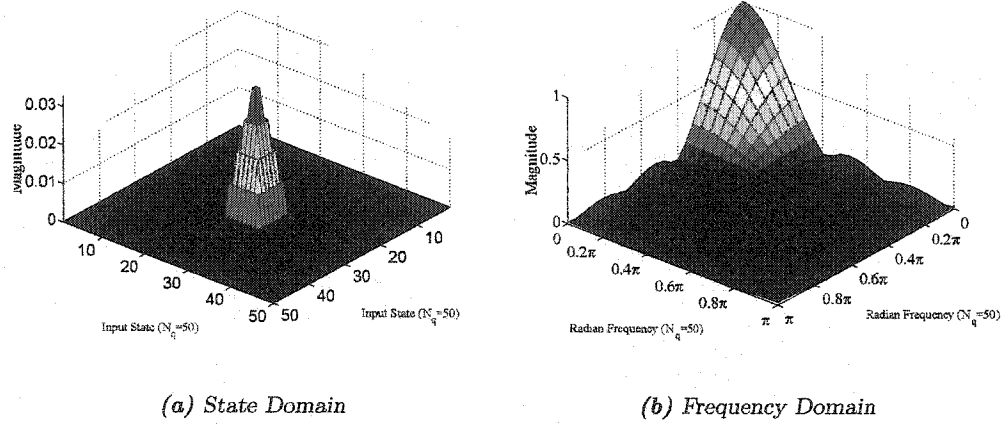
**Figure 4-13:** Albus Receptive Field (2D,  $C=4$ )

figure 4-15(a) shows the receptive field model for the CMAC with a linear receptive field and a generalization of 4. The set of numbers included in this field are  $\{1, 66, 66, 1\}$ . One might have expected the values to be  $\{1, 2, 2, 1\}$ , which would have a different spectral representation. It can be seen that the linear models have a larger bandwidth and lower sub-bands, as one would expect from the one-dimensional linear receptive field.

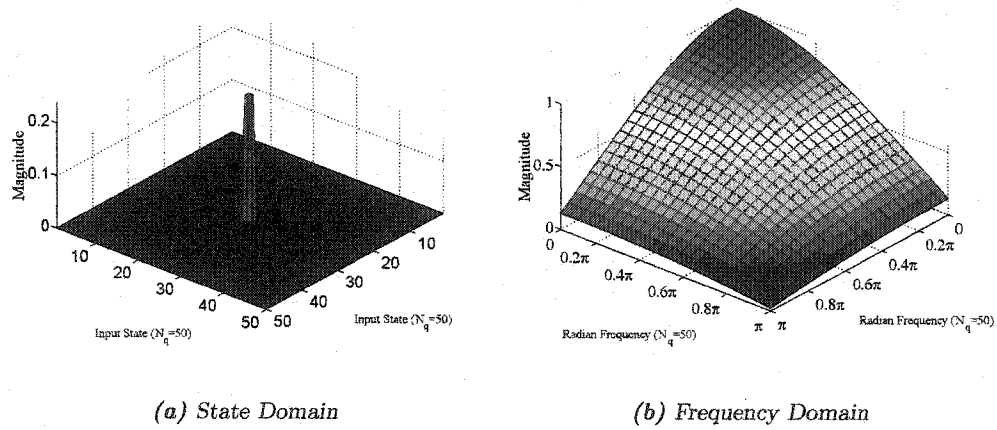
The actual simulation data with which to compare is shown in the final section of this chapter. As will be seen in the following sections, the decimation effects dominate the response of the network, and comparison with the simulation results are presented after that discussion.

### 4.3.2 Decimation Effects of the Lattice Structure

The concept of decimation in time is commonly used in signal processing to reduce the amount of data to the bare essential amount to still reconstruct the output. This is possible as long as there is no aliasing in the spectrum after decimation. CMAC uses a multi-dimensional lattice structure to decimate the number of weight locations from the total number of inputs. The amount of decimation is controlled by the generalization parameter.

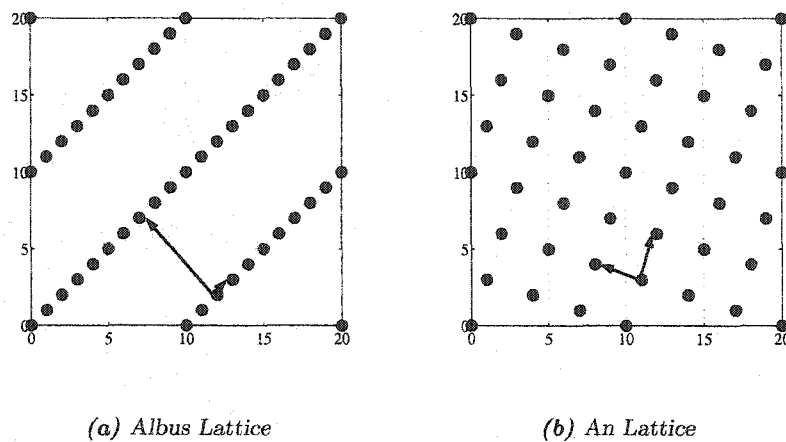


**Figure 4-14:** Linear Receptive Field (2D, C=10)



**Figure 4-15:** Linear Receptive Field (2D, C=4)

As seen in the previous two sections, the generalization parameter, along with the receptive field shape, also defines the frequency response of the network. It is important that the decimation does not exceed values that cause severe aliasing, or the effective bandwidth of the network will be critically diminished. Since CMAC was not initially designed based on these principles, an aliasing problem in the network does exist. In this section of the research, the bandwidth of a particular CMAC implementation with respect to the output stage is first constructed and then the effects of decimation are applied to the frequency response of model. This facilitates the investigation of possible aliasing problems. The process used in the following sections is derived from research on processing multidimensional sampled signals [97][98].



**Figure 4-16:** Lattice Structure (Receptive Field Placement)

It is necessary to formulate a decimation analysis based on the different CMAC lattices. The Albus and An lattices were defined in a previous chapter as a vector of displacement, to coincide with the UNH CMAC implementation. The two different lattices can be seen in figures 4-16(a) and 4-16(b). These lattices are now described by sampling matrices that can be applied to a base spectrum to examine for aliasing.

The columns of the sampling matrix are defined by the linearly independent vectors that form the sampling lattice. For example, two vectors are drawn on figure 4-16(a). Using these vectors, the sampling lattice can be constructed. The longer vector has a length of -5 on the horizontal axis and 5 on the vertical axis. These two numbers represent the first column of the sampling matrix shown below. The second vector has a magnitude of 1 in each direction and this represents the second column of the matrix. Using this process, the lattice structure formed by the traditional Albus CMAC in two dimensions can be represented by the sampling matrix  $M_{albus}$

$$M_{albus} = \begin{bmatrix} -5 & 1 \\ 5 & 1 \end{bmatrix}. \quad (\text{Eqn 4.15})$$

Using the vectors superimposed on figure 4-16(b) as a guide, the lattice structure formed by the An CMAC can be represented by the matrix  $M_{an}$ .

$$M_{an} = \begin{bmatrix} -3 & 1 \\ 1 & 3 \end{bmatrix} \quad (\text{Eqn 4.16})$$

The sampling lattices can be check for linear independence between the columns by finding the inner product between the columns of the matrix. Both of the above matrices have linear independent, or orthogonal, column vectors. The same process can be used to find the sampling matrix for higher dimensional CMAC lattices, but the process gets increasingly difficult as the number of dimensions increases.

The determinant of the sampling matrix is the decimation factor and it is can be seen that both lattices produce the same decimation factor of 10.

$$|\det M_{albus}| = |\det M_{an}| = 10 \quad (\text{Eqn 4.17})$$

The decimation grows as a power of the number of inputs and the determinant of sampling matrix can be checked with know decimation factor for CMAC. The decimation factor for CMAC  $D_f$ , in terms of generalization and input states, is given as

$$D_f = C^{n-1} \quad (\text{Eqn 4.18})$$

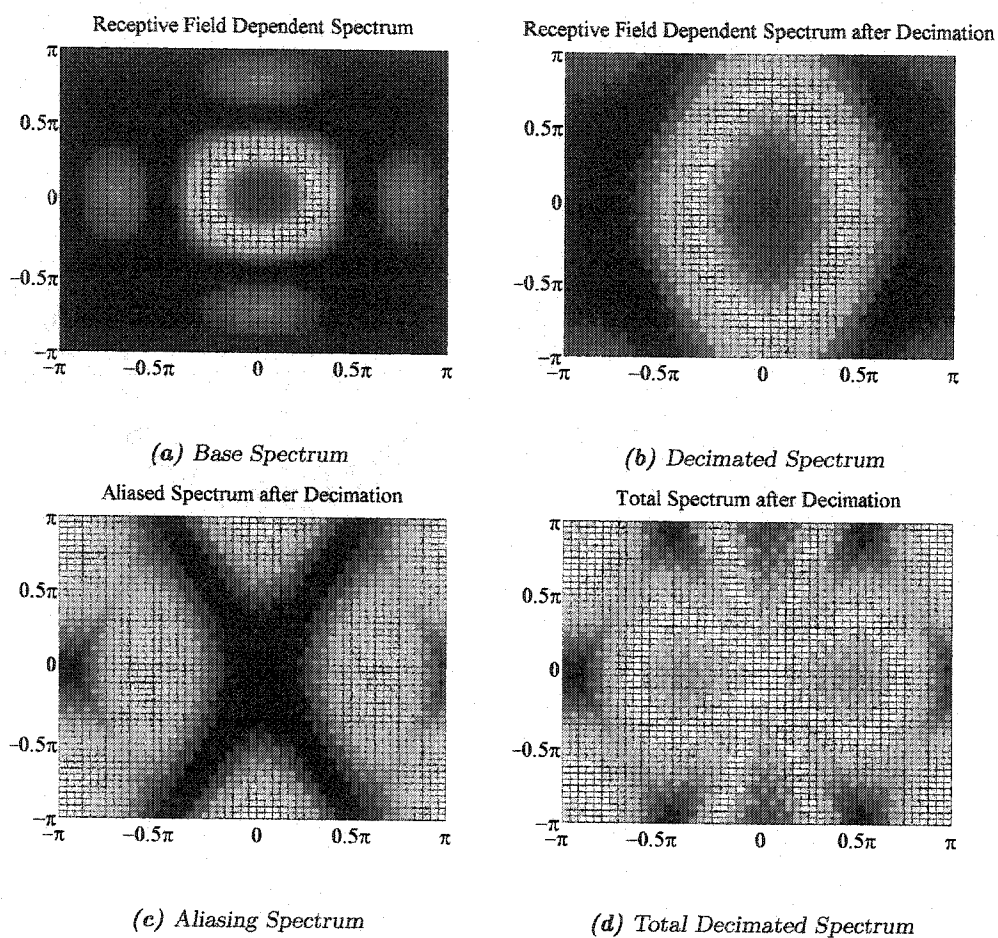
where  $C$  is the generalization and  $n$  is the number of input states. The sampling lattices can now be used to compute the spectrum considering the decimation effects. The equation is similar to the equation used for the quantization effects in one dimension, except that the generalization of the CMAC network is not expanded and the base spectrum is only effected by the original receptive field shape. The new spectrum is given as

$$Y(\omega) = \frac{1}{\det(M)} \sum_{k \in N(M^T)} X(M^{-T}(\omega - 2\pi k)) \quad (\text{Eqn 4.19})$$

where  $N(M^T)$  is the set of all integer vectors of the form  $M^T x$ ,  $x \in [0, 1)^n$  and  $n$  is the number of dimensions. The total number of vectors  $k$  is equal to the determinant of  $M$ ,  $|\det M|$ . The set of vectors  $k$  is sometimes referred to as the *polyphase shift vectors*[99]. When  $k$  is a null vector, the effects of decimation on the base spectrum can be analyzed. All other  $k$  vectors produce the aliasing and replica spectrum. The original spectrum  $X$ , that is used in the analysis, is determined by the receptive field shape.

From this set of equations, the different components of the spectrum are derived. This is used to analyze different receptive field, generalization and lattice structure combinations with respect to their effective bandwidth. The initial spectrum without decimation is presented for reference. This is followed by the decimated spectrum without any aliasing components. The aliasing components are then displayed alone and, finally, the total spectrum is constructed. The regions of overlapping support represent the amount of effective bandwidth lost to the aliasing components from the primary spectral components.

Figure 4-17 shows spectra that represent the traditional CMAC receptive field and lattice structure. The blue regions are low magnitude, while the red regions are the highest magnitude. All plots are shown with the same relative magnitude scale, so the amount of aliasing to the magnitude of the primary spectrum can be inferred. However, these spectral plots do not included all possible aliasing components, but rather a subset of the aliasing components to highlight overlapping regions of support. From figure 4-17(a), the traditional CMAC has one main lowpass band with a single sideband along each axis. After



**Figure 4-17:** The Effect of weight-space Decimation (2D,  $C=4$ , Albus Receptive Field and An Lattice)

the decimation, figure 4-17(b), the main band is elongated particularly along the vertical axis. This is due to the asymmetric sampling matrix,  $M_{albus}$ . Although, the lattice is not shown, the sampling matrix for an Albus lattice structure with a generalization width of 4 is

$$M_{albus} = \begin{bmatrix} -2 & 1 \\ 2 & 1 \end{bmatrix}. \quad (\text{Eqn 4.20})$$

A diagonal sampling matrix, also known as a rectangular sampling lattice, would have produced a symmetric growth in the spectral energy, like

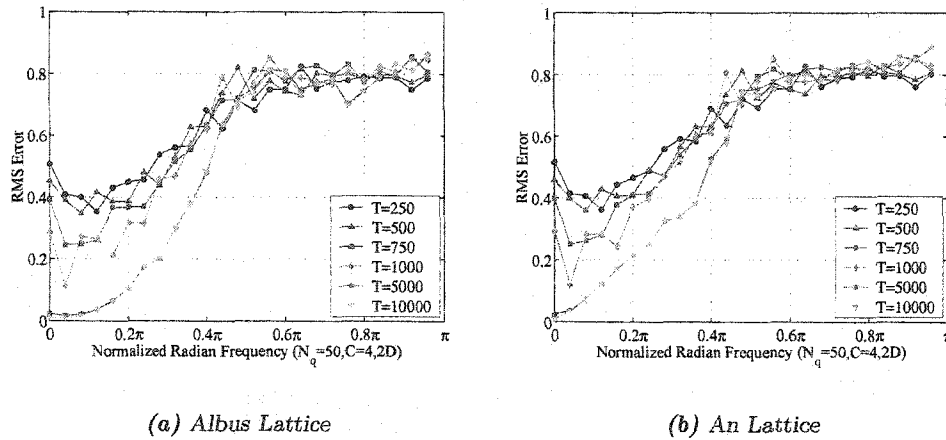
$$M_{rect} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}. \quad (\text{Eqn 4.21})$$

The spectrum is also truncated along this axis, meaning it will overlap with the replica spectra above and below it. Since these are discrete sampled systems, there are replica spectra at every interval of  $2\pi$ . The aliasing effects from these replica spectra are seen in figure 4-17(c). It is clearly evident that a large portion of the primary spectral region is occupied by aliasing components. The extremely low frequency component and the diagonal regions are the only areas without aliasing. Finally, the entire spectrum is shown in figure 4-17(d). The output of the network interpolates the decimated weight states by using the receptive field function. This process attempts to recover figure 4-17(a) from the spectrum of figure 4-17(d). The problem is any aliasing that has occurred cannot be corrected for and the energy in those regions is diminished by the proportion of aliasing.

This data is correlated with a two-dimensional CMAC of the same parameters attempting to learn the function

$$z = \sin \left( \frac{2\pi F}{50} (x + y) \right) \quad (\text{Eqn 4.22})$$

where  $x$  and  $y$  are the input dimensions. Each dimension has 50 possible input values. This function is used throughout the remainder of this chapter. Figures 4-18(a) and 4-18(b) show the actual CMAC simulation for an Albus receptive field with a generalization of 4 and the two different lattice implementations—the Albus and An lattices. The lattice, used

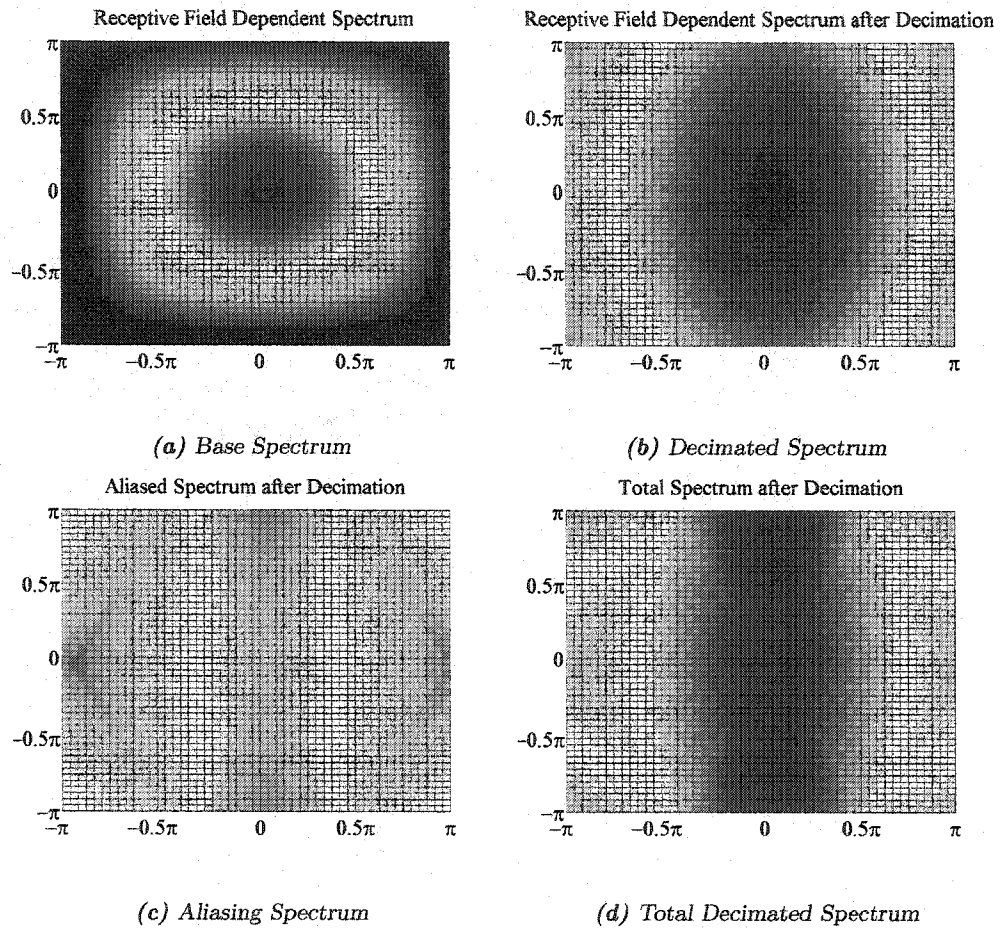


**Figure 4-18:** Simulation Data for Albus Receptive Field Shape (2D,  $C=4$ )

in the network which produced the simulation results in figure 4-18(b), is not strict to An's definitions and could more appropriately be labeled the off-diagonal lattice. With a generalization of 4, there is a relatively small amount of decimation and the lattice change does not improve much since it essentially just changes orientation of the aliasing. From figure 4-17, it is evident that there is aliasing at all but the lowest frequencies and this correlates with the simulation data. It can also be seen that since the unaliased spectrum after decimation is effectively the region from 0 to  $|0.5\pi|$ , then the decimation truncates the spectrum. There is a sharp fall-off at this point in the simulation data also. In the one-dimensional case, it was seen that the spectral region outside the primary passband continued to converge, albeit at a very slow pace. In the multi-dimensional case, the decimation reduces the number of possible unique solutions, resulting in the upper-half band not converging at all.

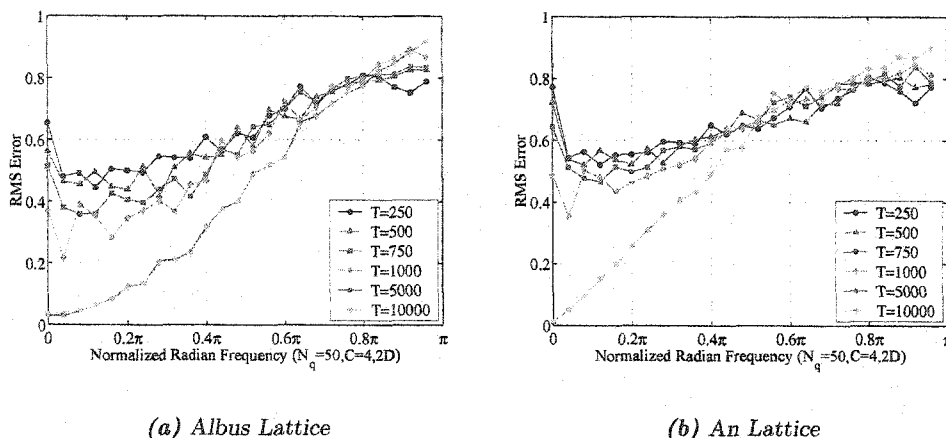
The spectrum of the linear receptive field in two dimensions with the standard Albus lattice structure is shown in figure 4-19. From the previous one-dimensional case, it is expected that the overall bandwidth would be much greater, and this is clearly seen in the figure 4-19(a). However, the bandwidth is so great that plots of the decimation show the spectrum spreading beyond the Nyquist region of  $-\pi$  and  $\pi$ . This results in the severe





**Figure 4-19:** The Effect of weight-space Decimation (2D,  $C=4$ , Linear Receptive Field and An Lattice)

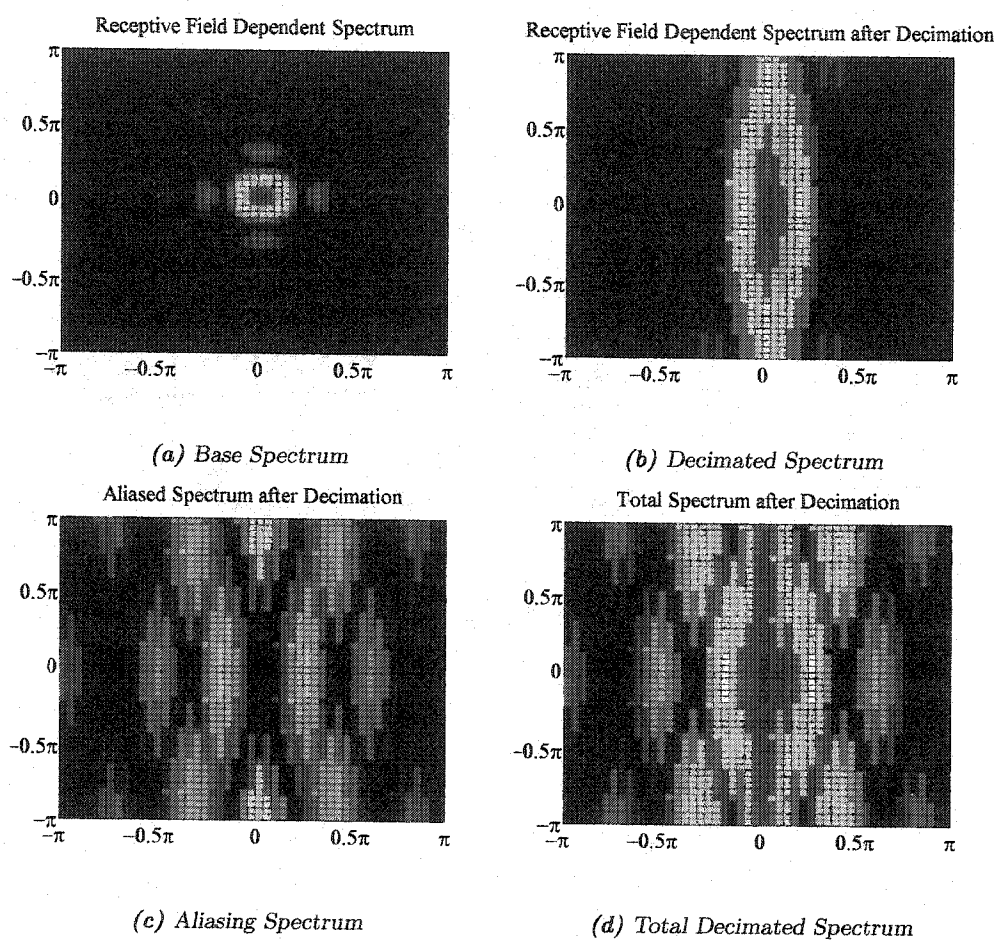
aliasing in figures 4-19(c) and 4-19(d). In this case, the advantage of the wider bandwidth receptive field turns out to be detrimental, since it is also responsible for the severe aliasing.



**Figure 4-20:** Simulation Data for Linear Receptive Field Shape (2D, C=4)

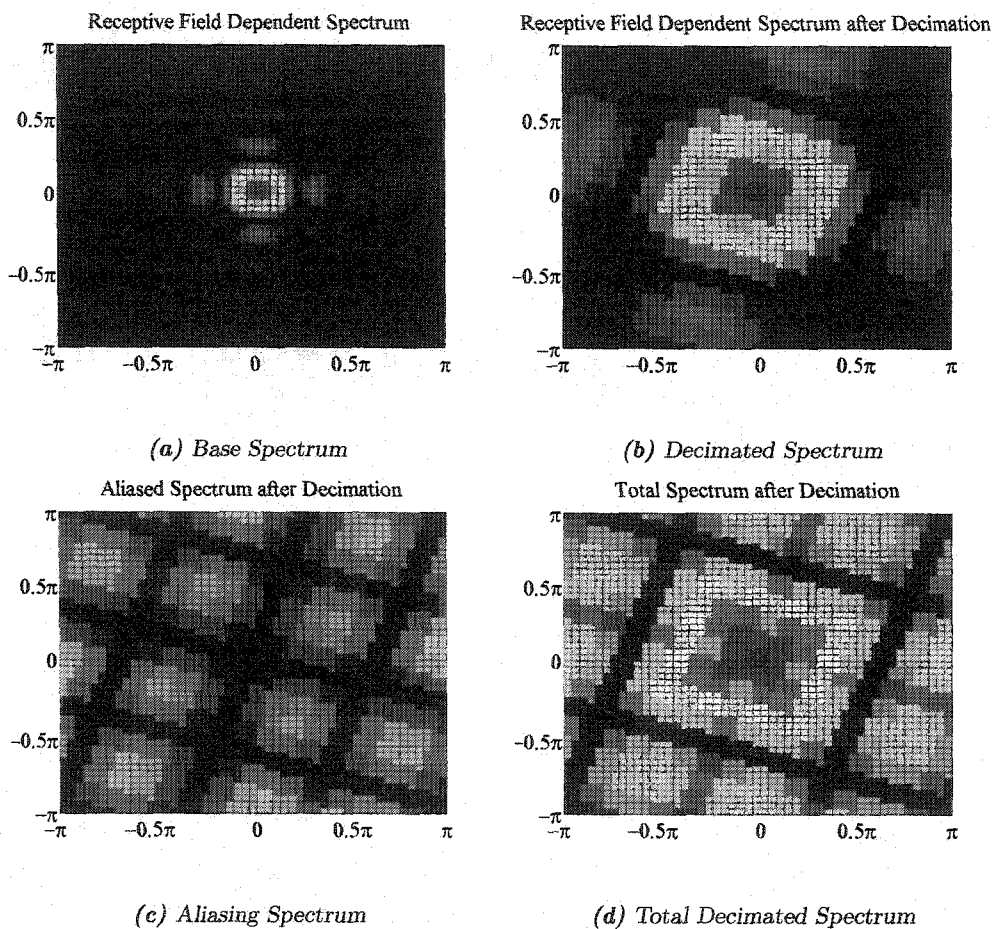
The CMAC simulations corresponding to the generalization of 4 and linear receptive field are shown in figures 4-20(a) and 4-20(b). Since both the traditional Albus lattice and An lattice, coupled with this receptive field, showed severe aliasing, there is little or no improvement between the lattice structures. In fact, both simulations show no region of convergence beyond the DC or frequency equal to zero point. The linear receptive field in this case is actually so broad that the decimation causes severe aliasing, consequently hampering the network from converging at any frequency.

As the generalization increases, the effective bandwidth decreases and decimation increases. The Albus CMAC of generalization of 10 with two input states is shown in the series of plots in figure 4-21. In this example, the uneven decimation effect is dramatically shown in figure 4-21(b). The elongated spectrum produces aliasing effects in the same direction, while little or no interference is shown in the other directions. The alias spectrum shows minimal interference at the origin and therefore, it can be expected that this network would



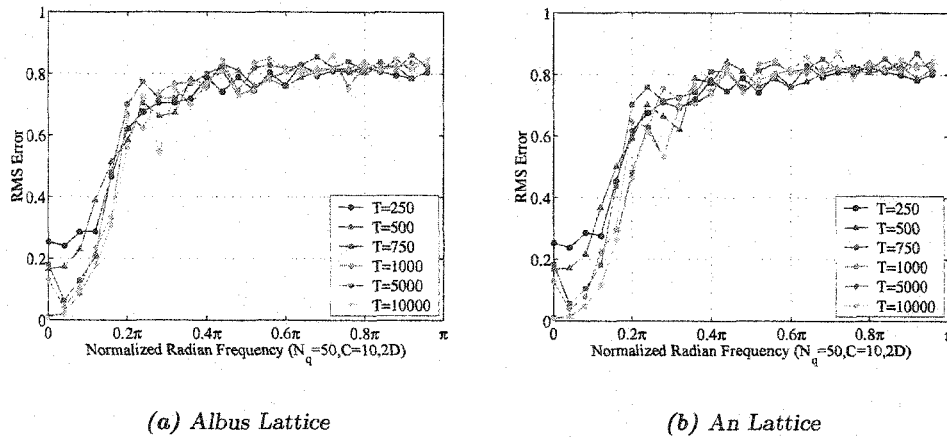
**Figure 4-21:** The Effect of weight-space Decimation (2D,  $C=10$ , Albus Receptive Field and Albus Lattice)

converge to an accurate solution over this small window of low-frequency functions. The rectangular receptive field, however, has an extremely wide spectrum which, in turn causes some amount of aliasing as can be seen in figure 4-21(c) at almost all other frequencies. Clearly, the magnitude of the aliasing is not as great as the aliasing in the cases of generalization of 4. Consequently, the aliasing is not the dominate effect in this network's ability to converge, but rather the minimal bandwidth of the original receptive field function.



**Figure 4-22:** The Effect of weight-space Decimation (2D,  $C=10$ , Albus Receptive field and An Lattice)

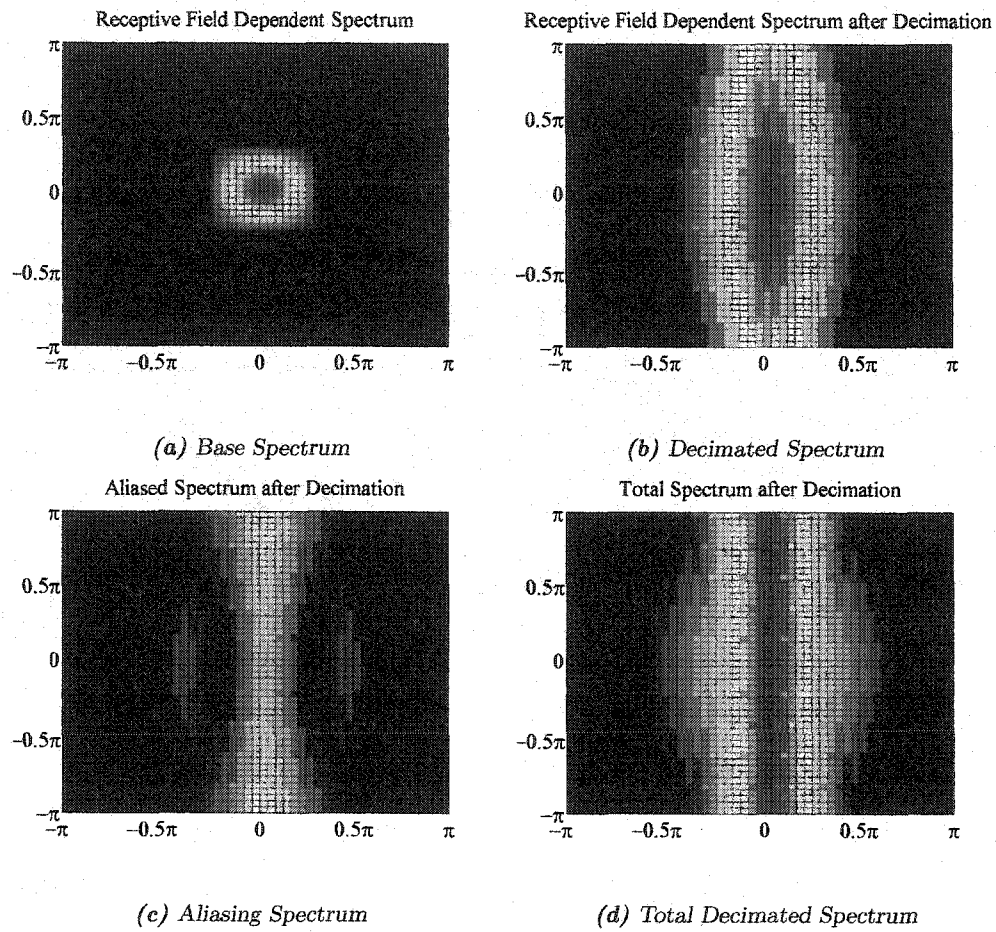
By using the An placement strategy, the spectrum is evenly decimated as shown by the series of plots in figure 4-22. However, the spectrum of the original receptive field shape continues on indefinitely and produces some amount of aliasing over almost the entire frequency range. Once again, the aliasing is a much smaller magnitude than the main passband magnitude. Although there is a slight reduction in the aliasing by more symmetrically decimating the weight-space, it does not produce a great improvement in the overall convergence of the network, since the width of the main lobe of the receptive field is a more dominant effect.



**Figure 4-23:** Simulation Data for Albus Receptive Field Shape (2D, C=10)

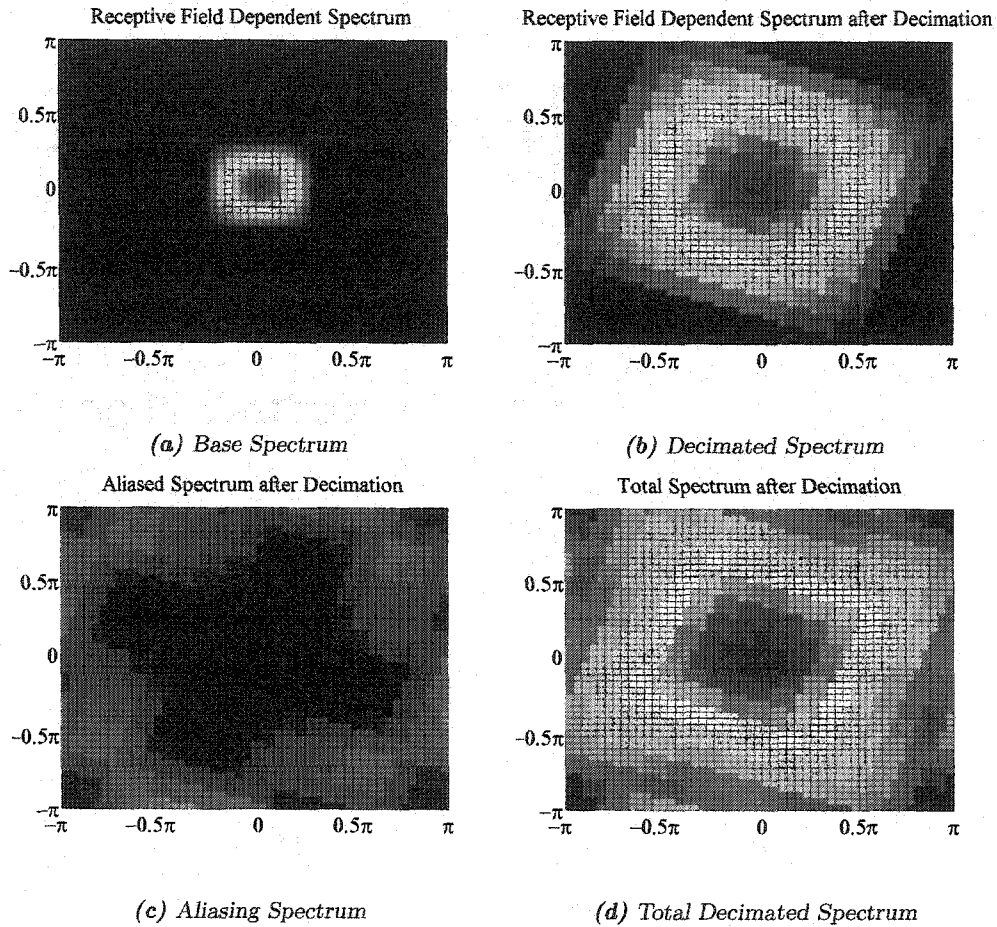
As expected, there is little difference in the simulation data for the two-dimensional CMAC with the Albus receptive field across the two lattice structure implementations, figures 4-23(a) and 4-23(b). The An lattice structure effectively widens the main lobe of the frequency response, but only by a small amount.

Figure 4-24 is the linear receptive field implementation with the Albus lattice structure for the same two-dimensional network with a generalization of 10. The linear receptive field has a broader main pass band, but the uneven sampling of the Albus lattice elongates and



**Figure 4-24:** The Effect of weight-space Decimation (2D,  $C=10$ , Linear Receptive Field and Albus Lattice)

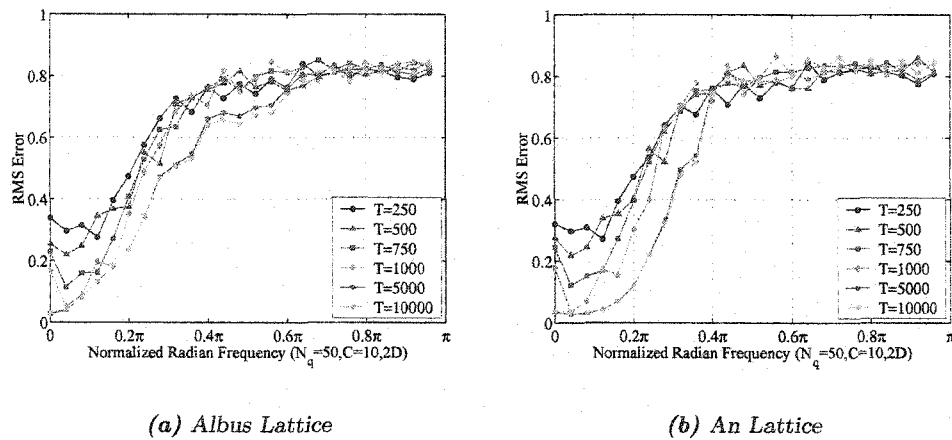
truncates this pass band. Although the sidebands are small, the aliasing of the elongated main passband causes aliasing all the way to the origin. Since this lattice structure does not support bandwidth of the receptive field, it is expected that the network will have poor convergence properties.



**Figure 4-25:** The Effect of weight-space Decimation (2D,  $C=10$ , Linear Receptive Field and An Lattice)

The true learning enhancement gained from the An placement is seen in the series of plots in figure 4-25. In this case, the linear receptive field has a broad spectrum, relative

to the Albus receptive field of the same generalization. The even sampling expands the receptive field radially without any significant truncation of the main pass band. This lack of truncation, coupled with the lack of side bands, produces a very low amount of aliasing. Therefore, the balance between symmetric sampling and suppressed side lobes allows the reconstruction of the entire original pass band. This is the best example of CMAC as a perfect reconstruction filter with decimation. In other words, the filter bandwidth and amount of decimation are well aligned.



**Figure 4-26:** Simulation Data for Linear Receptive Field Shape (2D, C=10)

Finally, the actual CMAC demonstrates this effect in figures 4-26(a) and 4-26(b). The linear receptive field with the traditional lattice again causes aliasing from the improper lattice structure and wider bandwidth with receptive field. For this particular combination, all the frequencies have some aliasing and the usable spectrum or region where perfect reconstruction could occur is minimal. The An placement minimizes the aliasing of the main pass band and significantly broadens the region of convergence of the network, as seen in figure 4-26(b).

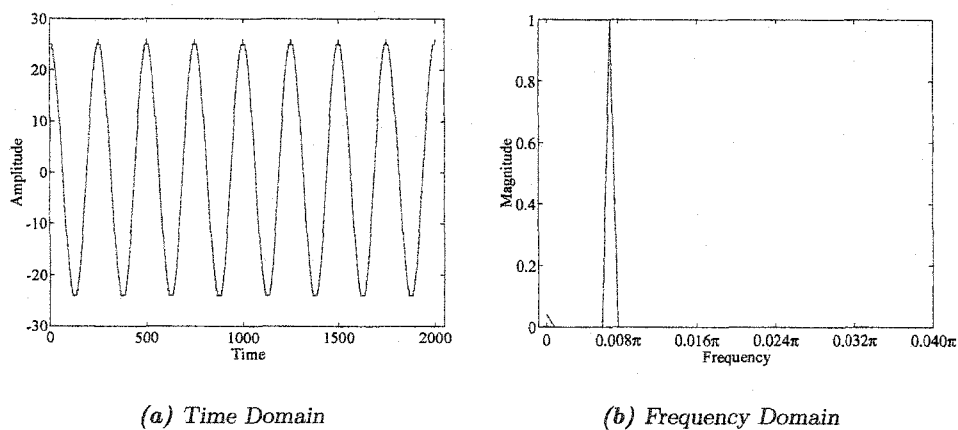


## 4.4 Mapping Functions into CMAC

This chapter has focused on the bandwidth and spectral properties of CMAC. The entire analysis was performed with well understood and defined target functions. However, it is sometimes difficult to understand the spectral properties of the function being mapped into CMAC. To illustrate this problem, an example is adapted from research on active disturbance cancellation using Time-Delay CMAC models[100]. A simple disturbance source is defined as

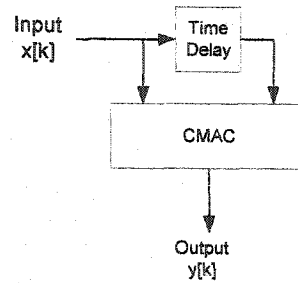
$$x[k] = 25 * \sin\left(\frac{\pi}{125}k\right) \quad (\text{Eqn 4.23})$$

where  $k$  is the discrete time counter. A plot of this simple function versus time is shown in figure 4-27(a) and the expected frequency spectrum is shown in figure 4-27(b). Clearly, this



**Figure 4-27:** Sine Wave

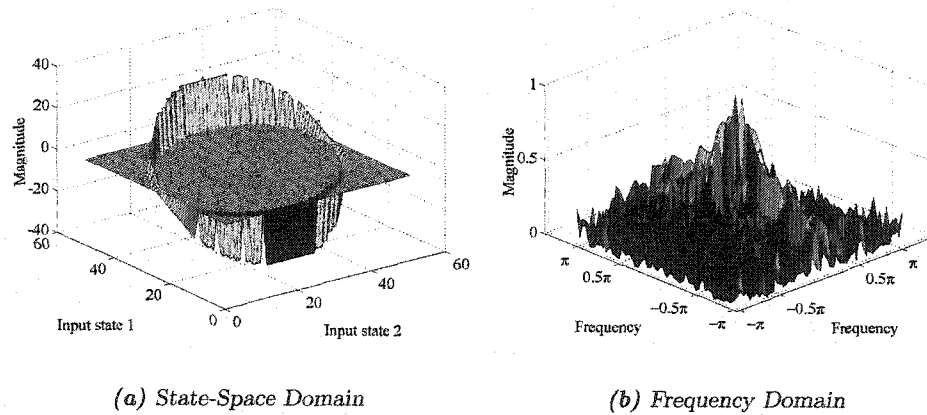
is a simple function, but it needs to be mapped onto the CMAC state-space. The discrete time counter,  $k$ , continues to infinity and therefore, cannot be used as an input dimension. A solution to this problem is using a Time-Delay CMAC model, described in figure 4-28. The disturbance source can be now described as two-dimensional function



**Figure 4-28:** Time-Delay CMAC Model

$$y[k] = 0.5x[k] + 0.5x[k - 200] \quad (\text{Eqn 4.24})$$

where  $x[k]$  is the sampled input signal and  $y[x]$  is the target function. This system maps into a two-dimensional CMAC where in the input-states are  $x[k]$  and  $x[k - 200]$ . This



**Figure 4-29:** Time Delayed Sine Wave

two-dimensional function produces a circular function in the state-space, as shown in figure 4-29(a). The spectrum of the function in figure 4-29(a) is displayed in figure 4-29(b). It is clear that this spectrum no longer represents a single one-dimensional sine wave, but rather

a more complex multidimensional function. Although this dissertation is focused mostly on the functional bandwidth of the CMAC, this example demonstrates the need to understand both the bandwidth of the network and how a function is mapped onto the network.

## 4.5 Discussion

This chapter demonstrated two dominant effects on the ability of CMAC to learn a broad spectrum of functions in the Fourier sense. First, the frequency response generated by a given receptive field shape dominates the rate of convergence, especially in the one-dimensional case where there is a full set of weights that can possibly represent any arbitrary solution. In the multidimensional case, the weight set is decimated and therefore orthogonal patterns will always exist that can never be learned. It was demonstrated that the lattice structure, which controls the decimation, plays an important role in the effective bandwidth of the network. This decimation, or sub-sampling, coupled with original pass band of the receptive field is the dominant effect which controls the network bandwidth. Traditionally, there is a limited amount of flexibility in the bandwidth of the receptive field and the decimation factor, since they are both controlled by the generalization factor. Consequently, there are lattice structure, receptive fields and combinations of the two which severely diminish the network's ability to learn. Finally, it is clearly evident from simulation and analysis that CMAC is fundamentally an adaptive low-pass filter.

# CHAPTER 5

## THE WAVELET BASED CMAC

### 5.1 Introduction

From the previous chapter, it is clearly evident that the CMAC network is an effective, adaptable, multidimensional lowpass filter, which is analogous to the concept of "common inputs give common outputs," a phrase which is typically used to describe associative neural networks. A lower-frequency target function allows for wider generalization, which produces a broad coverage of the state-space. If the target function contains higher frequency components, the generalization of the network has to be reduced to maximize the bandwidth of the network. Consequently, the distribution of information associated with each target pair is also reduced in the state-space, potentially leaving significant untrained areas.

There is no physical limitation that excludes high frequency functions from being widely generalized. In fact, the ability to generalize is limited by the bandwidth and not an absolute frequency. For example, the Fourier Analysis generalizes widely from the lowest frequency to the highest frequency with a series of sine waves that make up its basis set.

The main dilemma in associative networks is the inability to widely generalize higher frequency functions. However, if a narrowband function exists, a network should still be able to generalize widely, as long as its basis function is closely related to the target frequency. In fact, the wavelet transform and other techniques of multiresolution analysis can generalize at a variety of widths across the entire frequency range.

This chapter explores a more flexible network architecture inspired by the wavelet transform and multiresolution analysis. First and foremost, the network uses a wavelet basis function for the receptive field. This receptive field can be modulated to produce a band-pass, highpass, or the traditional lowpass filter of the CMAC network. The next important feature is a disassociation of the decimation factor of the weight-space from the receptive field size. The weight reduction is controlled from a separate level of weight quantization, which results in an even distribution of the weights across the state-space. Furthermore, this quantization method enables the computational load of the network to be varied and potentially optimized.

The resulting network is a more flexible architecture that allows for selective bandwidth learning and reduced computation in many cases. This chapter follows the same process as the previous chapter, beginning with simple one-dimensional problems and completing with complex multidimensional problems involving decimation.

## 5.2 The Wavelet

The basis of wavelet analysis is to derive frequency content within a localized region of a larger surface, thus producing a multiresolutional view, or localized frequency content as a function of input space. In audio processing, it separates the variation in frequency across time, producing resolution simultaneously in both domains, which gave rise to the term multiresolution analysis. This is done by scaling and translating a base wavelet across the input domain, or state-space. This is in essence the same principle as using receptive fields in associative neural networks. The neural network uses the receptive field as the localizing filter. One essential requirement of the wavelet is compact support such that basis function is only nonzero over some finite region less than the input domain size. This finite width in an associative neural network is defined by the generalization.

There are other requirements typically associated with wavelets and the wavelet trans-

forms. One property in particular is orthogonality of the basis functions, which minimizes redundant information and also allows perfect reconstruction of the target function. In the network design to follow, the wavelets are typically over-complete and non-orthogonal. Therefore, some strict definitions of the wavelet analysis are not met. However, the network is based on a compactly supported oscillatory function that is scaled and translated in the state-space. It will also be shown that the network uses the principles of decimation that make the wavelet transform highly efficient. The network is therefore referred to as the Wavelet based CMAC (WCMAC), since it continues to incorporate the general concepts of the original CMAC network, including the associative structure and hashing of the weight-space.

The wavelet chosen for this network is the costrap function,

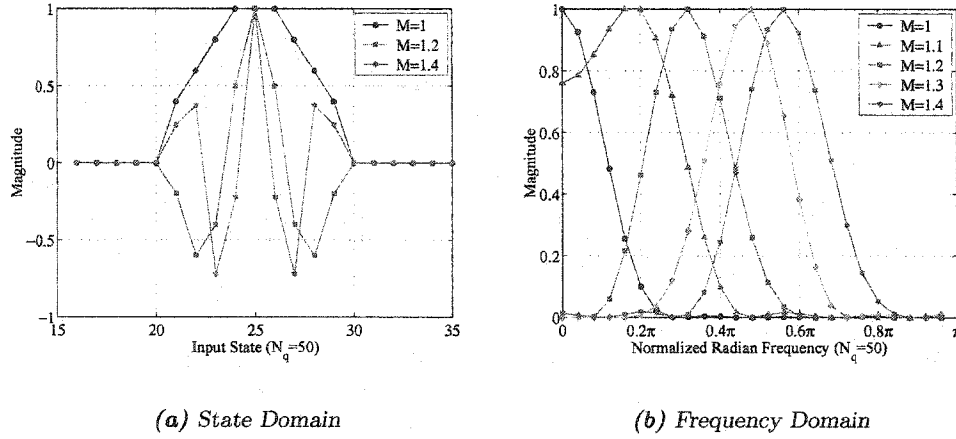
$$\text{costrap}(x, a, b, M) = \cos \left( \frac{2\pi}{M} x \right) \min \left\{ \max \left[ \frac{b - |x|}{b - a}, 0 \right], 1 \right\} \quad (\text{Eqn 5.1})$$

where  $a$  is considered the dilation parameter and  $b$  is the translation parameter. The  $M$  parameter modulates the location of the passband, allowing the construction for a frequency-selectable bandpass filter. Consequently, it is referred to as the modulation parameter in subsequent sections. The dilation parameter can be set to the width of the receptive field or generalization parameter. Since the reference point, or center, of the wavelet will also be the position of the weight, the translation parameter can be set to zero. In essence, the wavelet function is simply a new receptive field shape. The new shape is implemented using a look-up table.

### 5.3 One-Dimensional Wavelet Based CMAC

For simplicity and clarity, the network structure will be derived starting with the one-dimensional case. The initial step is to use the wavelet function as described before as the receptive field. From the previous chapter, it is evident that the frequency response of the receptive field defines the CMAC response. Figure 5-1(a) shows a wavelet of generalization

11 with a variety of modulation values. The corresponding frequency response of the network with this receptive field is shown in 5-1(b). The exact same process, including the number of input states and target functions that was outlined in the previous chapter, is used to examine the bandwidth of the WCMAC model.



**Figure 5-1:** Wavelet Receptive Field in One-Dimension

To accommodate the new wavelet receptive fields, the learning algorithm is adjusted to handle receptive field values with a mean of zero, i.e. no D.C. component. First, the output generation is modified to a more general equation that handles the WCMAC and the previous receptive fields.

$$y_s = \frac{\mathbf{c}_s \mathbf{w}_s}{\|\mathbf{c}_s\|} \quad (\text{Eqn 5.2})$$

where  $y_s$  is the output at sample  $s$ ,  $\mathbf{c}_s$  is the receptive field vector and  $\|\mathbf{c}_s\|$  is the norm of the receptive field vector or  $\sqrt{\mathbf{c}_s \mathbf{c}_s^T}$ . The input-space, in this case a scalar value,  $x_s$ , activates  $C$  weights, where  $C$  is the generalization parameter. The addresses of the weights

are

$$\mathbf{a}_s = \begin{bmatrix} x_s - \delta_r + 1 & x_s - \delta_r + 2 & \dots & x_s - \delta_r + C \end{bmatrix} \quad (\text{Eqn 5.3})$$

$$= \begin{bmatrix} a_{s1} & a_{s2} & \dots & a_{sC} \end{bmatrix} \quad (\text{Eqn 5.4})$$

where  $\delta_r$  is the offset to the edge of the receptive field defined as  $((C+1)/2)$ . The distance vector  $\mathbf{d}_s$  is the integer value of the distance to each weight.

$$\mathbf{d}_s = \begin{bmatrix} -\delta_r + 1 & -\delta_r + 2 & \dots & -\delta_r + C \end{bmatrix} \quad (\text{Eqn 5.5})$$

$$= \begin{bmatrix} d_{s1} & d_{s2} & \dots & d_{sC} \end{bmatrix} \quad (\text{Eqn 5.6})$$

The distance vector in this case is only related to the generalization, which is also the offset index to the address vector,  $\mathbf{a}_s$ , and is therefore only computed once. The receptive field vector is a function of the wavelet versus this distance vector

$$\mathbf{c}_s = \psi(\mathbf{d}_s) = \text{costrap}(\mathbf{d}_s, a, b, M) \quad (\text{Eqn 5.7})$$

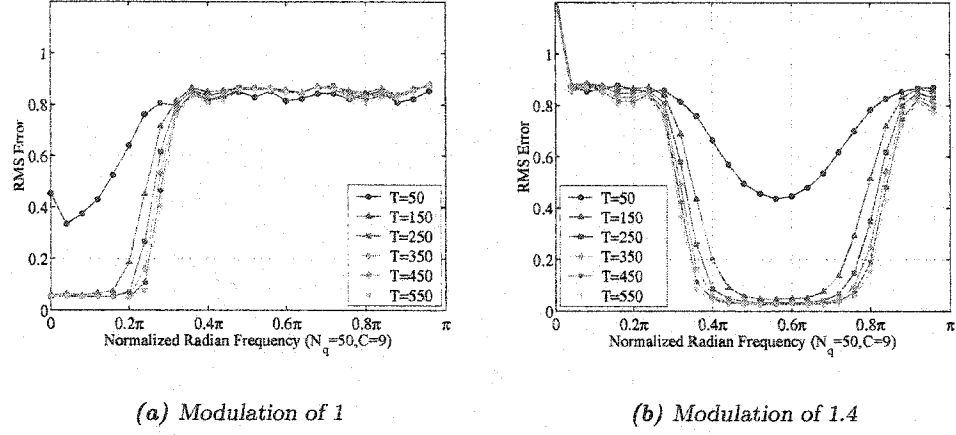
where  $\psi()$  is the base wavelet function with translation, dilation and modulation parameters fixed. The weight update algorithm for the WCMAC is

$$\Delta \mathbf{w}_s = \alpha \frac{\mathbf{c}_s^T}{\|\mathbf{c}_s\|} (\hat{y}_s - y_s) \quad (\text{Eqn 5.8})$$

where  $\hat{y}_s$  is the target output value for an input of  $x_s$ ,  $y_s$  is the WCMAC output and  $\alpha$  is the learning rate.

Stability of this learning algorithm is analyzed in the following chapter, including a discussion on further minimizing the computation time. Using these formulas and the costrap receptive field function from figure 5-1(a), the one-dimensional WCMAC is applied to the Fourier test in the previous chapter. The actual simulation results, figures 5-2(a) and 5-2(b), verify the predicted frequency response in figure 5-1(b). The CMAC limitation as a lowpass filter or network that does not generalize well at higher frequencies has been





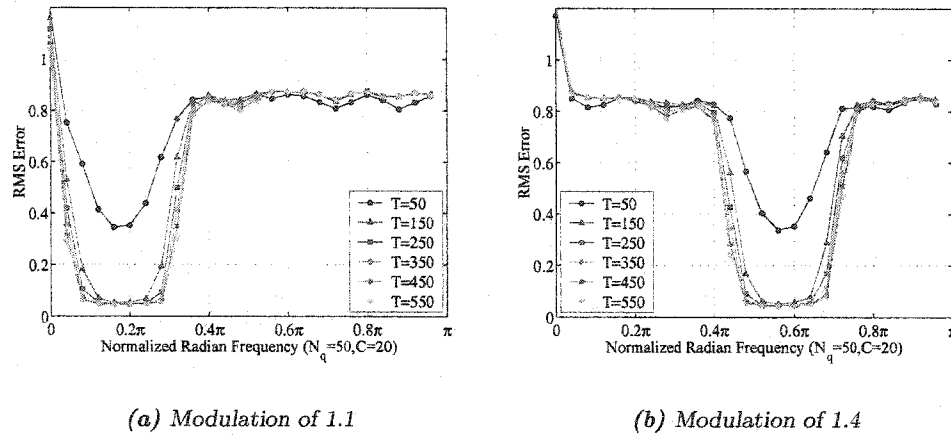
**Figure 5-2:** WCMAC Convergence for Different Harmonics (1D, C=9)

conquered by wavelet based receptive fields and the corresponding learning algorithm to support it.

By extending the generalization, the receptive field shape acts as a higher order filter, making the passband narrower. Unlike previous CMAC implementations, this does not restrict the ability to learn higher-frequency content. By increasing the generalization parameter and the modulation parameter, the network is generalizing high frequency data widely across the input space. Figures 5-3(a) and 5-3(b) demonstrate the narrowing of the passband, but the ability to still learn a wide range of frequencies. In fact, the WCMAC can widely generalize across any frequency range by simply tuning the modulation parameter. The WCMAC can learn all the functions originally targeted by CMAC and entirely new functions in different frequency ranges. Each simulation uses a network with a generalization of 20.

### 5.3.1 Quantization Effects in One-Dimension

As previously noted, the single dimension CMAC has a unique weight for each input state. Therefore, it needs to compute an address and value for each weight, even when the band-



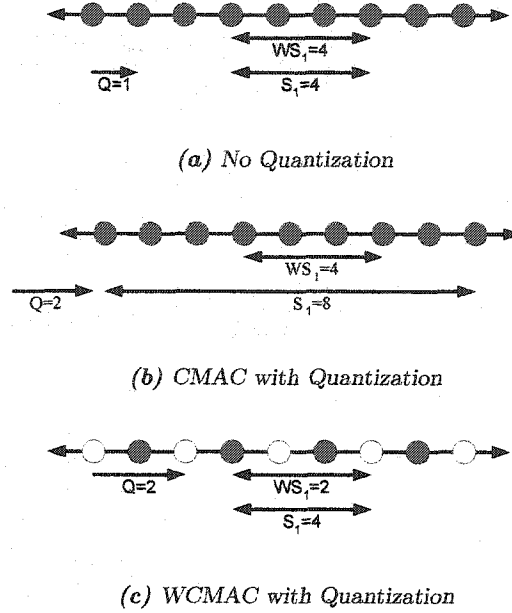
**Figure 5-3:** WCMAC Convergence for Different Harmonics (1D, C=20)

width only requires a small number of weights to be used. One of the strengths of the wavelet transform and also the Fast Fourier Transform(FFT) is the decimation process which reduces the computational load to the bare minimum value. The traditional CMAC uses a decimation process in multiple dimensions to reduce storage and computation. However, the quantization of the input stage in the traditional CMAC causes severe aliasing.

The standard quantization in CMAC reduces the input bit precision and essentially averages the target response over the quantized states. Furthermore, the generalization region grows by the same factor as the quantization. This further reduces the bandwidth of the system. The quantization only reduces the amount of weights stored, when it could also reduce the computational load.

As demonstrated, the WCMAC has a well-controlled frequency response and therefore quantization will be used to reduce the weight storage and the computational load, instead of quantizing the input values, which results in expanding the generalization region. The WCMAC quantizes the weight-space. The region of generalization remains constant with this type of quantization. For example, a WCMAC network with a generalization of 11 and a quantization of 4 accesses only 3 weights and covers an input space of 11 states.

A traditional CMAC with the same generalization and quantization parameters access 11 weights and covers an input of 44 states.



**Figure 5-4:** Quantization Effects in One-Dimension of CMAC vs. WCMAC

Figure 5-4(a) shows the one-dimensional CMAC implementation for a generalization of 4 without quantization. The nomenclature is:  $Q$  represents the quantization value (the arrow is referenced to appropriate weight or state-space),  $W$  is the number of weights to be accessed,  $S$  is the number of input states covered by the access. The weight addresses for the WCMAC with quantization is

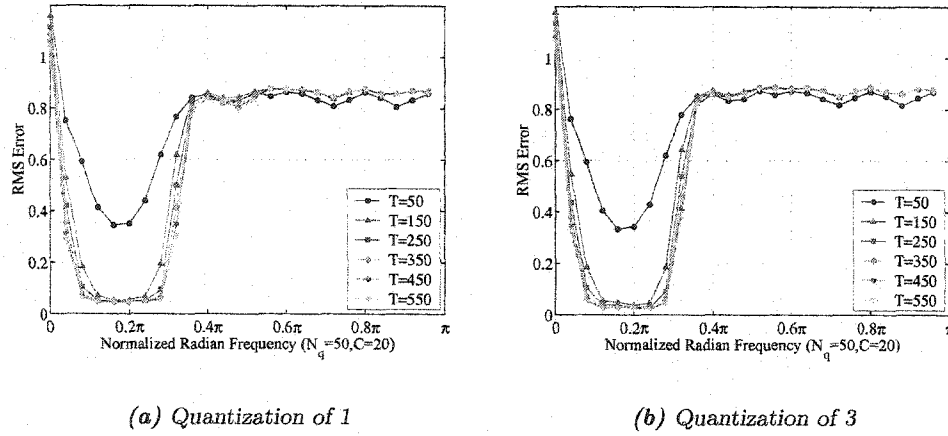
$$\mathbf{a}_s = \left[ \text{int}\left(\frac{x_s - \delta_r}{Q} + 1\right) \quad \text{int}\left(\frac{x_s - \delta_r}{Q} + 2\right) \quad \dots \quad \text{int}\left(\frac{x_s - \delta_r}{Q} + N_w\right) \right] \quad (\text{Eqn 5.9})$$

where  $N_w$  is the truncated integer value of  $C/Q$ . The number of weights to be computed is equal to  $N_w$ . The distance vector  $\mathbf{d}_s$  is the integer value of the distance to each weight

from the unquantized input vector.

$$\mathbf{d}_s = \begin{bmatrix} x_s - a_{s1}Q & x_s - a_{s2}Q & \dots & x_s - a_{sN_w}Q \end{bmatrix} \quad (\text{Eqn 5.10})$$

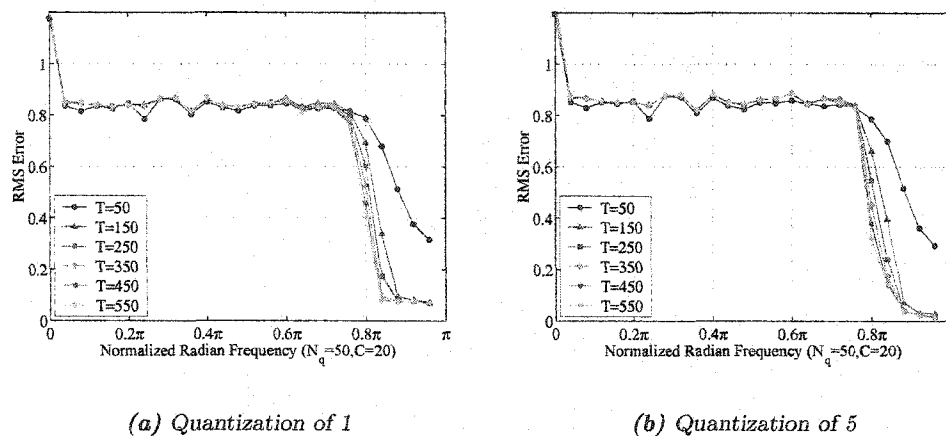
By quantizing the weight-space, the actual input state information is maintained and can be used to reference the weight location; see the above equation for  $\mathbf{d}_s$ . This allows the receptive field shape to interpolate between the weight-space and the input-space. If the receptive field is band-limited and the quantization is limited to avoid aliasing, the effective bandwidth of network is preserved and computational load is decreased by a factor of the  $C/Q$ . By quantizing the weight-space by 2, the upper half of the frequency band, ( $|f| > \pi/2$ ), folds on the lower half band, ( $|f| < \pi/2$ ). Therefore, the effective bandwidth of the network must be strictly limited to one of the half bands. Figure 5-5(a) and 5-5(b)



**Figure 5-5:** WCMAC Convergence for Different Quantizations (1D, C=20, M=1.1)

are one-dimensional convergence plots for the WCMAC with the quantization of 1 and 3, respectively. Both plots have the same modulation parameter of 1.1. There is no significant difference in capabilities of these two networks that can be seen in the plots. However, the computational load of figure 5-5(b) is reduced by a factor of 3.33. Since the weight-space is

quantized by a factor of 3, only 6 of the 20 weights actually exist, therefore, no computation is performed on weights removed by the quantization. The necessary weight storage is also reduced by a factor of 3. Although the weight storage is relatively insignificant in this example, it becomes far more important as the problems increase dimensionally.

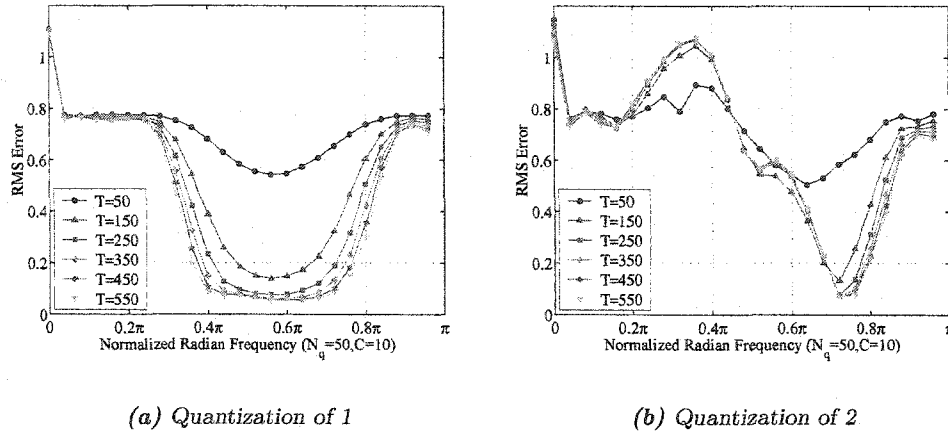


**Figure 5-6:** WCMAC Convergence for Different Quantizations (1D,  $C=20$ ,  $M=1.9$ )

The same principles of quantization work for the highpass filter structures. Figures 5-6(a) and 5-6(b) are the same network structure with a generalization of 20 and modulation of 1.9. Again, the difference is that figure 5-6(b) has a quantization of 5, reducing the computational load and the storage by that factor.

Problems arise when the effective bandwidth of the network overlaps the folding frequency, thus causing aliasing in the spectrum of the decimated weight-space. Figures 5-7(a) and 5-7(b) demonstrate this problem. The network configuration has a modulation value of 1.9, a generalization of 10 and quantizations of 1 and 2, respectively. The aliased upper half band changes the phase of the lower half band and increases the RMS error to a value greater than the RMS value of the function itself.

All spectral ranges can be learned and reconstructed in conjunction with decimation by



**Figure 5-7:** WCMAC Convergence with Aliasing Problems (1D, C=10, M=1.4)

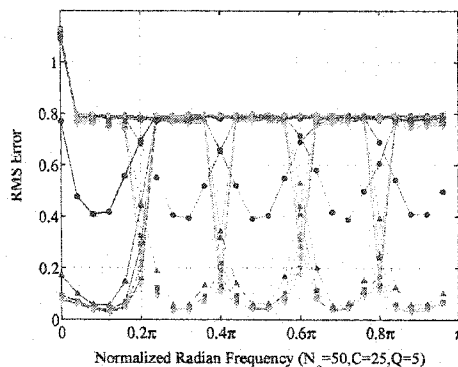
moving the folding frequencies outside the functional passband of the receptive field. For example, if the weight quantization is set to 5, there exists 5 regions where the spectrum can be acceptably reconstructed.

$$[(0 \dots 0.2\pi), (0.2\pi \dots 0.4\pi), (0.4 \dots 0.6\pi), (0.6 \dots 0.8\pi), (0.8 \dots \pi)] \quad (\text{Eqn 5.11})$$

Figure 5-8 shows five different convergence patterns overlaid, one for each region to be reconstructed. The generalization is expanded to 25 in order to reduce the width of the bandpass. This example demonstrates the ability to generalize widely over any region of the spectrum and still achieve the computational gain of quantizing the weights. It is also shown in the next chapter that different wavelets can be used together to develop full-bandwidth models.

## 5.4 Multidimensional WCMAC

The major difference between the one-dimensional and multidimensional CMAC implementation is the decimation that occurs in the weights space. In CMAC, the decimation is directly controlled by the generalization parameter and the lattice structure defined by the



*Figure 5-8: WCMAC Convergence Across Entire Spectrum (1D, C=25, Q=5)*

displacement vector. The traditional CMAC is band-limited by this decimation. As previously discussed, there have been multiple research projects focused on the placement of the weights after the decimation occurs. The major purpose is to evenly distribute the weights and still allow efficient calculation of their location. Another common requirement is that an even number of weights is constantly accessed. The general method involves a displacement vector and a modulus operation to bound the region.

The decimation limits the possible frequency response of the network and since this is tied to the generalization, the network can only widely generalize very low or very high frequencies. In the case of the traditional CMAC, the network can only learn the low frequency range when it widely generalizes.

The simplest adjustment to increase the bandwidth of the network is to remove the decimation. At the cost of increasing the computation dramatically, the network will compute a weight value for every state accessed within the generalization region. For example, a traditional two-dimensional network of a generalization of 10 will access 10 weights. In the non-decimated WCMAC, the network computes 100 weights. Clearly, this is not a computational advantage over the traditional network, but it does allow for any frequency band to be learned, if that was the required goal. In this base line, the weight-space becomes

a direct map of the input space. This is also a commonly used approach in Radial Basis Function networks, where all unique training pairs are given a receptive field center. It is important to notice that there is significant computation done to compute the receptive field centers. In the direct map method, a table can be precomputed that offsets the input state to all the selected weights.

The same basic wavelet function from the previous section is employed to learn the multidimensional function from the previous chapter. Once again, by modulating the wavelet function, the network can selectively learn any region of the input function. However, this involves a more complex receptive field structure. The traditional CMAC with a variable receptive field, use a look-up table that relates the receptive field strength to a distance function, like Manhattan, Euclidean, or minimal edge. This receptive field function works well for lowpass filters since it is radially distributed around the origin. However, it is difficult to form receptive field spectra with minimal side lobes using this method for highpass filters. The multidimensional wavelet decomposition and the multidimensional FFT handle multiple dimensions through separability. For example, the transform is processed on the rows and then the columns and the result is the product of the two values. This concept can be extended by the separation of each dimension and the wavelet receptive field function becomes

$$\psi_m(x, y, z) = \psi_1(x)\psi_2(y)\psi_3(z) \quad (\text{Eqn 5.12})$$

for the case of a three dimensional receptive field. In this chapter, the focus is on symmetric receptive fields where

$$\psi_1(x) = \psi_2(y) = \psi_3(z). \quad (\text{Eqn 5.13})$$

This is not a strict limitation of the network, and non-isotropic or asymmetric WCMAC implementations are covered in the next chapter. Once again, the address generation is a direct offset of the sample state. The corner of the receptive field is found as

$$\mathbf{x}'_s = \begin{bmatrix} x_{s1} - \delta_r & x_{s2} - \delta_r & \dots & x_{s3} - \delta_r \end{bmatrix}. \quad (\text{Eqn 5.14})$$



The number of offsets to all the weights in a generalization window is  $C^n$  where  $C$  is the generalization number and  $n$  is the number of dimensions in the problem. The offset table,  $\Delta_g$  is of size  $\{C^n, n\}$  and a single row is computed as

$$\Delta_{g\{i,1\dots n\}} = \left[ \text{int}\left(\left(\frac{i}{C^0}\right)\%C\right) \quad \text{int}\left(\left(\frac{i}{C^1}\right)\%C\right) \quad \dots \quad \text{int}\left(\left(\frac{i}{C^{n-1}}\right)\%C\right) \right] \quad (\text{Eqn 5.15})$$

where  $i = 1 \dots C^n$  and  $\%$  represents the modulus operator. The offset table is computed once during the initialization of the network since it is static for all inputs. The weight address table is identical in size to the offset table and a single row is computed as

$$\mathbf{A}_{s\{i,1\dots n\}} = \left[ x'_{s1} + \Delta_{g\{i,1\}} \quad x'_{s2} + \Delta_{g\{i,2\}} \quad \dots \quad x'_{sn} + \Delta_{g\{i,n\}} \right] \quad (\text{Eqn 5.16})$$

where  $i = 1 \dots C^n$ . At this point, the matrix can be linearized or hashed into a memory as is typically done in a CMAC system

$$\mathbf{w}_s = H(\mathbf{A}_s) \quad (\text{Eqn 5.17})$$

where  $H$  is a pseudorandom lookup table that translates the state address to virtual addresses and returns the value at each address. The distance vector expands to matrix  $\mathbf{D}_s$  with a row for each weight offset,

$$\mathbf{D}_{s\{i,1\dots n\}} = \left[ -\delta_r + \Delta_{g\{i,1\}} \quad -\delta_r + \Delta_{g\{i,2\}} \quad \dots \quad -\delta_r + \Delta_{g\{i,n\}} \right], \quad (\text{Eqn 5.18})$$

where the number of columns represents the number of input states. Finally, the value at any receptive field location is computed.

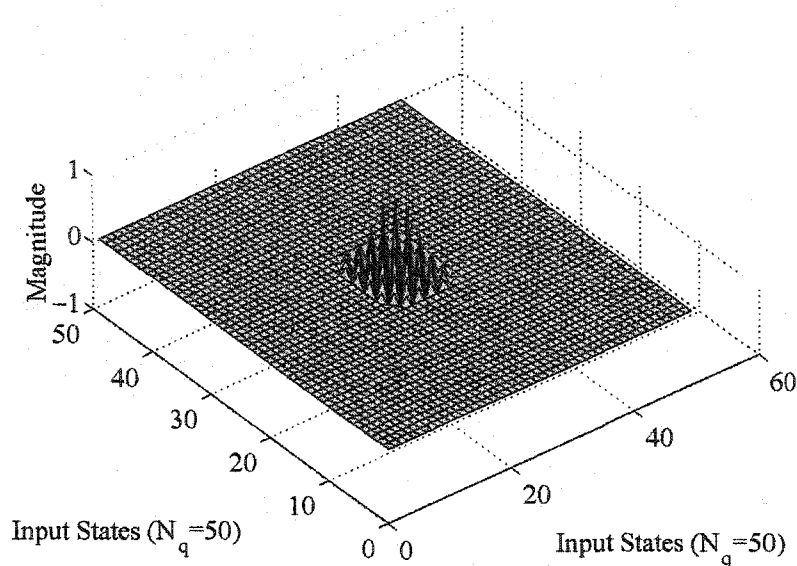
$$c_{si} = \psi(\mathbf{D}_{s\{i,1\}})\psi(\mathbf{D}_{s\{i,2\}})\dots\psi(\mathbf{D}_{s\{i,n\}}) = \prod_{j=1}^n \psi(\mathbf{D}_{s\{i,j\}}). \quad (\text{Eqn 5.19})$$

The entire receptive field is then formulated as

$$\mathbf{c}_s = \begin{bmatrix} c_{s1} & c_{s2} & \dots & c_{sl} \end{bmatrix} \quad (\text{Eqn 5.20})$$

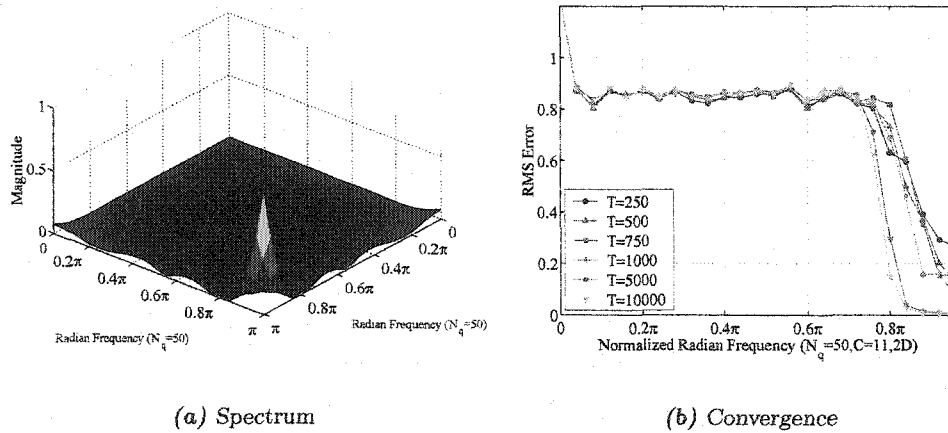
where  $l$  equals  $C^n$ . This method, along with the traditional CMAC address hashing, effectively vectorized any multidimensional problem to a one-dimensional problem that is suitable for standard computer architectures.

Figure 5-9 illustrates an example of a two-dimensional, high-frequency receptive field function. Clearly, the receptive field function is not radially symmetric. Therefore, the standard distance versus strength function is not appropriate for this receptive field function.



**Figure 5-9:** Wavelet Receptive Field in Two-Dimensions ( $C=11$ ,  $M=1.9$ )

The network is modified such that the receptive field strength is accessed from a fully-dimensional lookup table. The lookup table is the precomputed kernel function, or total receptive field. This does not add any computational load to the calculation of the receptive field over the variable receptive field CMAC model. However, it does require an increase in the size of the receptive field table. The receptive field can also be implemented as a single dimension lookup table or function for symmetric receptive fields, since it is the separation of the same function across each dimension. This requires slightly more computation during training.

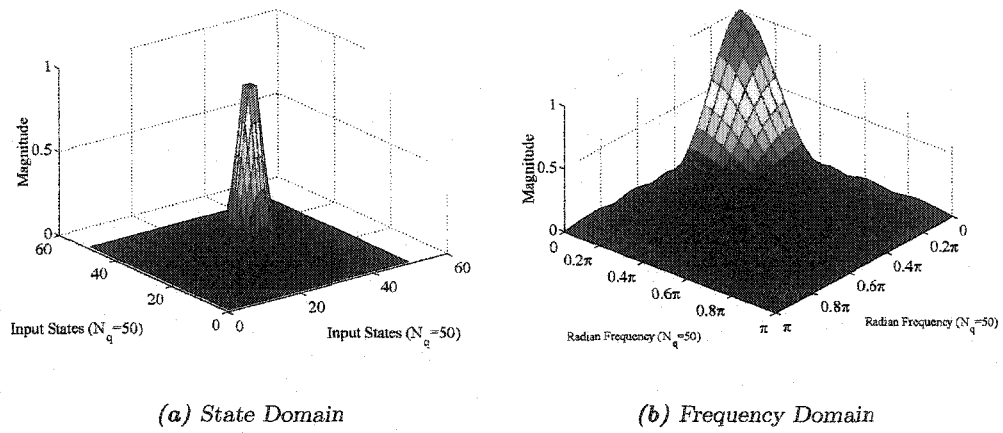


**Figure 5-10:** Wavelet Receptive Field Spectrum and Convergence (2D,  $C=11$ ,  $M=1.9$ )

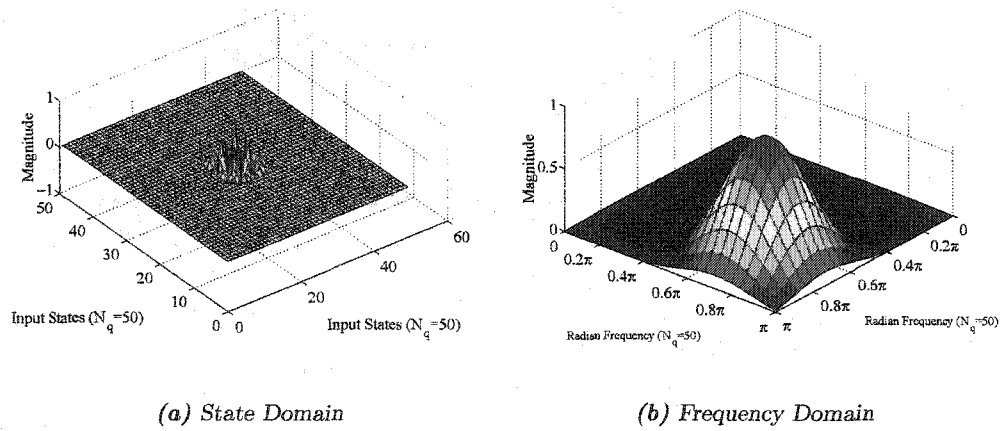
The corresponding WCMAC network with the receptive field in figure 5-9 produced the spectrum and convergence plot in figures 5-10(a) and 5-10(b). This example shows the flexibility of the WCMAC to learn multidimensional, high-frequency problems and still generalize in the state-space. The WCMAC is essentially a superset of the other low-pass CMAC models. Figures 5-11(a) and 5-11(b) use a modulation of 1 to produce the traditional lowpass function.

A variety of bandpass configurations are also easily designed and demonstrated in figures 5-12(a) and 5-12(b). The convergence plots for the multidimensional WCMAC with modulation constants of 1 and 1.4 are shown in figures 5-13(a) and 5-13(a). These WCMAC simulations with different receptive fields demonstrates a direct scaling of the non-decimated properties in the single dimension case to multiple dimensions.

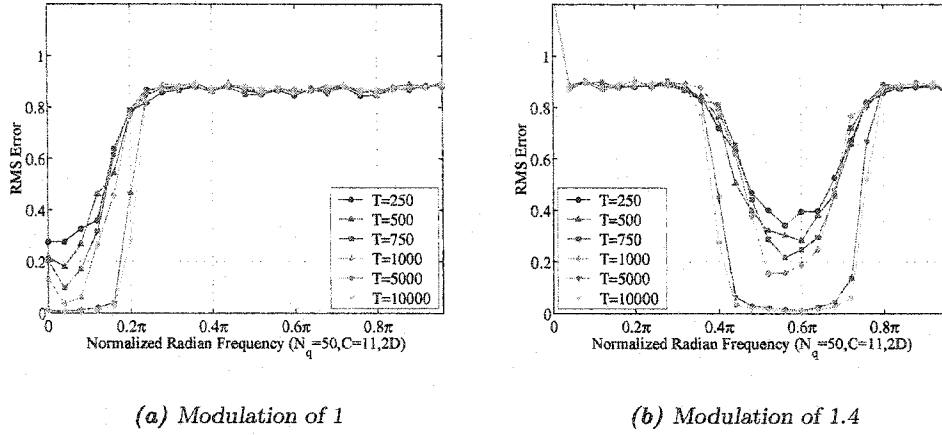
Although, all of the functions shown are symmetric functions based on the costrap function, asymmetric and other complex receptive fields can be handled with the modification of the receptive field function to a fully dimensioned look-up table. WCMAC has the capability to learn functions that other CMAC configurations were not previously capable of learning.



**Figure 5-11:** Wavelet Receptive Field in Two-Dimensions ( $C=11$ ,  $M=1$ )



**Figure 5-12:** Wavelet Receptive Field in Two-Dimensions ( $C=11$ ,  $M=1.4$ )



**Figure 5-13:** WCMAC Convergence for Different Harmonics (2D, C=11)

#### 5.4.1 Multidimensional Decimation and Lattice Structure

The previous section has extended the flexibility seen in the one-dimensional WCMAC to the multidimensional WCMAC. However, it has come at a large penalty in computation. In the previous examples, the WCMAC had a generalization of 11, which meant it accessed 121 weights during each training cycle. The traditional CMAC models would access only 11 weights, therefore, reducing the computational load by a factor of 11. In many cases, the decimation is warranted and when performed properly the decimation reduces the computational load without a loss in the bandwidth of the network.

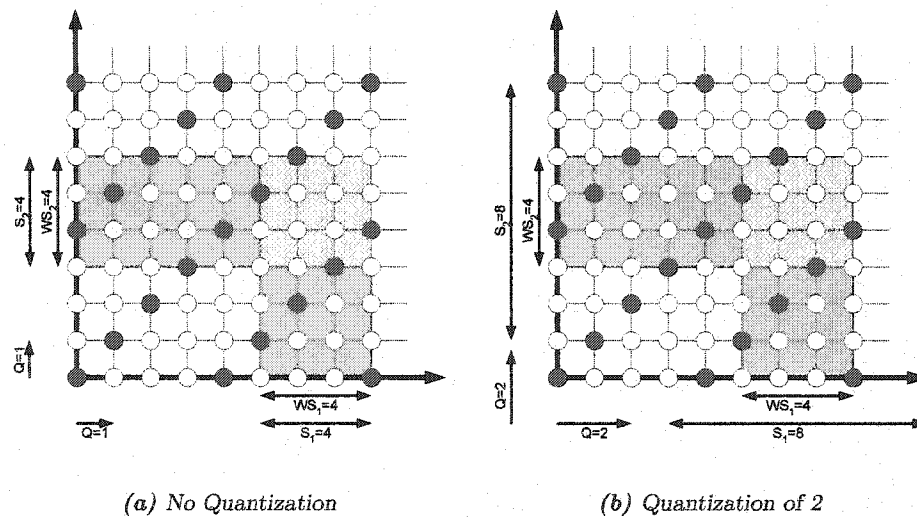
With the decimation of the weight-space controlled by the generalization and the lattice structure, there was essentially no flexibility in the original CMAC. An [65] changed this by introducing a new lattice structure that more evenly distributed the weights. However, it is still difficult to find the proper lattice structure for high-dimensionality problems. Furthermore, there was still no implementation that allowed for variation in the decimation factor. The fixed decimation factor,  $N_r$ , for the traditional CMAC is

$$N_r = C^{n-1} \quad (\text{Eqn 5.21})$$

where  $n$  is the number of input states or dimensions.

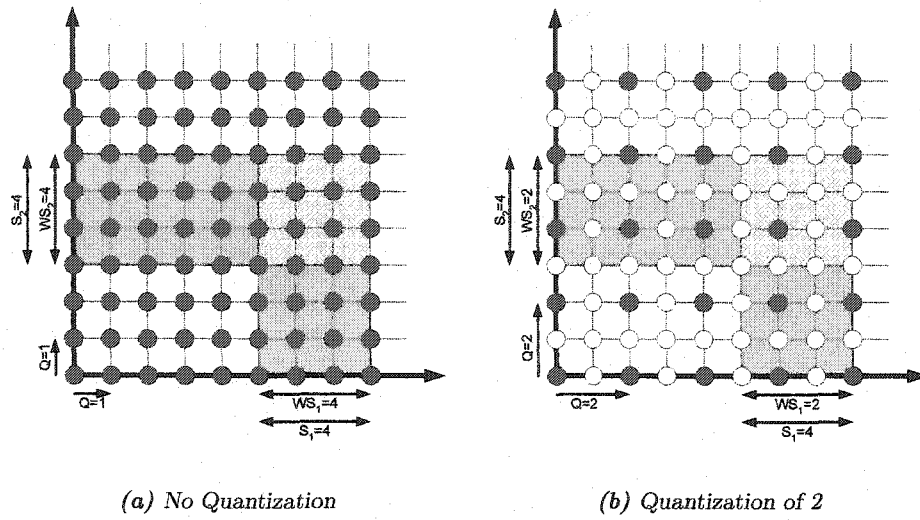
A rectangular decimation pattern for a symmetric multidimensional system would remove weights based on a diagonal decimation matrix. This would radially expand the spectrum in all directions equally. This structure is not conducive to the general theory that CMAC has an symmetric distribution of weights in each generalization region. However, it does produce globally symmetric weight distribution across the entire input space. The simplest way to produce a diagonal lattice matrix is to subsample each input state, thus removing entire rows and columns in the two-dimensional model.

Figure 5-14 shows the traditional CMAC weight distribution and the quantization effect. The lightly shaded blue area of the 4 by 4 weight matrix is the region of generalization for this input-state. The nomenclature is the same as figure 5-4(a). The difference between figure 5-14(a) and 5-14(b) is the quantization value,  $Q$ , of 1 and 2, respectively. The increase in quantization to 2 expands the generalization region in the state-space to an 8 by 8 matrix, as defined by the  $S$  arrows.



**Figure 5-14:** Quantization Effects in Two-dimension of CMAC

The original CMAC was designed based on generalization planes and the modulus operator. The WCMAC is designed to prove the appropriate decimation along any dimension. In the WCMAC, the modulus operation and staggered generalization planes are removed. A direct mapping of state-space to weight-space which replaces them. This is different from input quantization and is therefore called weight quantization.



**Figure 5-15:** Quantization Effects in Two-dimension of WCMAC

Figure 5-15 shows the effective quantization of the weight-space. The plot demonstrates that the number of weights is reduced by the decimation factor,

$$N_r = Q^n \quad (\text{Eqn 5.22})$$

where  $Q$  is the same quantization factor used in the one-dimensional case. For variable quantization across different axes,  $Q$  can be expanded to a vector. The number of activated weights in any region is significantly decreased but may not be centered in the generalization space. Figure 5-15(b) shows the off-center weights that are left after quantization. It is also important to note that coverage of the state-space remains constant. Since this is a direct

map of the input with decimation, it takes only a few operations to compute. The input of the target pair is offset by a predetermined and precomputed table, therefore, the typical modulus search operation is no longer necessary.

This method effectively compresses the weight-space versus the state-space. Since the WCMAC generalizes in the input-space, rather than the weight-space, the number of weights that need to be computed for any target pair has also been reduced by the decimation factor. A computational gain exists in both the weight value computation and the weight addressing. This direct-weight quantization technique solves the complex problem of finding the appropriate lattice structure with no computational overhead versus the original model. The density of the weight structure is left to the control of the user, so it can be optimized for the problem at hand. The corner of the receptive field is found as

$$\mathbf{x}'_s = \begin{bmatrix} x_{s1} - \delta_r & x_{s2} - \delta_r & \dots & x_{sn} - \delta_r \end{bmatrix} \quad (\text{Eqn 5.23})$$

if there is no quantization effect. The number of offsets to all the weights in the generalization window is reduced to  $(N_w)^n$  where  $N_w$  is the integer value of  $C/Q$  for the symmetric quantization case. The offset table,  $\Delta_g$  is of size  $\{(N_w)^n, n\}$  and is computed as

$$\Delta_{g\{i,1\dots n\}} = \begin{bmatrix} \text{int}((\frac{i}{(N_w)^0}) \% N_w) & \text{int}((\frac{i}{(N_w)^1}) \% N_w) & \dots & \text{int}((\frac{i}{(N_w)^{n-1}}) \% N_w) \end{bmatrix} \quad (\text{Eqn 5.24})$$

where  $i = 1 \dots (N_w)^n$ . The weight address table is identical in size to the offset table and a single row is computed as

$$\mathbf{A}_{s\{i,1\dots n\}} = \begin{bmatrix} \text{int}(\frac{x'_{s1}}{Q}) + \Delta_{g\{i,1\}} & \text{int}(\frac{x'_{s2}}{Q}) + \Delta_{g\{i,2\}} & \dots & \text{int}(\frac{x'_{sn}}{Q}) + \Delta_{g\{i,n\}} \end{bmatrix}. \quad (\text{Eqn 5.25})$$

The distance matrix,  $\mathbf{D}_s$ , is computed from each quantized weight address to the location of the state-space vector. This is one of the subtle differences between the WCMAC implementation and the traditional CMAC. The traditional CMAC computes the distance from the quantized input state, but WCMAC looks at the input state directly. The single row of

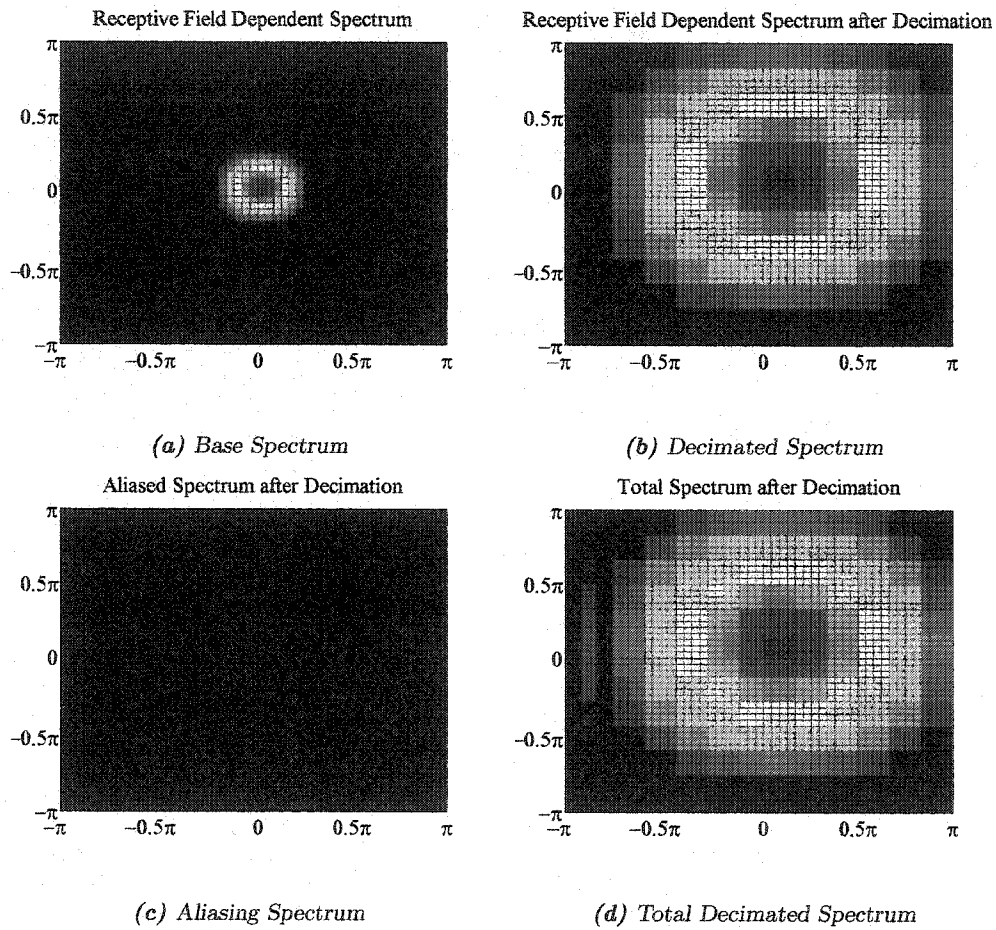


the distance matrix is

$$\mathbf{D}_{s\{i,1\dots n\}} = \begin{bmatrix} x_{s1} - \mathbf{A}_{s\{i,1\}}Q & x_{s2} - \mathbf{A}_{s\{i,2\}}Q & \dots & x_{sn} - \mathbf{A}_{s\{i,n\}}Q \end{bmatrix}, \quad (\text{Eqn 5.26})$$

$$= \mathbf{x}_s - \mathbf{A}_{s\{i,1\dots(N_w)\}}Q, \quad (\text{Eqn 5.27})$$

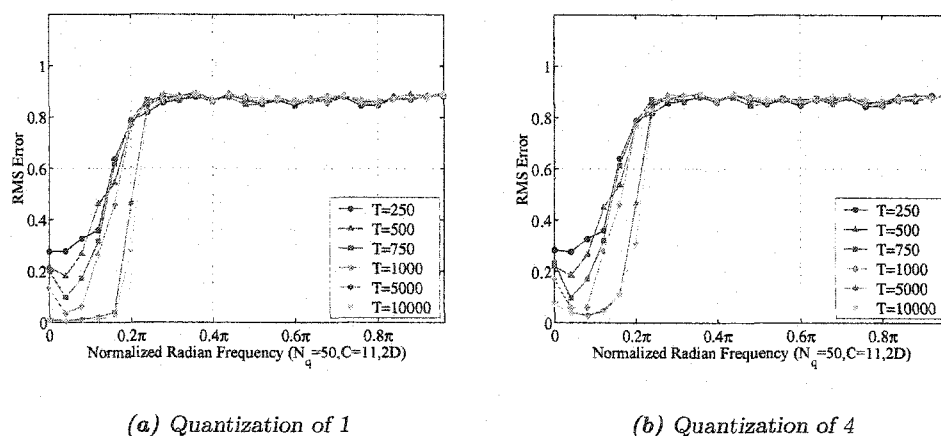
where there are  $(N_w)^n$  rows. The receptive field strength vector,  $\mathbf{c}_s$ , and the weight vector,  $\mathbf{w}_s$ , are now computed from  $\mathbf{D}_s$  and  $\mathbf{A}_s$ , as they are done in the full non-decimated model.



**Figure 5-16:** The Effect of weight-space Decimation (2D, C=11, Q=4, M=1.0)

Figure 5-16 shows the effect of decimation with this technique on a lowpass, two-

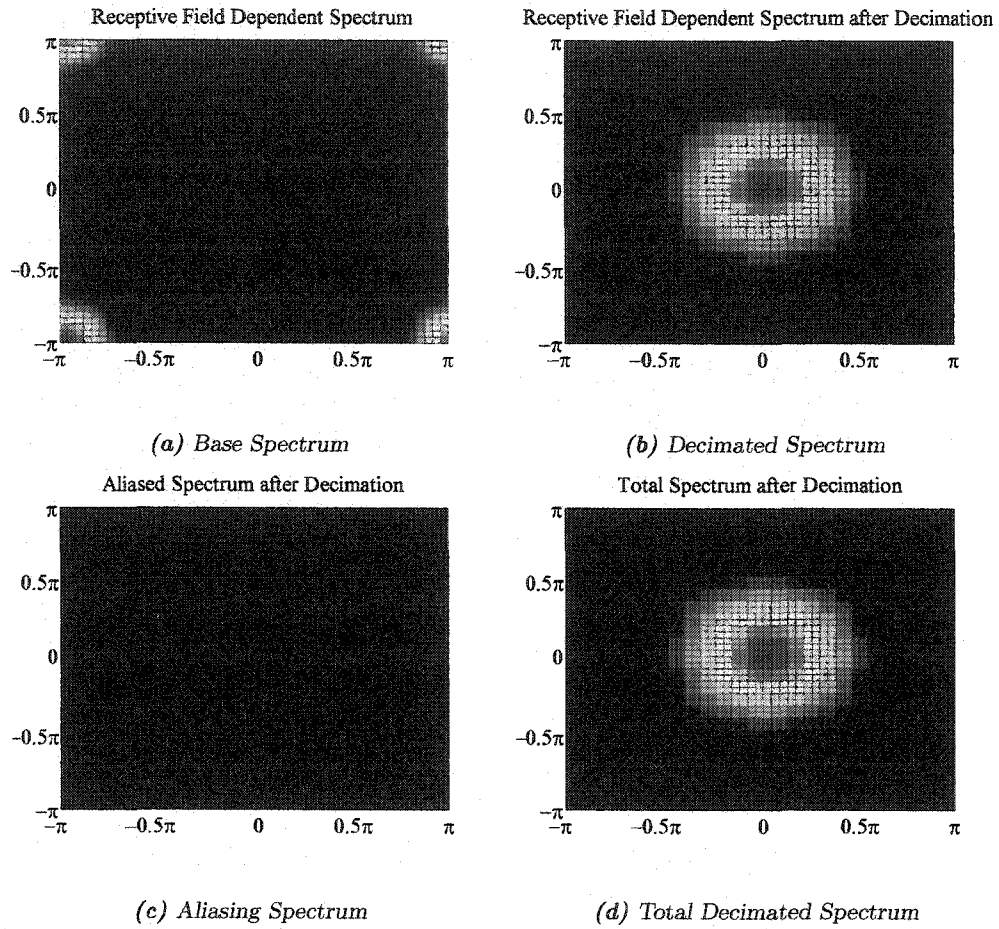
dimensional receptive field. This receptive field, which is similar to a linear receptive field, produces the spectrum in figure 5-16(a). After quantizing each dimension of the weight-space by 4, the spectrum expands radially but does not truncate any part of the main passband, figure 5-16(b). The spectrum is so well contained by the wavelet receptive field that any aliasing components are completely void from figure 5-16(c). Figure 5-16(d) shows there is no significant interference from other aliased spectra, and the result matches the decimated base spectrum shown in figure 5-16(b). Consequently, the passband of the network remains intact, while the computation is reduced by a factor of 16. This configuration has less computation than the variable receptive field CMAC model in both the number of weights and addresses to compute. The actual WCMAC simulation results of the quantized data versus the unquantized data are essentially the same, figures 5-17(a) and 5-17(b).



**Figure 5-17:** WCMAC Convergence for Different Quantization (2D, C=11, M=1)

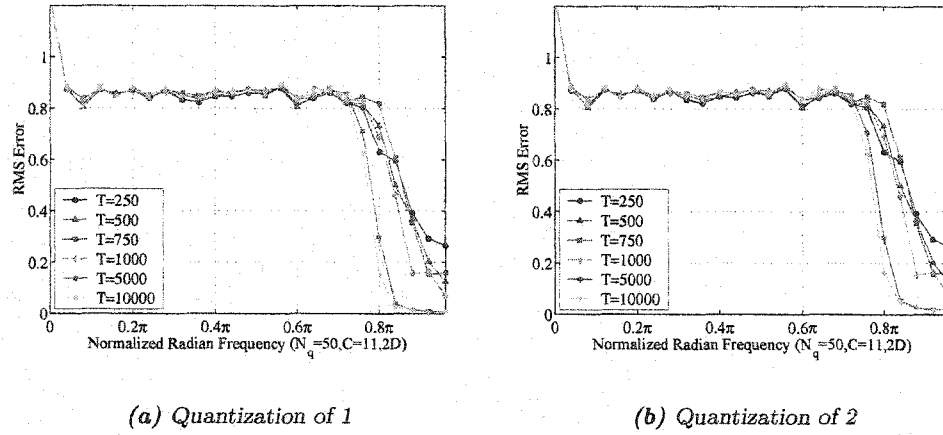
The highpass network can also be decimated to reduce both computation and storage. A WCMAC highpass spectrum is shown in figure 5-18 for a generalization of 11, a modulation 1.9 and a quantization of 2.

After decimation of a highpass band, the reduced sampling aliases the upper passband



**Figure 5-18:** The Effect of weight-space Decimation (2D,  $C=11$ ,  $Q=2$ ,  $M=1.9$ )

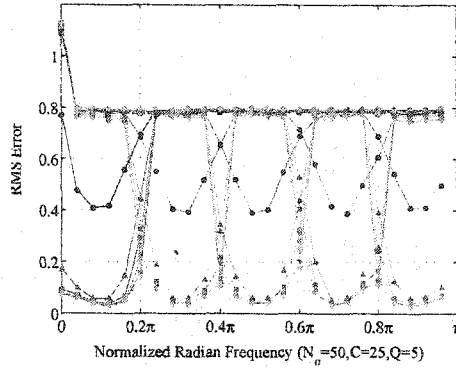
into the lower passband. Figure 5-18(b) is the expanded spectrum of figure 5-18(a), but aliased in to the low frequency region. It can clearly be seen that this quantization factor is not optimal, since the spectrum could be further spread without any aliasing. The WCMAC results are shown in figures 5-19(a) and 5-19(b). The quantized results show the same performance of the network; however, it requires only one quarter of the computation.



**Figure 5-19:** WCMAC Convergence for Different Quantization (2D, C=11, M=1.9)

As seen in the one-dimensional case, the WCMAC can also learn all of the mid-pass frequencies with decimation. The figure 5.4.1 shows the exact same plot that was used in the one-dimensional case, but is actually targeting a two-dimensional function. This plot is the overlaid convergence plot of five different CMAC simulations. The legend has been removed to show the entire spectrum, but the legend is identical to the legend in figure 5-19(b). This clearly demonstrates how the WCMAC has been expanded to handle any band of frequencies and effectively reduce computation and storage.

Through controlled decimation and direct weight mapping, the WCMAC computation has been reduced to, or possibly below, the computational level of the variable receptive field CMAC. The WCMAC has the ability to adjust the decimation level to optimize bandwidth



**Figure 5-20:** WCMAC Convergence Across the Entire Spectrum (2D, C=25, Q=5)

and computational concerns with the ability to learn any spectral region.

## 5.5 WCMAC Mathematical Model

The complete formulation for the WCMAC is given below, although, each equation has already been presented. This section is meant as a quick reference of the entire model. The input is a state-space vector,  $\mathbf{x}_s$ , of length  $n$ . The state-space vector is adjusted to find the corner of the generalization region,

$$\mathbf{x}'_s = \begin{bmatrix} x_{s1} - \delta_r & x_{s2} - \delta_r & \dots & x_{sn} - \delta_r \end{bmatrix}, \quad (\text{Eqn 5.28})$$

where  $\delta_r = \text{int}((C+1)/2)$ . The total number of weights accessed is equal to

$$N_{tw} = \prod_{j=1}^n \text{int}(C/Q) \quad (\text{Eqn 5.29})$$

where  $Q$  is the weight quantization value and  $C$  is the generalization parameter. The number of weights accessed along any input dimensions is

$$N_w = \text{int}(C/Q) \quad (\text{Eqn 5.30})$$

The offset from  $\mathbf{x}'_s$  to each weight activated is defined by the offset matrix,  $\Delta_g$ . One of the  $N_{tw}$  rows of the offset matrix is defined as

$$\Delta_{g\{i,1\dots n\}} = \begin{bmatrix} \text{int}((\frac{i}{(N_w)^0}) \% N_w) & \text{int}((\frac{i}{(N_w)^1}) \% N_w) & \dots & \text{int}((\frac{i}{(N_w)^{n-1}}) \% N_w) \end{bmatrix} \quad (\text{Eqn 5.31})$$

A single row of the weight address table is computed as

$$\mathbf{A}_{s\{i,1\dots n\}} = \begin{bmatrix} \text{int}(\frac{x'_{s1}}{Q}) + \Delta_{g\{i,1\}} & \text{int}(\frac{x'_{s2}}{Q}) + \Delta_{g\{i,2\}} & \dots & \text{int}(\frac{x'_{sn}}{Q}) + \Delta_{g\{i,n\}} \end{bmatrix} \quad (\text{Eqn 5.32})$$

where the number of rows is identical to the rows  $\Delta_g$ . Each address row indexes a single weight through

$$\mathbf{w}_s = H(\mathbf{A}_s) \quad (\text{Eqn 5.33})$$

where  $H$  is a hashing function. The weight vector has exactly  $N_{tw}$  elements. The distance of each weight to the original input,  $\mathbf{D}_s$ , is computed from each quantized address location. The  $i^{\text{th}}$  row of the distance matrix is

$$\mathbf{D}_{s\{i,1\dots n\}} = \mathbf{x}_s - \mathbf{A}_{s\{i,1\dots n\}} Q \quad (\text{Eqn 5.34})$$

where there are again  $N_{tw}$  rows. A receptive field strength value is computed from each row of the distance matrix as

$$c_{si} = \prod_{j=1}^n \psi(\mathbf{D}_{s\{i,j\}}) \quad (\text{Eqn 5.35})$$

where the wavelet function,  $\psi$ , in this case is defined as

$$\psi(x) = \text{costrap}(x, a, b, M), \quad (\text{Eqn 5.36})$$

and  $a$  is considered the dilation parameter,  $b$  is the translation parameter and  $M$  is the modulation parameter. The entire receptive field is then formulated as

$$\mathbf{c}_s = \begin{bmatrix} c_{s1} & c_{s2} & \dots & c_{sL} \end{bmatrix}, \quad (\text{Eqn 5.37})$$

and the output is formulated as

$$y_s = \frac{\mathbf{c}_s \mathbf{w}_s}{\|\mathbf{c}_s\|}. \quad (\text{Eqn 5.38})$$

Finally, the weight update algorithm for the WCMAC is

$$\Delta \mathbf{w}_s = \alpha \frac{\mathbf{c}_s^T}{\|\mathbf{c}_s\|} (\hat{y}_s - y_s) \quad (\text{Eqn 5.39})$$

where  $\hat{y}_s$  is the desired system response. The WCMAC also scales to multiple outputs for a given input in the same manner as the original CMAC.

## 5.6 Discussion

The traditional CMAC model has well-defined limitations, particularly the inability to effectively learn high frequencies and, especially, widely generalize high frequencies. The WCMAC model, developed in this chapter, uses a flexible wavelet receptive field to allow functional learning in a larger region of the Nyquist spectra. It also extends easily to higher-dimension problems.

Another issue with the traditional CMAC models is the decimation of the weight-space. The WCMAC has a configurable decimation routine for the weight-space. The weight-space is mapped directly to the state-space and the decimation is controlled through quantization or sub-sampling of the weights. The receptive field strength is referenced to the input-space. This method forces the weights to evenly distribute on the state-space, which may not be true for a particular generalization zone.

Furthermore, this method is analogous to the sub-sampling methods in signal and image processing. The direct mapping also simplifies weight address generation, thus reducing computation. The computation is also controlled by the quantization and can be used in the appropriate situation to increase the efficiency of the WCMAC algorithm. The WCMAC model is a more flexible model than the traditional CMAC. It can learn a wider range of functions with decreased computation, in many cases.

# CHAPTER 6

## EXTENSIONS TO THE WCMAC

### 6.1 Introduction

The formulation of the WCMAC in the previous chapter demonstrated the ability to learn different frequency ranges by modifying the receptive field function and applying it symmetrically across all input dimensions. This allows the implementation to use a single lookup table for the receptive field function, and single values for both the generalization and decimation factors. This is in-line with the traditional CMAC models and therefore can be easily adapted to fit existing CMAC implementations.

In this chapter, each input dimension is designed independently. This requires more resources than the traditional CMAC implementations. Consequently, it is discussed separately. A similar concept was applied to the original CMAC, called the generalized CMAC or GCMAC [74]. The concept allows the standard hypercube generalization to be expanded to a hyperparallel-piped structure with varying dimensions along each state-space axis. The WCMAC is easily expanded to allow independent generalization values and receptive field models along each access. The decimation can also be optimized along each axis. In other words, the network is asymmetric or non-isotropic along the input dimensions. This expanded network is referred to as the Asymmetric WCMAC (AWCMAC). The additional flexibility allows the AWCMAC to learn any frequency range in the multidimensional case. Finally, the different receptive field layers are combined to develop full bandwidth models that can learn the entire input frequency range or other wide-band frequency responses.



## 6.2 Asymmetric WCMAC

In the previous chapter, the receptive field was equal across all the input dimensions of the network

$$\psi_1(x_1) = \psi_2(x_2) = \dots = \psi_n(x_n) \quad (\text{Eqn 6.1})$$

where  $n$  is the number of input dimensions in the network. The asymmetric WCMAC allows for the case of each input dimension being independent or

$$\psi_1(x_1) \neq \psi_2(x_2) \neq \dots \neq \psi_n(x_n) \quad (\text{Eqn 6.2})$$

with a different wavelet function across each dimension. Since the effective bandwidth and center frequency of each input dimension is different, the generalization and decimation factor are also specific to the input axis. The generalization is represented in the form

$$\mathbf{C} = \begin{bmatrix} C_1 & C_2 & \dots & C_n \end{bmatrix}, \quad (\text{Eqn 6.3})$$

previously,  $\mathbf{C}$  could be represent as single scalar value. The decimation factor is represented as

$$\mathbf{Q} = \begin{bmatrix} Q_1 & Q_2 & \dots & Q_n \end{bmatrix}. \quad (\text{Eqn 6.4})$$

The decimation vector can also be translated into a basic lattice matrix

$$M_{AWCMAC} = \begin{bmatrix} Q_1 & 0 & 0 \\ 0 & Q_2 & 0 \\ 0 & 0 & Q_3 \end{bmatrix}, \quad (\text{Eqn 6.5})$$

which defines the weight locations. This lattice is typically referred to as the rectangular sampling lattice. It is commonly used in image processing and other multidimensional signal processing algorithms. There are some advantages to other lattice structures, like polar and hexagonal. It has been shown that some problems, for example phased array antennas, can be designed with few samples, if a nonrectangular lattice is used[97]. However, the rectangular lattice is easy to implement and design, particularly in the case of this

asymmetric network. The adaptive modeling techniques of neural networks are typically applied to problems that are difficult to model or when there is limited knowledge of the function to be learned. Therefore, it can be difficult to design the exact lattice structure for a given problem. The rectangular lattice has the lowest computational load and any limitation of the lattice could simply be addressed by over-sampling, i.e reducing the decimation of weights.

The total number of weights,  $N_{tw}$ , accessed for the AWCMAC is simply the product of the number of weights across each dimension or

$$N_{tw} = \prod_{j=1}^n \text{int}(C_j/Q_j). \quad (\text{Eqn 6.6})$$

The number of weights accessed on each dimension is

$$N_{wj} = \text{int}(C_j/Q_j). \quad (\text{Eqn 6.7})$$

where  $j$  is the reference to the appropriate input state. As done in the WCMAC implementation, the state-space vector is adjusted to find the corner of the generalization region,

$$\mathbf{x}'_s = \begin{bmatrix} x_{s1} - \delta_{r1} & x_{s2} - \delta_{r2} & \dots & x_{sn} - \delta_{rn} \end{bmatrix}, \quad (\text{Eqn 6.8})$$

where

$$\delta_{rj} = \text{int}((C_j + 1)/2). \quad (\text{Eqn 6.9})$$

It is important to note that there is a different generalization value,  $C_j$ , for each input dimension. The offset from  $\mathbf{x}'_s$  to each weight activated is again defined by the offset matrix,  $\Delta_g$ . One of the  $N_{tw}$  rows of the offset matrix is defined as

$$\Delta_{g\{i,1\dots n\}} = \begin{bmatrix} \text{int}((\frac{i}{1}) \% N_{w1}) & \text{int}((\frac{i}{N_{w1}}) \% N_{w2}) & \dots & \text{int}((\frac{i}{N_{w1}N_{w2}\dots N_{w(n-1)}}) \% N_{wn}) \end{bmatrix} \quad (\text{Eqn 6.10})$$

A single row of the weight address table is computed as

$$\mathbf{A}_{s\{i,1\dots n\}} = \begin{bmatrix} \text{int}(\frac{x'_{s1}}{Q_1}) + \Delta_{g\{i,1\}} & \text{int}(\frac{x'_{s2}}{Q_2}) + \Delta_{g\{i,2\}} & \dots & \text{int}(\frac{x'_{sn}}{Q_n}) + \Delta_{g\{i,n\}} \end{bmatrix} \quad (\text{Eqn 6.11})$$

where the number of rows is identical to the rows  $\Delta_g$ . The equation for the receptive field strength values remains the same as the WCMAC case,

$$c_{si} = \prod_{j=1}^n \psi_j(\mathbf{D}_{s\{i,j\}}). \quad (\text{Eqn 6.12})$$

The computation for the output from the receptive field strength is also unchanged

$$y_s = \frac{\mathbf{c}_s \mathbf{w}_s}{\|\mathbf{c}_s\|}, \quad (\text{Eqn 6.13})$$

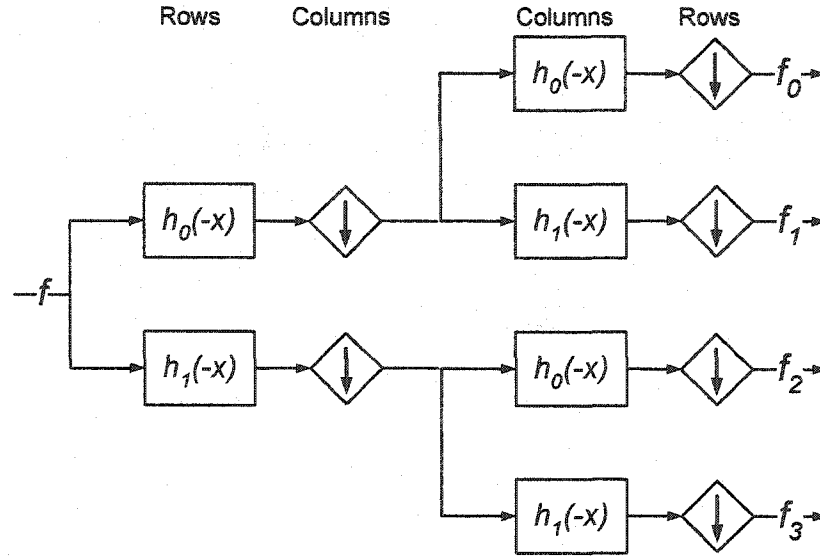
as well as the weight update algorithm

$$\Delta \mathbf{w}_s = \alpha \frac{\mathbf{c}_s^T}{\|\mathbf{c}_s\|} (\hat{y}_s - y_s). \quad (\text{Eqn 6.14})$$

The only major difference in the computation is the need to maintain tables for the receptive field across each input dimension, or to maintain a multidimensional table that is precomputed for the entire generalization region.

As seen in the previous chapter, the generalization width and the weight decimation need to be designed in concert to effectively maximize bandwidth, while still minimizing the computational load. With the AWCMAC model, each input is designed individually and put together to form the complete network model. The practice of having various generalization widths and receptive field functions along different input states is not uncommon in controls applications. The inputs, or states, are often controlled by various types of sensors, like accelerometers, pressure sensors or other feedback systems. This was one of the driving forces that lead to the development of the GCMAC model, which handles different generalization widths across each dimension, but was still limited to the lowpass frequency range. The AWCMAC is essentially a wavelet based GCMAC.

The process of treating each dimension separately is also not uncommon in multidimensional signal processing. Image processing techniques, such as the FFT and DWT, compute each dimension separately during the transform. This process essentially mirrors the rectangular lattice structure. Figure 6-1 demonstrates the filtering and decimation process



**Figure 6-1:** Discrete Wavelet Transform Data Flow

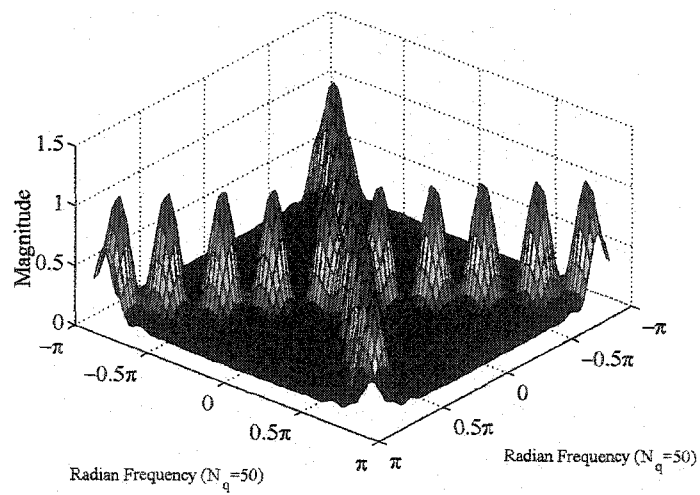
done by the Discrete Wavelet Transform for a two-dimensional image. It is clearly evident that rows and column are computed separately. The same process is used to design the AWCMAC and, in fact, the DWT was the inspiration for the AWCMAC algorithm.

A map of the decomposed spectrum for the DWT data flow is shown in figure 6-2. By applying the same wavelet function across each dimension the WCMAC is only capable of learning the  $f_0$  and  $f_3$  regions. A traditional CMAC model is only capable of learning the  $f_0$  spectral region.

However, the AWCMAC model can effectively model all four regions,  $f_0$ ,  $f_1$ ,  $f_2$  and  $f_3$ . Figure 6-3 demonstrates the learning regions for the symmetric WCMAC model for the Fourier series analysis covered in the previous chapter. This particular wavelet receptive field is designed for a decimation factor of 5 and a generalization of 25 along each input dimension. This symmetry forces the supported frequency regions along the diagonals of the frequency domain plot. The AWCMAC does not have this limitation and can support

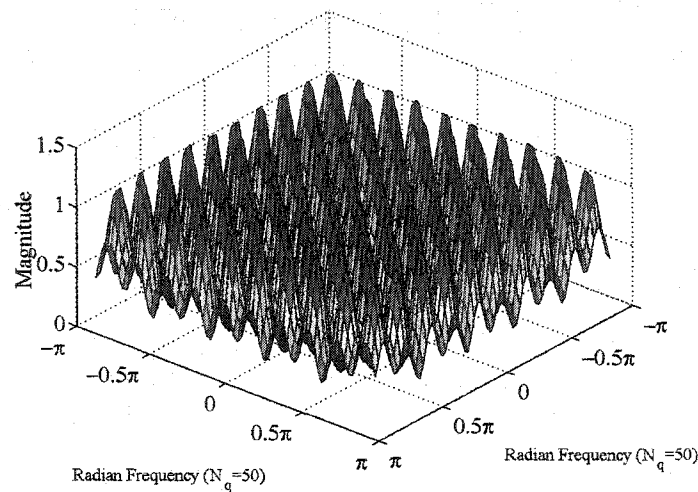
$f_3$	$f_1$	$f_3$
$f_2$	$f_0$	$f_2$
$f_3$	$f_1$	$f_3$

**Figure 6-2:** Decomposition in the Frequency Domain



**Figure 6-3:** Symmetric WCAMC in the Frequency Domain (2D)

effective learning regions across the entire spectrum, as shown in figure 6-4. The resulting enhancements cover the entire Nyquist or sampled spectrum. This result of the AWCMAC further extends the capabilities of the CMAC to learn an even larger set of functions. It should also be noted that many of these functions could be learned by a traditional CMAC, if they were modulated to the base-band frequency or used a creative input mapping scheme. However, the AWCMAC model does not require these additional operations, so it is easier to design and has a higher computational efficiency.

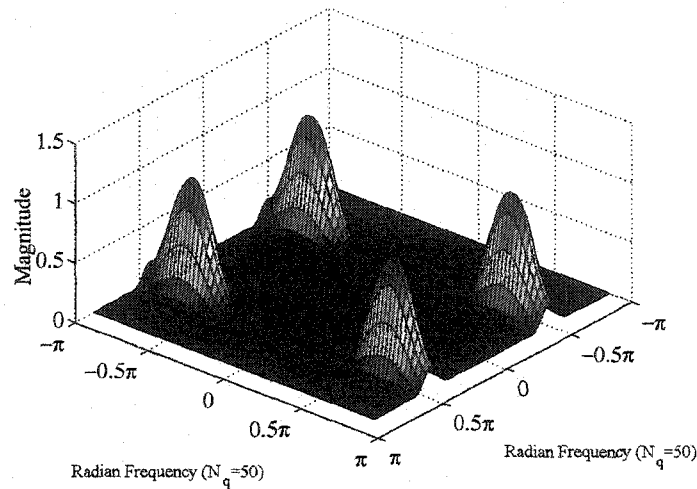


*Figure 6-4: Asymmetric WCMAC in the Frequency Domain (2D)*

### 6.2.1 Input Quantization for WCMAC models

The final component of the traditional CMAC that has not been covered in the WCMAC models is the support for different quantization values along each input dimension. This is known as input, or state-space quantization in the traditional CMAC models. Input quantization in the traditional CMAC increases the generalization by spreading the effective region covered by the receptive fields. As seen in the chapter on spectral properties of

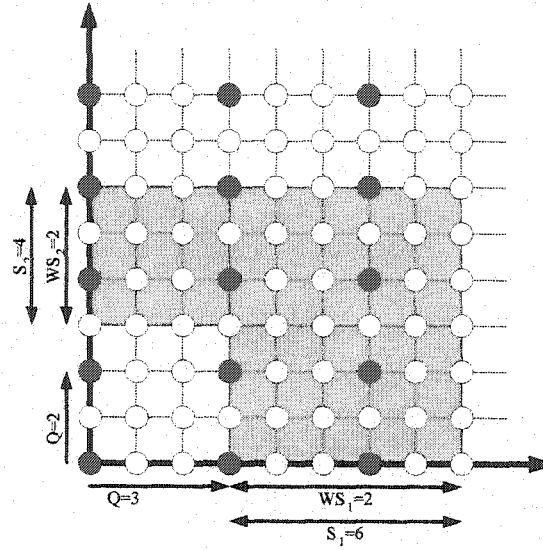
CMAC, this can induce severe aliasing and limit the effectiveness of the model. Furthermore, CMAC maps each block of quantized input states to a single weight. As long as the learning rate is significantly low, the CMAC learns the average target response for the block of quantized states. Averaging is a lowpass filtering effect. Therefore, quantizing the inputs in this manner, will effect the WCMAC's ability to learn the higher frequency functions.



*Figure 6-5: Asymmetric WCMAC with Different Generalization Values (2D)*

With the AWCMAC model, the same control is applied by directly controlling the decimation factor of the weights as well as the receptive field shape and width. A subtle, but important difference is that the WCMAC model references the state-space vector directly, whenever calculating the receptive field distance function. This allows the network to interpolate between the weights with the modulated wavelet functions. This is essential for learning frequency bands above the lowpass region. Figure 6-5 shows the frequency response of a two-dimensional network with variable generalization widths along the different input axes. The different spectral widths call for varied decimation factors across the different input dimensions in order to optimize the computation. An example of this type of weight-

space is shown in figure 6-6.



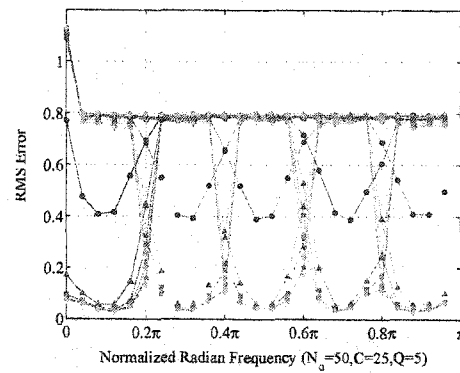
**Figure 6-6:** Asymmetric WCMAC Weight Decimation (2D)

### 6.3 Wide or Full Bandwidth WCMAC Models

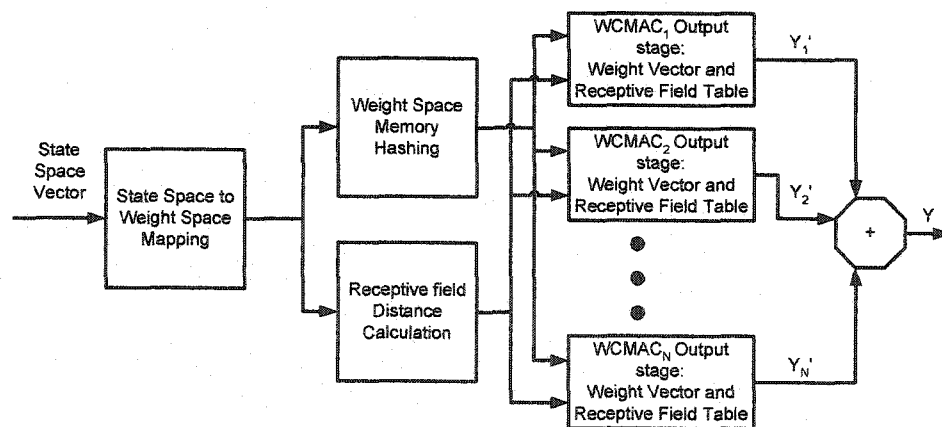
The combination of the previously discussed WCMAC models, including both the symmetric and asymmetric versions, have the capability to learn a very large range of functions. The weight-space mapping and receptive functions can be further combined into single networks to learn full bandwidth functions or more complex spectral patterns built from different wavelet functions. Figure 6-7 is repeated from the previous chapter. It is clear from this figure that different WCMAC receptive field functions can be designed to learn each spectral component of this one-dimensional problem.

The weight-space and output stages can be grouped into a single network, as shown in figure 6-8. Some additional computation is required to subtract the effects of each individual





**Figure 6-7:** WCMAC Convergence Across Entire Spectrum (1D,  $C=25$ ,  $Q=5$ )



**Figure 6-8:** Wide Band Model

output stage during the learning process. Without this step, each output stage attempts to correct for the entire error. This essentially multiplies the learning coefficient by the number of output stage and can severely effect the stability of the network. Each separate weight-space,  $\mathbf{w}_s^N$ , is updated by the traditional learning function,

$$\Delta \mathbf{w}_s^N = \alpha \frac{\mathbf{c}_s^{NT}}{\|\mathbf{c}_s^N\|} (\hat{y}_s'^N - y_s^N) \quad (\text{Eqn 6.15})$$

except the target function,  $\hat{y}_s'^N$ , is the desired response minus the output generation value of the other output stages. For example, the desired response for the first weight vector is

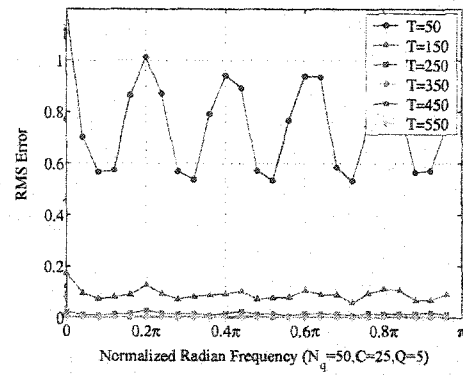
$$\hat{y}_s'^1 = \hat{y} - y_s^2 - y_s^3 \cdots - y_s^N. \quad (\text{Eqn 6.16})$$

The receptive field activation vector,  $\mathbf{c}_s^N$ , used in the update algorithm, is also specific to the weight vector being updated, since they are all built from different wavelets.

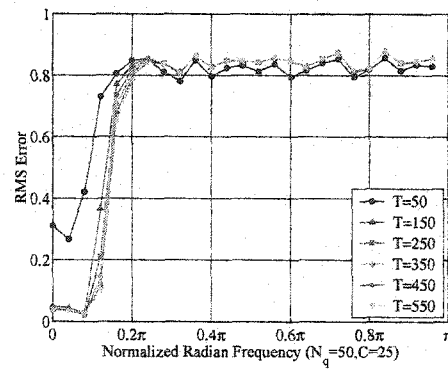
However, the weight indexing, hashing and receptive field indexing are computed only once for all output stages. Depending on the CMAC size and configuration, these computations can be the largest component of the CMAC algorithm. The learning response of a concatenated model based on the wavelet-based CMAC models which produced the simulation results in figure 6-7 shows the model converges for all spectral components, see figure 6-9.

The corresponding learning response for a CMAC network with a linear variable receptive field model is shown in figure 6-10. It is significant to point out that the wide bandwidth model and the CMAC with linear receptive fields have essentially the same computation. The traditional CMAC computes all the positions and activation functions for 25 weights, while the WCMAC model only computes 5 weight positions and simply uses 5 different receptive field functions.

The same process can be used to develop multidimensional WCMAC models that learn wider or more complex spectral regions. Once again, the same models used in the previous chapter are combined to construct the wider bandwidth model. The frequency response of



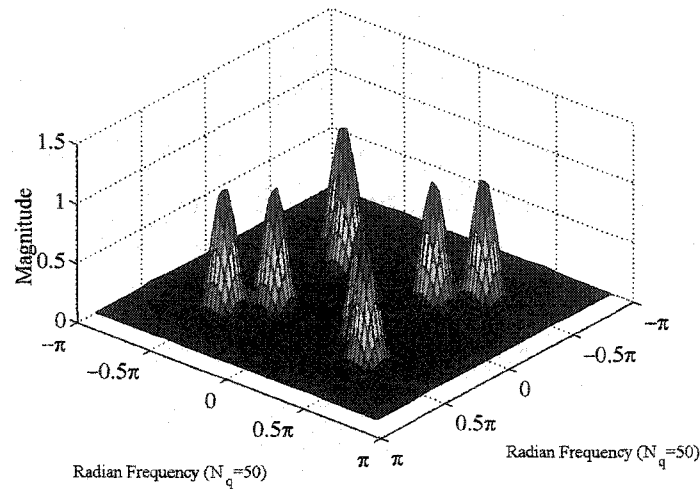
**Figure 6-9:** WCMAC Convergence with Full Bandwidth (1D, C=25, Q=5)



**Figure 6-10:** CMAC Convergence with Linear Receptive Field (1D, C=25, Q=1)

the combined WCMAC model in figure 6-11 produces the learning convergence in figure 6-12. For clarity, the function being learned is once again given as,

$$z = \sin \left( \frac{2\pi F}{50}(x + y) \right). \quad (\text{Eqn 6.17})$$

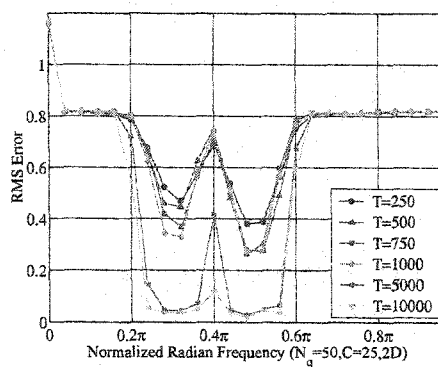


**Figure 6-11:** Spectrum of WCMAC with Combined Receptive Fields

The separation between the frequency response of the two WCMAC receptive fields is clearly visible in the learning convergence and the frequency domain plot, by the slow convergence at the center of the supported learning region. Learning could be improved in the region by including the asymmetric receptive field cases, surrounding the two functions chosen in this simulation.

## 6.4 Discussion

This chapter completes the WCMAC model by demonstrating methods to effectively learn the entire Nyquist spectrum. The WCMAC presented has all the capabilities of the



**Figure 6-12:** Convergence of WCMAC with Combined Receptive Fields

previous CMAC models mentioned, but supersedes its predecessor by including entirely new classes of functions that can be learned. Furthermore, by designing each input independently, the quantization of weights and generalization widths can be tuned to optimize the computational load and learning convergence of the network. Broad bandwidth models can be solved by including receptive fields of a variety of frequency ranges into a single network. The WCMAC has essentially been expanded to cover problems typically handled by wavelet transforms, Fourier transforms and complex filter banks.

# CHAPTER 7

## STABILITY AND CONVERGENCE PROPERTIES OF WCMAC

### 7.1 Introduction

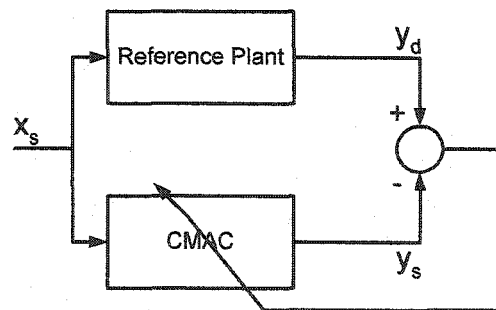
The previous chapters demonstrated the ability of WCMAC network configurations to model different functions that represented a Fourier analysis but only using a single learning rate. In this chapter, the stability boundaries with respect to the learning rate are explored. The WCMAC learning algorithm is analyzed with the Lyapunov method using a reference CMAC. This is the method originally used by Campagna and Kraft to prove the stability of the traditional CMAC [5]. There have also been other stability studies done on the traditional CMAC. However, the variable strength receptive field models and other advanced CMAC models have not been studied. It is also shown that the stability proof for the WCMAC is inclusive of the three CMAC models of interest to this dissertation: the Albus or binary receptive field CMAC, the variable strength receptive field model and the WCMAC.

A second learning algorithm that is designed to reduce the computational load is introduced and examined through the same stability analysis. This learning algorithm uses a scalar to approximate the Euclidean norm, used in the output formulation and weight update algorithm. The stability analysis gives insight to the effect of this simplification. The

reduced computation model and the standard learning rate are compared in terms of the learning convergence. Furthermore, the convergence of the WCMAC is directly compared to the learning convergence of the traditional CMAC models.

## 7.2 CMAC Stability

The following Lypanov stability proof was published by Campanga and Kraft[5]. It is simply reformulated here to be consistent with the notation of this dissertation. Figure 7-1 shows the basic configuration used in the open-loop stability analysis. The reference plant is typically considered a CMAC that has already been trained. Therefore, the analysis proves the stability of CMAC for the class of functions which can be modeled with the reference plant. The two systems, CMAC and the reference plant, are assumed to have the same weight structure and generalization value. The common state-space vector,  $x_s$ , essentially keeps the systems in lock step. Only the memory in the CMAC is being updated, while the reference system is assumed to have static memory. The results of the analysis bound the range of stable values for the learning rate.



*Figure 7-1: Base System for CMAC Stability*

The proof is then extended to the WCMAC. In the previous chapter, the  $c_s$  was the

receptive field strength of all the nonzero values. For simplicity of the Lyapunov proof,  $\mathbf{c}_s$  is now considered equal in length to all the weights in the system, and the weight vector,  $\mathbf{w}_s$ , is also the same length.

The error signal at each training point  $s$  is defined as

$$e_s = (\hat{y}_s - y_s) \quad (\text{Eqn 7.1})$$

where  $\hat{y}_s$  is defined as  $y_d$  in the figure 7-1. Since both systems are accessed by the state-space vector, they have identical receptive field activation vectors. Using this property, the error function can be simplified to the difference in the weight vectors,

$$e_s = \mathbf{c}_s \hat{\mathbf{w}}_s - \mathbf{c}_s \mathbf{w}_s = \mathbf{c}_s \delta \mathbf{w}_s. \quad (\text{Eqn 7.2})$$

The Lyapunov function is selected as,

$$V_s = \delta \mathbf{w}_s^T \delta \mathbf{w}_s \quad (\text{Eqn 7.3})$$

with the goal of proving

$$V_s \geq V_{s+1} \quad \forall \quad s. \quad (\text{Eqn 7.4})$$

The standard CMAC update law is applied to the weight vector, such that

$$\delta \mathbf{w}_{s+1} - \delta \mathbf{w}_s = -\mathbf{c}_s^T \left( \frac{\alpha e_s}{C} \right) = -\mathbf{c}_s^T \left( \frac{\alpha \mathbf{c}_s^T \delta \mathbf{w}_s}{C} \right) \quad (\text{Eqn 7.5})$$

substituting in for  $V_{s+1}$  gives the following simplified result:

$$V_{s+1} = V_s + (\alpha^2 - 2\alpha) \frac{e_s^2}{C}, \quad (\text{Eqn 7.6})$$

since  $e_s^2$  and  $C$  are strictly positive, then if  $\alpha^2 - 2\alpha$  is less than zero the system converges. The conditions for  $\alpha$  that meet this criteria are  $0 < \alpha < 2$ . As long as the constraint for  $\alpha$  is met, all input states are repeatedly visited and the system is also under continuous excitation, the learning does not diverge by the second Lyapunov method. Again, the above proof is directly lifted from Campanga [5]. It is only presented as the framework for the proofs in the following sections on the WCMAC learning convergence.



### 7.2.1 WCMAC stability

The second method of Lyapunov is now applied to the WCMAC model. The basic framework of a reference plant and training network is once again applied. The output formulation for WCMAC is slightly different than the traditional CMAC and is given as

$$y_s = \frac{\mathbf{c}_s \mathbf{w}_s}{\|\mathbf{c}_s\|}. \quad (\text{Eqn 7.7})$$

The supporting weight update algorithm for WCMAC is

$$\Delta \mathbf{w}_s = \alpha \frac{\mathbf{c}_s^T}{\|\mathbf{c}_s\|} (\hat{y}_s - y_s) \quad (\text{Eqn 7.8})$$

where  $\hat{y}_s$  is the response of the reference plant. The error function is defined as

$$e_s = (\hat{y}_s - y_s) = \frac{\mathbf{c}_s \hat{\mathbf{w}}_s}{\|\mathbf{c}_s\|} - \frac{\mathbf{c}_s \mathbf{w}_s}{\|\mathbf{c}_s\|} = \frac{\mathbf{c}_s \delta \mathbf{w}_s}{\|\mathbf{c}_s\|}. \quad (\text{Eqn 7.9})$$

The Lyapunov function is again chosen as square of the difference between the weight vectors,

$$V_s = \delta \mathbf{w}_s^T \delta \mathbf{w}_s \quad (\text{Eqn 7.10})$$

with same goal of proving

$$V_s \geq V_{s+1} \quad \forall \quad s, \quad (\text{Eqn 7.11})$$

which can be expressed directly in terms of the difference in weight vectors between the reference plant and the WCMAC network,

$$\delta \mathbf{w}_s^T \delta \mathbf{w}_s \geq \delta \mathbf{w}_{s+1}^T \delta \mathbf{w}_{s+1} \quad \forall \quad s. \quad (\text{Eqn 7.12})$$

The weight update algorithm for WCMAC,

$$\delta \mathbf{w}_{s+1} = \delta \mathbf{w}_s - \alpha e_s \frac{\mathbf{c}_s^T}{\|\mathbf{c}_s\|}, \quad (\text{Eqn 7.13})$$

is substituted for  $\delta \mathbf{w}_{s+1}$  and solved.

$$\delta \mathbf{w}_s^T \delta \mathbf{w}_s \geq \delta \mathbf{w}_{s+1}^T \delta \mathbf{w}_{s+1}, \quad (\text{Eqn 7.14})$$

$$\geq \left[ \delta \mathbf{w}_s - \alpha e_s \frac{\mathbf{c}_s^T}{\|\mathbf{c}_s\|} \right]^T \left[ \delta \mathbf{w}_s - \alpha e_s \frac{\mathbf{c}_s^T}{\|\mathbf{c}_s\|} \right], \quad (\text{Eqn 7.15})$$

$$\geq \delta \mathbf{w}_s^T \delta \mathbf{w}_s - \left( \alpha e_s \frac{\delta \mathbf{w}_s^T \mathbf{c}_s^T}{\|\mathbf{c}_s\|} \right) - \left( \alpha e_s \frac{\mathbf{c}_s \delta \mathbf{w}_s}{\|\mathbf{c}_s\|} \right) + \left( \alpha^2 e_s^2 \frac{\mathbf{c}_s \mathbf{c}_s^T}{\|\mathbf{c}_s\|^2} \right), \quad (\text{Eqn 7.16})$$

$$\geq \delta \mathbf{w}_s^T \delta \mathbf{w}_s - \left( 2\alpha \frac{e_s^2}{\|\mathbf{c}_s\|} \right) + \left( \alpha^2 \frac{e_s^2}{\|\mathbf{c}_s\|} \right), \quad (\text{Eqn 7.17})$$

$$\geq \delta \mathbf{w}_s^T \delta \mathbf{w}_s - (2\alpha - \alpha^2) \frac{e_s^2}{\|\mathbf{c}_s\|}, \quad (\text{Eqn 7.18})$$

This expression simplifies to

$$V_s \geq V_{s+1} - (2\alpha - \alpha^2) \frac{e_s^2}{\|\mathbf{c}_s\|} \quad (\text{Eqn 7.19})$$

where  $e_s^2$  and  $\|\mathbf{c}_s\|$  are both strictly positive. The conditions for  $\alpha$ , such that WCMAC does not diverge by the second method of Lyapunov, are  $0 < \alpha < 2$  with the same criteria that applied to the original CMAC proof. It should also be noted that this solution applies to the original CMAC, the variable receptive field strength CMAC and the WCMAC models. This property can be proved by simple substitution of the state activation vector,  $\mathbf{c}_s$ , for the different network models.

### 7.2.2 Reduced Computation WCMAC learning

Throughout this dissertation, the concept of minimizing the amount of computation is continually stressed. The most significant computation added to the WCMAC over the variable strength receptive field CMAC models is the square root function performed in finding the norm of the receptive field strength vector,  $\mathbf{c}_s$ . In the case of a WCMAC of any dimension without decimation, the norm of the receptive field strength vector is constant. When decimation is included, the receptive field strength varies depending on how close the input state is to the nearest weight.

The reduced computational model is derived by simply substituting in a scalar,  $A$  for the norm in the output generation equation

$$y_s = \frac{\mathbf{c}_s \mathbf{w}_s}{A} \quad (\text{Eqn 7.20})$$

and the weight update algorithm

$$\Delta \mathbf{w}_s = \alpha \frac{\mathbf{c}_s^T}{A} (\hat{y}_s - y_s). \quad (\text{Eqn 7.21})$$

The proof of stability by Lyapunov's second method is applied again using the same error functions, conditions and Lyapunov function. The basic algebraic derivation is show below.

$$\delta \mathbf{w}_s^T \delta \mathbf{w}_s \geq \delta \mathbf{w}_{s+1}^T \delta \mathbf{w}_{s+1}, \quad (\text{Eqn 7.22})$$

$$\geq \left[ \delta \mathbf{w}_s - \alpha e_s \frac{\mathbf{c}_s^T}{A} \right]^T \left[ \delta \mathbf{w}_s - \alpha e_s \frac{\mathbf{c}_s^T}{A} \right], \quad (\text{Eqn 7.23})$$

$$\geq \delta \mathbf{w}_s^T \delta \mathbf{w}_s - \left( \alpha e_s \frac{\delta \mathbf{w}_s^T \mathbf{c}_s^T}{A} \right) - \left( \alpha e_s \frac{\mathbf{c}_s \delta \mathbf{w}_s}{A} \right) + \left( \alpha^2 e_s^2 \frac{\mathbf{c}_s \mathbf{c}_s^T}{A^2} \right), \quad (\text{Eqn 7.24})$$

$$\geq \delta \mathbf{w}_s^T \delta \mathbf{w}_s - (2\alpha e_s^2) + \left( \alpha^2 e_s^2 \frac{\mathbf{c}_s \mathbf{c}_s^T}{A^2} \right), \quad (\text{Eqn 7.25})$$

$$\geq \delta \mathbf{w}_s^T \delta \mathbf{w}_s - \left( 2\alpha + \alpha^2 \frac{\mathbf{c}_s \mathbf{c}_s^T}{A^2} \right) e_s^2. \quad (\text{Eqn 7.26})$$

Finally, the boundary conditions for the learning to maintain the open-loop Lyapunov stability are

$$0 \leq \alpha \leq \frac{2A^2}{\mathbf{c}_s \mathbf{c}_s^T}. \quad (\text{Eqn 7.27})$$

Another interpretation of the above results is that the effective learning rate of the WCMAC when using the reduced computation model, will vary inversely with the ratio of the squared approximation,  $A$ , to the inner product of the receptive field strength vector with itself, or the magnitude of  $\mathbf{c}_s$  squared. If the scalar approximation,  $A$ , is set near the average of the norm of the receptive field strength for all possible excitation patterns, then the effective learning rate over a large number of samples will approach the full computation model. There is an expected variation on the sample-to-sample basis. This is explored in the following convergence section.

## 7.3 Convergence

### 7.3.1 Reduced Computational Model vs. WCMAC Model

The reduced computational model was inspired from some previous CMAC implementations in hardware done by the author. The traditional CMAC architecture is easily implemented in standard electronics components, such as Field Programmable Gate Arrays (FPGA) and standard memory devices. If the generalization widths are limited to only powers of two, the division operation used during the output generation and weight update algorithm becomes a simple bit shift.

The norm of the receptive field state vector is used in the output formulation and the weight update algorithm of the WCMAC. The calculation of a norm involves squaring individual values, accumulating the squared values and finding the square root of the accumulated sum. Each of these operations, except the accumulation operation, are difficult to implement in hardware and are also time consuming operations in software. Some implementations might trade-off the minimal error and high rate of convergence with a simpler implementation that can process more sample sets per second.

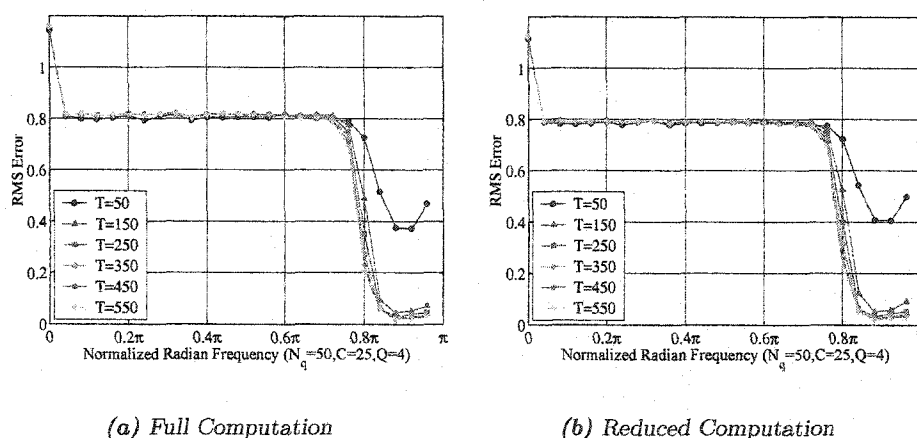
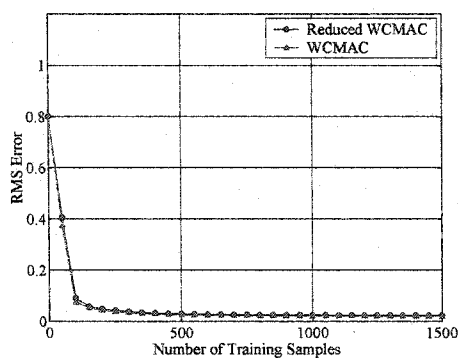


Figure 7-2: Full Computation WCMAC vs. Reduced Computational Model

Figures 7-2(a) and 7-2(b) compare the region of convergence for the reduced computation network and full computational model. This data was gathered from the same experiment that was performed in each of the previous chapters. Again, the experiment is essentially a Fourier analysis of the network configuration, by attempting to learn a series of harmonics. The network was designed to act as a highpass filter, a function that the traditional CMAC would not be able to learn. For this simple one-dimensional model, the reduced computation model has a slightly slower rate of convergence and a barely noticeable difference in RMS error. Figure 7-3 show the sample-to-sample convergence of the two networks for the harmonic at  $0.9\pi$ .



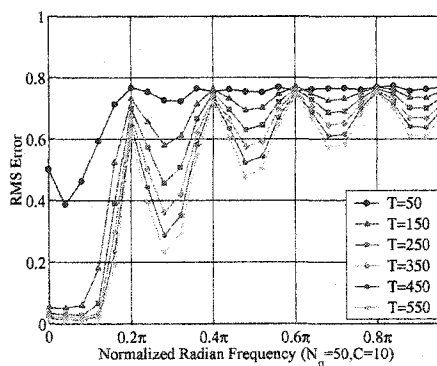
*Figure 7-3: Sample by Sample Convergence*

It is quite evident from this plot that the convergence rate and final error value are extremely close for both networks. This network simplification highly reduces the computation during both output generation and weight updating and coupling, this result, with the previous analysis of the stability, gives an extremely powerful implementation for systems with highly strained resources.

### 7.3.2 Previous CMAC Models vs. WCMAC Model

It is difficult to compare the rate of convergence for the WCMAC to previous CMAC models, since the WCMAC will learn functions that the other models are not capable of learning. The rate of convergence is also related to the bandwidth of the network and the chosen generalization values. A brief experiment is designed here to compare the convergence of three networks: WCMAC, Albus CMAC and the linear tapered receptive field CMAC. The three networks are normalized by the approximate bandwidth of the model and not simply the generalization or learning coefficient. In other words, the three networks are limited to learning a similar class of functions.

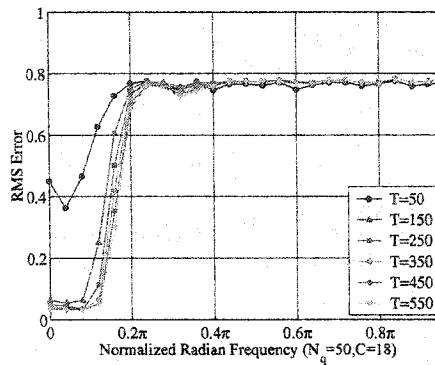
Initially, an Albus CMAC model is designed with a learning coefficient of 0.5 and generalization of 10 for one-dimension. This network was then applied to the Fourier analysis that has already been used many times. The results show a network with its first null at  $0.2\pi$  or a passband from 0 to  $0.2\pi$ . The spectral convergence of this model is shown in figure 7-4.



**Figure 7-4:** Albus Receptive Field Convergence

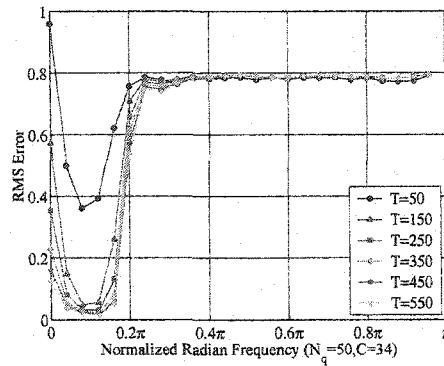
The linear tapered receptive field model is then designed to approximate the bandwidth

of the Albus receptive field network. In order to match the same bandwidth, the generalization had to be increased to a width of 18. This performance of this network is shown in figure 7-5.



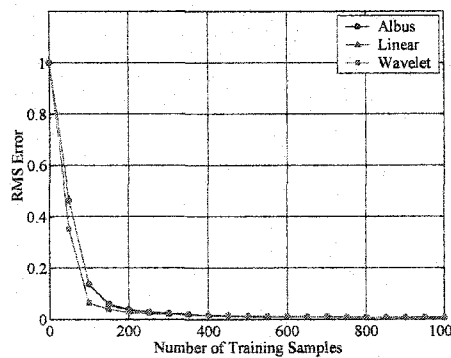
**Figure 7-5:** Linear Tapered Receptive Field Convergence

Finally, the WCMAC model is designed with the same criteria and the generalization is increased to 34, see figure 7-6. It is clear that the high-order receptive field functions can generalize to a wider range and still provide the same bandwidth. This is actually a very important property when CMAC is used in real-time problems. Using very small generalization values means a smaller number of weights are accessed for each sample. In some problems, a few weights may be accessed commonly, while other weights are only infrequently accessed. The difference in magnitude of these weights has a tendency to drift apart, due to this uneven sampling. The resulting learned function may have nonlinear steps or a poor derivative approximation. Weight smoothing or weight normalization can be used to minimize this effect. The WCMAC minimizes this effect by using large generalization values for the same bandwidth. In the case of this particular problem, the WCMAC accesses about 70 percent of the total weights during a training cycle, while the traditional CMAC only accesses 20 percent of the weights.



**Figure 7-6:** Wavelet Receptive Field Convergence

In terms of convergence rate on a sample-to-sample basis for these three networks, the WCMAC is clearly the fastest for this one example, but it should be noted that this is not the major advantage of the WCMAC, figure 7-7. The ability to learn frequency ranges that were beyond the scope of previous CMAC networks and the ability to widely generalize at any frequency are the true advantages of the WCMAC model.



**Figure 7-7:** Wavelet Receptive Field Convergence



## 7.4 Discussion

This chapter proved that the open-loop stability of the WCMAC is bound by the same range of values for the learning coefficient as the traditional CMAC model. This proof also directly applies to the linear taper receptive field models. A new learning algorithm that trades off computation time for speed of convergence and accuracy is presented and included in the stability proof. This new algorithm is ideal for systems limited by the rate at which samples can be computed or have other resource limitations. Finally, some simple examples on the rate of convergence were included where the networks are normalized to bandwidth and not generalization width.

# CHAPTER 8

## CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

### 8.1 Summary

Through the years of CMAC development and analysis, there has continually been confusion about the network's learning ability or bandwidth, mostly due to the analysis of CMAC with a single input dimension. The one-dimensional CMAC model problem is a unique case, in that no decimation exists between the state-space and the weight-space. The multidimensional case, however, uses decimation that scales as a power of the number of input states. Consequently, decimation is the dominant factor in limiting the ability to learn multidimensional problems. For one-dimensional problems, the dominant factors are the nulls in the frequency response of the receptive field and the overall frequency response of the receptive field. A variety of CMAC implementations were examined to understand the bandwidth of the traditional Albus CMAC with binary receptive fields, linear receptive models and finally, spline-based receptive field networks. By removing the decimation component in the weight-space and using a single impulse in the weight-space, the frequency response for the output stage of the network is found by sweeping

the input states and taking the Fourier transform of the output. This was done for both one-dimensional and multidimensional cases.

The spectrum of the output is then adjusted to include the decimation factor and separated into both primary frequency band components and replica bands, which by definition are always present in the sampled systems. The decimation factor is a matrix that defines the linearly independent set of vectors used to build the sampling lattice. The primary bands and replica bands are compared for overlapping areas of support which cause destructive aliasing. The information in the region of overlap is corrupted and cannot be fully recovered. This multidimensional Nyquist sampling theory analysis emphasizes the coupling of information storage and network bandwidth. In other words, the sampling lattice and receptive field shape have to be designed in concert to maximize bandwidth and minimize weight computation.

Another outcome of this analysis is the clear demonstration that CMAC is a lowpass-only adaptive structure, with minimal or no ability to generalize higher frequency components. Furthermore, it is also evident that designing multidimensional lattice structures and the supporting receptive fields are extremely difficult above two dimensions. Both of these problems have been solved by image processing and multidimensional signal processing techniques. The Discrete Fourier Transform and the Discrete Wavelet Transform essentially generalize widely across the entire frequency domain. These transforms treat each dimension separately which results in a rectangular lattice structure for sampling.

The derivation of the Wavelet based CMAC was developed on the principles of the Discrete Wavelet Transform. The wavelet provides the compact support that is necessary to bandlimit the response on the network and allow for the decimation, which provides the computational advantage. The ability to modulate the frequency responses of the Wavelet and move the passband of the network to any frequency range makes the WCMAC an adaptive structure that is not limited to only low-frequency content. The ability to widely generalize is also not limited to the lowpass region. In fact, the generalization only limits

the width of the passband, but it can be modulated to any frequency. This is analogous to the Fourier transform, which generalizes across the entire input domain with sine waves at each frequency point.

The computational advantage is maintained in the network by controlling the decimation in such a manner that the folding frequencies, which are defined by the decimation values, are not within a passband of the network's response. By designing each dimension individually and choosing both the wavelet receptive field parameters and decimation values, such that the passband of the wavelet and the folding frequencies do not intersect, but are essentially bound to each other, the computation will be optimized. Each dimension is designed separately and placed into the full network using a rectangular lattice structure. As long as there does not exist aliasing in the design of any single dimension, the overall network will also be void of aliasing.

It should be noted that the rectangular lattice is not the optimal lattice for all problems, but the advantages of ease in design and fast computation outweigh its disadvantages. Especially when considered in the application of neural networks where the purpose is to learn functions that are difficult to model, are unpredictable or where there is minimal up-front knowledge. For these problems, designing the optimal static sampling lattice is essential impossible. The disadvantages of the rectangular sampling lattice can typically be overcome by simply increasing the sampling density, which will increase the computation time. The other approach is an adaptive lattice structure, which in many cases severely increases computation time by removing the modular, fast-search algorithm for finding the receptive field centers. Once again, an increase in the sampling lattice density might be the more appropriate choice.

Another advantage to designing a separate receptive field function along each of the dimensions is the ability to support different bandwidths, generalization widths and frequency ranges along each input state. This can be helpful when each state is controlled by a device with a different dynamic range. It was also demonstrated that sets of networks can

be combined in a simple algebraic manner to learn complex functions and simultaneously decompose the spectral content of the function being learned.

The Lyapunov stability proof, that was developed for the Albus CMAC, was reformulated and applied to the WCMAC. The results demonstrated that the WCMAC has the same stability boundary, with respect to the learning rate, as the traditional CMAC. This simple proof extension also covers many of the variable strength receptive field CMAC models. A reduced computation learning algorithm is also included and its convergence rate is compared with the basic learning algorithm.

## 8.2 Conclusions

The application of basic information theory, Nyquist sampling theory and multidimensional signal processing techniques to the traditional CMAC revealed the primary limitations and dynamics of the CMAC model. The insight gained led to the development of the Wavelet based CMAC which greatly enhanced the learning capabilities over the traditional CMAC. Entire new classes of functions and frequency ranges can now be handled with WCMAC. The knowledge gained from analyzing the bandwidth of the traditional CMAC clarifies the necessary balance that must exist between the bandwidth of the receptive field and the weight structure, where the structure involves both placement and decimation values.

There have previously been a variety of lattice structures developed for CMAC with two main caveats. In the case of static receptive fields, they become more difficult to design as the dimensionality increases. Adaptive lattice structures typically give rise to severe increases in computation time. A simple and straight forward approach of using rectangular lattice implementations allows multidimensional problems to be divided into a series of one-dimensional problems. This simplifies the design process of balancing weight count and receptive field bandwidth. It also helps remove the ambiguous relationship between generalization and weight decimation.

The resulting WCMAC model has many properties akin to popular discrete time transforms, including the Fast Fourier Transform and the Discrete Wavelet Transform, in that bandwidth and information extraction is balanced by computation. Furthermore, like these transforms, the WCMAC is not limited to any particular frequency range. All these enhancements were made to the network without compromising the two components that make CMAC such a powerful tool: extremely fast computation and rapid learning convergence.

### 8.3 Suggestions for Future Work

The most common outcome of all research endeavors is the generation of more unanswered questions and different directions to explore. This research is no different. The goal of this research was to further understand the basic dynamics of CMAC and extend its capabilities with that knowledge gained. The research was also strongly influenced by the secondary objectives of minimizing computation and reducing the complexity of the overall network design. However, significant work still exists in the formalization of the WCMAC for the balance of optimal computation and bandwidth.

The next obvious undertaking is the development of a formalized approach to the design of the receptive fields and the decimation, that would devise orthogonal wavelet functions with the appropriate generalization width and decimation factor for a user-defined set of parameters, such as a high-pass frequency, a low-pass frequency and the number of input states. In other words, automate the process of designing receptive field functions and decimation values; this process was visually done during this research. The resulting network would have optimal weight storage for the particular a bandwidth, which also gives rise to minimal computation. There is also an unlimited number of wavelet functions which might be more appropriate.

This dissertation used examples of target functions where most of the frequency components were already known, resulting in straightforward designs of WCMAC networks to

learn the function. Neural networks are commonly used in situations where limited a priori knowledge exists; therefore, an on-line adaptation of the network with respect to its effective frequency range could also be advantageous. It may also be designed in such a manner that it used discrete passbands, such that the folding frequency never entered a significant region of the passband. This would facilitate a static lattice structure and minimize the computational requirements.

The rectangular lattice structure proposed in this research is recognized as having certain limitations, but it facilitates the design of a multidimensional network and has certain computational advantages. An investigation into the inappropriate uses of the rectangular lattice would be prudent for the WCMAC. This research has already been performed for the traditional CMAC. However, the WCMAC is more sensitive to the placement of folding frequencies than the traditional CMAC, since it is not limited to the low-pass region. This would not be a simple rehash of previous work, but a new investigation that facilitates WCMAC network designs with a variety of lattice structures.

Finally, there are a series of Hierarchical CMAC models that were developed to increase the class of functions that CMAC is capable of learning. An interesting study would compare the hierarchical CMAC capabilities to the WCMAC. This could also be followed by a study of the WCMAC capabilities in similar hierarchical configuration.

# Bibliography

- [1] J. Hertz, A. Krogh, and R. Palmer, *Introduction To The Theory Of Neural Computation*. Reading, NH: Addison-Wesley, 1991.
- [2] P. Churchland, *Neurophilosophy: Toward a Unified Science of the Mind/Brain*. Cambridge, MA: MIT Press, 1986.
- [3] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ: Prentice Hall, 1994.
- [4] I. Aleksander and H. Morton, *A Introduction to Neural Computing*. London, UK: Chapman & Hall, 1990.
- [5] D. P. Campagna, *Stability and Weight Smoothing in CMAC Neural Networks*. PhD thesis, University of New Hampshire, Durham, NH, May 1998.
- [6] W. McCulloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.
- [7] D. O. Hebb, *The Organization of Behavior*. New York: Wiley & Sons, 1949.
- [8] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.
- [9] F. Rosenblatt, *Principles of Neurodynamics*. Washington D.C.: Spartan Books, 1962.
- [10] B. Widrow, "Adaptive sampled-data systems - a statistical theory of adaptation," *1959 IRE WESCON Convention Record*, vol. 4, pp. 74–85, 1959.
- [11] B. Widrow and M. Hoff, "Adaptive switching circuits," in *1960 IRE WESCON Convention Record*, vol. 4, pp. 96–104, New York: IRE, 1960.
- [12] B. Widrow, "Generalization and information storage in networks of adaline "neurons",," in *Self-Organizing Systems 1962* (M. Yovits, G. Jacobi, and G. Goldstein, eds.), (Washington), pp. 435–461, (Chicago 1962), Spartan, 1962.
- [13] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969.
- [14] T. Kohonen, "An adaptive associative memory principle," *IEEE Transactions on Electronic Computers*, vol. C-23, pp. 444–445, 1974.
- [15] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.
- [16] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by backpropagating errors," *Nature*, vol. 323, pp. 533–536, 1986.



- [17] S. Grossberg, "How does the brain build a cognitive code?," *Psychological Review*, vol. 87, 1980.
- [18] J. Hopfield, D. Feinstein, and R. Palmer, "'Unlearning' has a stabilizing effect in collective memories," *Nature*, vol. 304, pp. 158-159, 1983.
- [19] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321-355, 1988.
- [20] J. Albus, "The cerebellar model articulation controller," *Trans. ASME*, vol. 97, no. 3, 1975.
- [21] J. Albus, "A new approach to manipulator control: the cerebellar model articulation controller," *Trans. ASME, J. Dynamic Syst. Meas. Contr.*, vol. 97, pp. 220-227, September 1975.
- [22] J. Albus, "A theory of cerebellar functions," *Mathematical Biosciences*, vol. 10, pp. 25-61, 1971.
- [23] J. S. Albus, *Theoretical and Experimental Aspects of a Cerebellar Model*. PhD thesis, University of Maryland, College Park, MD, 1972.
- [24] J. Albus, "Data storage in the cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control, Transactions of ASME*, vol. 97, pp. 228-233, September 1975.
- [25] J. Albus, "A model of the brain for robot control part3: A comparison of the brain and our model," *Byte Magazine*, 1979.
- [26] J. Albus, "Mechanisms of planning and problem solving in the brain," *Mathematical Biosciences*, vol. 45, pp. 247-293, 1979.
- [27] J. S. Albus, *Brains, Behavior and Robotics*. Peterborough, NH: BYTE Publications Inc., 1981.
- [28] W. T. Miller, F. H. Glanz, and L. G. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," *International Journal of Robotics Research*, vol. 6, no. 2, pp. 84-98, 1987.
- [29] W. T. Miller, F. H. Glanz, and L. G. Kraft, "CMAC: An associative neural network alternative to backpropagation," *Proceedings of the IEEE, Special Issue on Neural Networks*, vol. 78, pp. 1561-1567, October 1990.
- [30] L. G. Kraft and D. P. Campagna, "A comparison of CMAC neural network and traditional adaptive control systems," in *Proc. of the 1989 American Controls Conf.*, (Pittsburgh, PA), May 1989.
- [31] L. G. Kraft and D. P. Campagna, "A comparison between CMAC neural network control and two traditional control systems," *IEEE Control Systems Mag.*, pp. 36-43, Apr. 1990.
- [32] L. G. Kraft, W. T. Miller, and D. Dietz, "Development and application of cmac neural network-based control," in *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches* (D. A. White and D. A. Sofge, eds.), (New York, NY), Van Nostrand-Reinhold, 1992.

- [33] W. T. Miller, B. A. Box, E. C. Whitney, and J. M. Glynn, "Design and implementation of a high speed CMAC neural network using programmable CMOS logic cell arrays," in *Advances in Neural Information Processing Systems 3* (L. R. P. M. J. E. and T. D. S., eds.), (San Mateo, CA), pp. 1022–1027, Morgan Kaufmann, 1991.
- [34] J. S. Ker, Y. H. Kuo, R. C. Wen, and B. D. Liu, "Hardware implementation of CMAC neural network with reduced storage requirement," *IEEE Transactions on Neural Networks*, vol. 8, pp. 1545–1556, Nov. 1997.
- [35] B. A. Box, "Hardware implementation of a CMAC neural network using field programmable gate arrays," Master's thesis, University of New Hampshire, Durham, NH, 1990.
- [36] B. Yang, "A VLSI implementation of the CMAC neural network," Master's thesis, University of New Hampshire, Durham, NH, May 1992.
- [37] M. J. Carter, F. Rudolph, and A. Nucci, "Operational fault tolerance of CMAC networks," *NIPS-90, Denver*, 1990.
- [38] M. J. Carter, A. Nucci, W. Miller, E. An, and F. Rudolph, "Slow learning in cmac networks and implications for fault tolerance," *UNH Intelligent Structures Group Technical Report*, pp. 1–6, July 1990.
- [39] F. Glanz, W. Miller, and L. Kraft, "An overview of the CMAC neural network," in *IEEE Conference on Neural Networks for Ocean Engineering*, (Washington, DC), pp. 301–308, 1991.
- [40] W. T. Miller, "Real-time neural network control of a biped walking robot," *Control Systems Magazine*, pp. 41–48, February 1994.
- [41] W. T. Miller, P. J. Latham, and S. M. Scalera, "Bipedal gait adaptation for walking with dynamic balance," in *Proceedings of the 1991 American Controls Conference*, vol. 2, (Boston, MA), pp. 1603–1608, June 1991.
- [42] W. T. Miller, "A nonlinear learning controller for robotic manipulators," *Proc of the SPIE: Intelligent Robots and Computer Vision*, vol. 726, pp. 416–423, 1986.
- [43] W. T. Miller, "Real time application of neural networks for sensor-based control of robots with vision," *IEEE Transactions on Systems, Man, and Cybernetics, Special Issue on Information Technology for Sensory-Based Robot Manipulators*, vol. 19, pp. 825–831, July/August 1989.
- [44] W. T. Miller, R. P. Hewes, F. H. Glanz, and L. G. Kraft, "Real time dynamic control of an industrial manipulator using a neural network based learning controller," *IEEE Journal of Robotics and Automation*, vol. 6, pp. 1–9, 1990.
- [45] W. T. Miller, "Real-time neural network control of a biped walking robot," *IEEE Transactions on Automatic Control*, 1993.
- [46] W. T. Miller, "Sensor based control of robotic manipulators using a general learning algorithm," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 157–165, 1987b.
- [47] W. T. Miller, F. H. Glanz, and L. G. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," *International Journal of Robotics Research*, vol. 6, no. 3, pp. 84–98, 1987.

- [48] J. Woodward, "Application of CMAC neural network learning to a stepping quadruped robot," Master's thesis, University of New Hampshire, Durham, NH, December 1996.
- [49] B. Buchika, "Application of CMAC neural networks to a quadruped robot walking with a trotting gait," Master's thesis, University of New Hampshire, Durham, NH, May 1996.
- [50] A. L. Kun, *A Sensory-Based Adaptive Walking Control Algorithm for Variable Speed Biped Robot Gaits*. PhD thesis, University of New Hampshire, Durham, NH, May 1997.
- [51] F. H. Glanz and W. T. Miller, "Deconvolution and nonlinear inverse filtering using a neural network," in *International Conference on Acoustics and Signal Processing*, vol. 4, (Glasgow, Scotland), pp. 2349–2352, 1989.
- [52] F. H. Glanz and W. T. Miller, "Deconvolution and nonlinear inverse filtering using a neural network," in *First Annual Conference of the International Neural Network Society*, vol. 4, (Boston, MA), p. 440, September 1988.
- [53] K. F. Arehart, "A CMAC-based cursive handwriting recognizer for the windows for pen computing operating environment," Master's thesis, University of New Hampshire, Durham, NH, May 1994.
- [54] L. G. Kraft and J. Pallota, "Vibration control using CMAC neural network with optimized weight smoothing," in *Proceedings of the American Control Conference*, vol. 2, (San Diego, CA), pp. 1181–1185, 1999.
- [55] L. G. Kraft and J. Pallota, "Real-time vibration control using CMAC neural network with weight smoothing," in *Proceedings of the American Control Conference*, vol. 6, (Chicago, IL), pp. 3939–3943, 2000.
- [56] J. Pallota, "Vibration control using weight smoothing CMAC neural network," Master's thesis, University of New Hampshire, Durham, NH, May 1999.
- [57] J.-N. Lin, S.-M. Song, J.-N. Lin, and S.-M. Song, "Modeling gait transitions of quadrupeds and their generalization with CMAC neural networks," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 32, pp. 177–189, Aug. 2002.
- [58] Y. Kim and F. Lewis, "Optimal design of CMAC neural-network controller for robot manipulators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 30, pp. 22–31, Aug. 2000.
- [59] T. Tao, H.-C. Lu, and T.-H. Hung, "The CA-CMAC for downsampling image data size in the compressive domain," *2002 IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, pp. 555–560, 2002.
- [60] Y. Iiguni, "Hierarchical image coding via cerebellar model arithmetic computers," *IEEE Transactions on Image Processing*, vol. 5, pp. 1393–1401, Oct. 1996.
- [61] C. Shang, D. Reay, and B. Williams, "Adapting CMAC neural networks with constrained lms algorithm for efficient torque ripple reduction in switched reluctance motors," *IEEE Transactions on Control Systems Technology*, vol. 7, pp. 401–413, July 1999.
- [62] H. Shiraishi, S. Ipri, and D.-I. Cho, "CMAC neural network controller for fuel-injection systems," *IEEE Transactions on Control Systems Technology*, vol. 3, pp. 32–38, Mar. 1995.

- [63] R. Kurozumi, S. Fujisawa, T. Yamamoto, and Y. Suita, "Development of an automatic travel system for electric wheelchairs using reinforcement learning systems and CMACs," in *Proceedings of the 2002 International Joint Conference on Neural Networks*, pp. 1690–1695, 2002.
- [64] S. H. Lane, D. A. Handelman, and J. J. Gelfand, "Theory and development of higher order CMAC neural networks," *IEEE Control Systems Mag.*, pp. 23–30, Apr. 1992.
- [65] P.-C. E. An, *An Improved Multidimensional CMAC Neural Network: Receptive Field Function and Placement*. PhD thesis, University of New Hampshire, Durham, NH, September 1991.
- [66] P. E. An, W. T. Miller, and P. C. Parks, "Design improvements in associative memories for cerebellar model articulation controllers (CMAC).," in *Proc. Int. Conf. on Artificial Neural Networks, Helsinki*, pp. 1207–1210, North-Holland, 1991. Vol. 2.
- [67] W. S. Mischo, "Receptive fields for CMAC. an efficient approach," in *Artificial Neural Networks* (I. Aleksander and J. Taylor, eds.), vol. 2, pp. 595–598, Technische Hochschule Darmstadt, Elsevier Science Publishers B.V., 1992.
- [68] C.-M. Chen and C.-M. Hong, "A weighted grey CMAC neural network with output differentiability," *Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, vol. 2, pp. 1009–1014, July 2001.
- [69] G. Horvath and T. Szabo, "CMAC neural network with improved generalization property for system modeling," in *IEEE Instrumentation and Measurement Technology Conference*, (Anchorage, AK), pp. 1603–1608, May 2002.
- [70] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.
- [71] P. C. Parks and J. Militzer, "Improved allocation of weights for associative memory storage in learning control systems," in *IFAC Design Methods of Control Systems*, (Zurich, Switzerland), pp. 507–512, 1991.
- [72] Y. Jia, "Use of the sphere packing lattice to investigate the quality of CMAC receptive field placement," Master's thesis, University of New Hampshire, Durham, NH, May 1994.
- [73] M. Miwa, T. Furuhashi, M. Matsuzaki, and S. Okuma, "CMAC Modeling using psuedo-bacterial genetic algorithm and its acceleration," *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 1, pp. 250–254, October 2001.
- [74] F. J. Gonzalez-Serrano, A. R. Figueiras-Vidal, and A. Artes-Rodriguez, "Generalizing CMAC architecture and training," *IEEE-NN*, vol. 9, pp. 1509–1514, Nov. 1998.
- [75] S. L. Hung and C. S. Lin, "MS-CMAC neural-network learning model in structural engineering," *Journal of Computing in Civil Engineering*, vol. 13, no. 1, pp. 1–11, 1999.
- [76] S. L. Hung and C. S. Lin, "High-order MS-CMAC neural network," *IEEE Transactions on Neural Networks*, vol. 12, pp. 598–603, May 2001.
- [77] H.-M. Lee, C.-M. Chen, and Y.-F. Lu, "A self-organizing HCMAC neural network classifier," *Proceedings of International Joint Conference on Neural Networks*, vol. 3, pp. 1960–1965, July 2001.

- [78] D. Ellison, "On the convergence of the Albus Perceptron," *IMA Journal of Math. Control and Info.*, vol. 5, pp. 315–331, 1988.
- [79] P. C. Parks and J. Militzer, "Convergence properties of associative memory storage for learning control system," *Automation and Remote Control*, vol. 50, pp. 254–286, 1989.
- [80] P. C. Parks and J. Militzer, "A comparison of five algorithms for the training of CMAC memories for learning control systems," *Automatica*, vol. 28, no. 5, pp. 1027–1035, 1992.
- [81] D. Ellison, "On the convergence of the multidimensional Albus Perceptron," *International Journal of Robotics Research*, vol. 10, no. 5, pp. 338–357, 1991.
- [82] Y. Wong and A. Sideris, "Learning convergence in the cerebellar model articulation controller," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 115–121, 1992.
- [83] M. Brown and C. J. Harris, "Comments on "learning convergence in the cerebellar model articulation controller"," *IEEE Trans. on Neural Networks*, vol. 6, no. 4, pp. 1016–1018, 1995.
- [84] M. Brown, C. J. Harris, and P. C. Parks, "The interpolation capabilities of the binary CMAC," *Neural Networks*, vol. 16, pp. 429–440, 1994.
- [85] M. Brown and C. J. Harris, *Neurofuzzy Adaptive Modelling and Control*. Hemel Hempstead, UK: Prentice Hall, 1994.
- [86] M. Brown, *Neurofuzzy Adaptive Modelling and Control*. PhD thesis, Southampton University, 1993.
- [87] Y. Wong, "CMAC learning is governed by a single parameter," in *IEEE Conference on Neural Networks*, (San Fransico, CA), pp. 1439–1443, 1993.
- [88] F. J. Gonzalez-Serrano, A. R. Figueiras-Vidal, and A. Artes-Rodriguez, "Fourier analysis of the generalized CMAC neural network," *Neural Networks*, vol. 11, no. 3, pp. 391–396, 1998.
- [89] C. S. Lin and C. T. Chiang, "Learning convergence of CMAC technique," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1281–1292, 1997.
- [90] G. Kraft and K. Liu, "Stability of CMAC neural network on closed-loop vibration control systems," in *IASTED Controls and Applications Conference*, (Cancun, Mexico), 2000.
- [91] H. Liu and D. Wang, "Convergence of CMAC network learning control for a class of nonlinear dynamic systems," in *Proceedings of the 1999 IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, (Cambridge, MA), pp. 108–113, 1999.
- [92] M. Vetterli, "Wavelets, approximation and compression," *IEEE Signal Processing Magazine*, pp. 59–73, September 2001.
- [93] I. Daubechies, "Orthonormal bases of compactly supported wavelets," *Communication of Pure Applied Mathematics*, vol. 41, pp. 909–996, November 1988.
- [94] S. Mallat, "A theory of multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Recognition Machine Intell*, vol. 11, pp. 674–693, July 1989.

- [95] K. R. Castleman, *Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
- [96] P. Cristea, R. Tuduce, and A. Cristea, "Time series prediction with wavelet neural networks," *Neural Network Applications in Electrical Engineering, 2000. NEUREL 2000.*, pp. 5–10, Sept. 2000.
- [97] R. Mersereau and T. Speake, "The processing of periodically sampled multidimensional signals," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-31, pp. 188–194, February 1983.
- [98] P. Murthy and E. Lee, "Multidimensional synchronous dataflow," *IEEE Transactions on Signal Processing*, vol. 50, pp. 2064–2079, July 2002.
- [99] E. Viscito and J. Allebach, "Design of perfect reconstruction multi-dimensional filter banks using cascaded smith form matrices," *ISCAS*, vol. 8, pp. 831–834, 1988.
- [100] J. Canfield, *Active Disturbance Cancellation in Nonlinear Dynamical Systems Using Neural Networks*. PhD thesis, University of New Hampshire, Durham, NH, December 2003.

# APPENDIX A

## HARDWARE DEVELOPMENT

### PAPER

The following appendix is the replication of a paper submitted to the IEEE Conference on Field Custom Computing Machines in the year 2000. The paper was selected for a short presentation and publication, but was withdrawn by the authors, due to their inability to attend the conference. It is reproduced here for documentation purposes and a record of the accomplished work.

# An FPGA Based CMAC Processor for High-Speed Vibration Control

Brian Kirk<sup>\*,+</sup>, Dr. L. G. Kraft<sup>+</sup>

<sup>+</sup>Electrical and Computer Engineering  
University of New Hampshire  
Durham, NH 03824  
[gordon.kraft@unh.com](mailto:gordon.kraft@unh.com)

<sup>\*</sup>Compaq Computer Corporation  
110 Spitbrook Road  
Nashua, NH 03062  
[brian.kirk@compaq.com](mailto:brian.kirk@compaq.com)

## Abstract

*A CMAC neural network has been designed and implemented on a reconfigurable computing platform. The major motivation behind the work is to extend the frequency range of ongoing vibration control research. A new CMAC processor is expected to increase the working frequency range by a factor of 10. The PCI Pamette reconfigurable computing platform is used as the development platform to minimize cost and time of the continually changing hardware. The network structure and the hardware implementation are discussed in detail. The performance of the new network is compared to past hardware and software implementations. Finally, the vibration control system is simulated to demonstrate the increased dynamic range of the controller.*

research has been conducted in the areas of biped and quadruped walking algorithms with CMAC [5][6]. This research has led to the current studies of vibration control with CMAC. The initial vibration control studies with CMAC have proved successful [3][7].

Vibration control is crucial in a vast range of applications and environments. Submarine warfare, precision milling and earthquake protection are just a few examples. Typical methods for vibration control are passive systems. Some common devices used in these systems are rubber mounts and shock absorbers. These devices are simple, reliable, inexpensive and require limited maintenance. The problem with these devices is the material used in their construction limits the dynamic range of the device. Furthermore, vibrations are dependent on a structure's material and geometry. It becomes extremely difficult to match the frequency response of a traditional controller or passive device with the vibrations of a particular object. This has led to the use of adaptive or artificial neural network (ANN) based control techniques [8][9][10]. These adaptive controllers are based on backpropagation networks. These networks are computationally intensive and typically result in slow cycle times. CMAC has been proven a highly effective adaptive controller for real-time applications that require high-speed [4][5][6].

## 1 Introduction

The Cerebellar Model Arithmetic Computer (CMAC) was developed by Albus to model the functionality of the cerebellum [1]. The cerebellum controls neuromuscular and coordinated movements throughout the body. CMAC was initially planned as a controller for artificial limbs [2]. CMAC has many properties making it ideal for real-time modeling, signal processing and control problems including rapid training, low memory requirements and faster cycle times than other neural networks [3][4]. CMAC has been applied to multiple applications of robotic control. Particularly, extensive

## 2 CMAC

### 2.1 Network Structure

The CMAC network is an associative neural network, using only a subset of the network's weight structure



for determination of the output. With only a small number of weights activated and used in the accumulation of the output, the network can very quickly formulate outputs, a clear advantage over many other ANNs. The training cycle is also extremely fast because only the same subset of weights needs to be adjusted. Furthermore, the number of weights in the subset is always the same. These properties together result in a high-speed network that is also deterministic, ideal for controls applications.

The associative properties of the network create local generalization—similar inputs give correlated outputs; while distant inputs produce uncorrelated outputs. This local generalization can be seen in the conceptual view of CMAC, Figure 1. In general terms, the input to a CMAC is a point in a multidimensional space. This point is expanded upon, according to the generalization parameter ( $C$ ), to force an overlapping in the conceptual memory. This can be seen in the state space ( $S$ ) of Figure 1, where two of the input values are close but not equal. The conceptual memory ( $A$ ) for these two points overlaps; consequently, these inputs share information and their outputs will have some level of correlation.

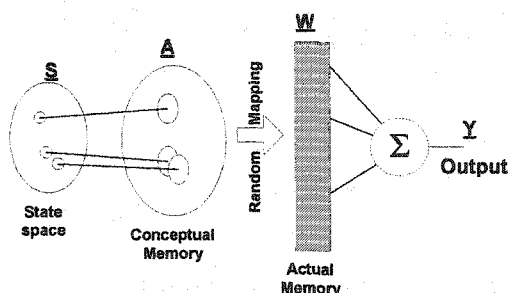


Figure 1. Conceptual View of CMAC

The inputs are mapped from the state space to a region of the CMAC conceptual memory. Each region of conceptual memory mapped by the input contains a specific and constant number of weights that is equal to the generalization parameter. These weights are also referred to as receptive field centers. The weights are typically stored in a traditional memory structure and a pseudo-random code is used to translate the conceptual memory address to the actual memory. The weights associated with each input are accumulated to form the network output. The mapping structure (translation from conceptual memory to actual memory) and the generalization parameter are predetermined and

held static. Adjusting the values within the weight vector produces the adaptive nature of the network. This is the only adaptive property in the network.

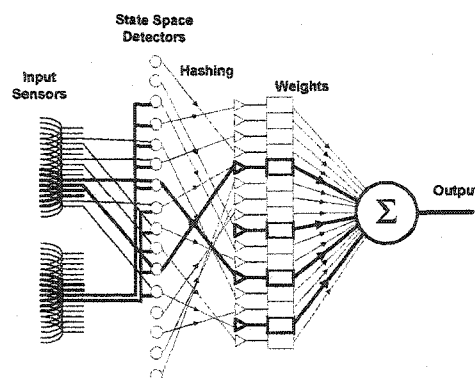


Figure 2. Two-Input CMAC Implementation

Figure 2 shows how an actual CMAC can be implemented with two inputs. The first stage of the network quantizes the input values and then generalizes both inputs over a larger area of the state space. All of the state space values included in the generalization are activated. These activated state spaces drive the state space detectors and the state space detectors perform a logical AND function. Each state space detector has a connection to each input dimension. When both of the connections are active, an associated weight is activated. The weight is consequently accumulated with other active weights to form the output value.

There are a few other important properties of the translation from conceptual memory to actual memory. First, it has already been mentioned that the mapping from the state space detectors to the weights is done in a pseudo-random fashion. A multidimensional input can map to an extremely large space. For example, a three input system with twelve-bit resolution will have 32 billion possible input states. Most applications will only use a very small subset of these possibilities. The random hashing allows us to map the extremely large input state space to a much smaller memory structure. This reduction in memory causes some inputs that are not meant to be associated with each other to map to the same weight. This phenomenon is known as a collision. Collisions do not cause a catastrophic error in the output because the output is formed by the accumulation of multiple weights. The collision only accounts for an amount that is dependent on the generalization parameter.

The second important property in the translation from input space to the weight structure is the fact that only the number of weights equal to the generalization is activated. In Figure 3, the generalization is four, therefore each input is spread over four states and together they form an area of sixteen states, but only four of these states map to receptive field centers, or weights. (The receptive field centers are represented by the black points in Figure 3.) The placement of these receptive fields is determined by the generalization. Each of the activated weights is offset along the hyper-diagonals of the input space. Each adjacent receptive field is offset by one quantization level.

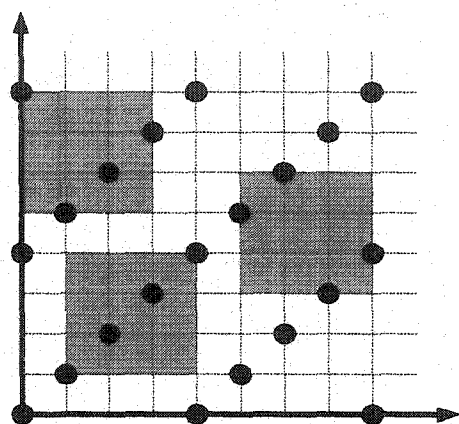


Figure 3. Receptive Field Centers

The shaded areas, in Figure 3, represent different inputs mapped onto these receptive fields. It can be seen that exactly four receptive fields are included in each shaded area. It is important to notice the reduction from the total number of possible inputs to the actual receptive field centers. This property and the hashing decrease the amount of memory needed.

In the original implementation of CMAC, each receptive field center is equally valued. It does not matter if the weight is a great distance from the actual input state. This property is more of a problem for networks with extremely large generalizations, like sixty-four. In these cases, it may not be prudent to equate weights at the fringes of the generalization area with weights in the center.

#### 2.1.1 Learning

The standard CMAC network's ability to learn is done through an adjustment of the weight vector, which represents the receptive field center. The following equation is the Least Mean Square learning algorithm. It is the most common learning algorithm.

$$\Delta W = (\beta/C)(f_o - w^T x_o)x_o \quad (1)$$

where  $\Delta W$  = weight adjustment value,  $w$  = current weight value,  $f_o$  = desired output,  $\beta$  = learning rate,  $C$  = generalization parameter, and  $x_o$  = field detector vector.

The field detector vector,  $x_o$ , controls the weights to be adjusted on each training pair. The traditional CMAC uses only binary values for the field detectors. This means a weight is either adjusted by the value of  $\Delta W$  or 0. This shows the local generalization of the weight vector, which is the main reason the CMAC network is so rapid. In common practice, the implemented algorithm only addresses the weights that are potentially adjusted and the full vector  $x_o$  is never actually used in the calculation.

#### 2.1.2 Rectangular Receptive Fields

Due to the fact that  $x_o$  is a binary vector, the output function typically takes the form of a staircase. As the input vector changes slightly, the output vector adds and subtracts weights without a smooth transition, thus forming sharp edges in the output. If the field detector vector is implemented as an integer value that varies with respect to the receptive field, weights can be added in proportion to the distance from the input vector, thus producing smoother output functions. Changing the receptive field shape from a rectangular function with binary values to a smooth, tapered function or a Gaussian shape can minimize the staircase output. These ideas were extensively studied [8][9]. The Gaussian receptive field is shown in Figure 4.

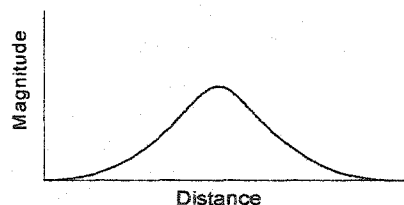


Figure 4. Gaussian Shaped Receptive Field

The distance from the receptive field center can be calculated in a variety of different manners. The three most common methods are the Euclidean method, the absolute minimum and the Manhattan. It is difficult to implement the square root function required by the Euclidean method in hardware. Consequently, the Manhattan distance will be used for this research. The Manhattan distance is the city block method where the distance is the accumulation of the distance along each input vector.

Another outcome of using the integer based field detectors is more flexibility in the learning algorithm. The follow equation explains the steepest descent weight updating method.

$$\Delta W = \frac{(\beta)(f_o - w^T x_o)x_o}{\sum_{n=1}^C (x_o)^2} \quad (2)$$

where  $\Delta W$  = weight adjustment value,  $w$  = current weight value,  $\beta$  = learning rate,  $f_o$  = desired output,  $C$  = generalization parameter, and  $x_o$  = field detector vector.

If the rectangular receptive field is used in this equation,  $x_o$  is a binary value with only  $C$  non-zero elements, and the equation diverts to the previous learning algorithm because the summation is simply  $C$ . In the case of a new receptive field shape, the weights are adjusted with respect to their impact on the output.

### 3 Hardware CMAC Implementation

#### 3.1 Reconfigurable Computing Platform

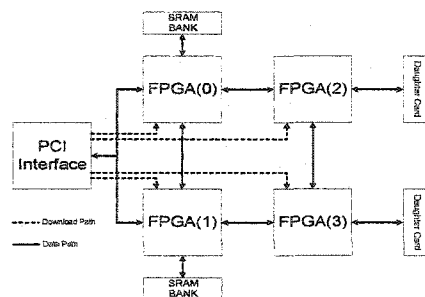


Figure 5. Pamette Reconfigurable Computing Platform

The Pamette board, shown in Figure 5, is the platform used during this research project [13].

As seen in the figure, the board consists of a 2x2 matrix of FPGAs. The FPGAs are XC4044 components from Xilinx. A PCI interface, also implemented in an FPGA, is used to get data to and from the board. It also controls the configuration of the user area FPGAs. The board also contains two SRAM banks for use as general-purpose memory devices. A set of connections is also available for a daughter card. Daughter cards can contain anything the developer desires, such as network devices or an A/D converter.

#### 3.2 General Network Components

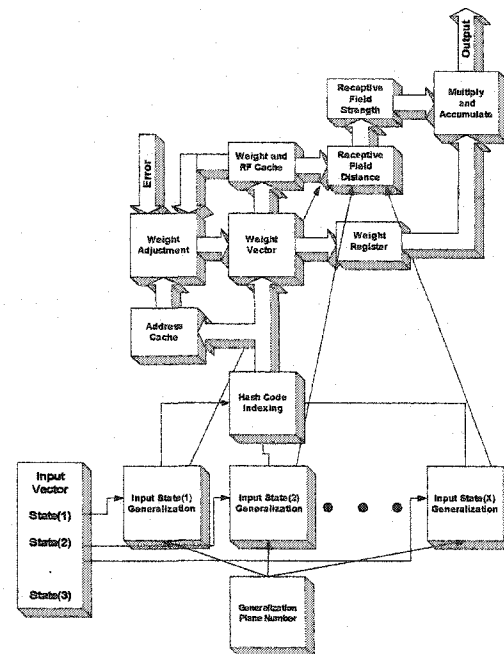


Figure 6. Hardware Block Diagram

The block diagram of the network is shown in Figure 6. Currently, the network is designed to handle a multiple number of inputs, multiple generalization values and different learning rates. There is only a single output value for the network, but two networks are mapped to the same reconfigurable board. It is important to emphasize that this is a reconfigurable platform and the current design is just a basis point. For example, extra outputs could be easily mapped, but they are not currently included because it would require dedicated weights and slow the response of the network.

The floor plan of the FPGA that contains the CMAC is shown in Figure 7. The weights are stored in a connected SRAM bank. A large portion of the FPGA is comprised of dedicated caches. The error correction or training components of the network use these caches for addresses, weights and receptive

field distances. This decreases the overall cycle time of the network. The accumulator is used to sum together weights and formulate the output. The multiplier is used to adjust the influence each weight has on the output. The address and SRAM control convert the input states to actual memory addresses. Finally, the control registers are used to set specific parameters of the network, like generalization and learning rate.

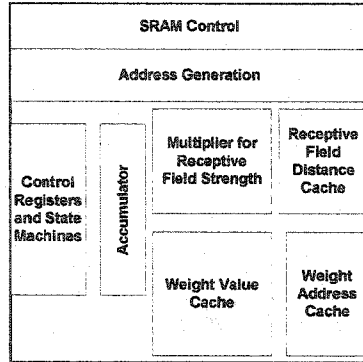


Figure 7. FPGA Floor plan

### 3.3 Address Generation Hashing

The address generation and hashing are extremely important in the network's ability to learn. The inputs need to be aligned along the hyper-diagonals of the weight space. This is accomplished using the following equation.

$$A_x = S_x - ((S_x - C_c) \bmod C) \quad (3)$$

where  $A_x$  = aligned address for input dimension  $x$ ,  $S_x$  = input state,  $C_c$  = generalization counter and  $C$  = generalization parameter.

Each input state forms its own aligned address. The bits are then reordered according to a preset configuration. Each input dimension uses a different bit ordering. These resulting single dimension pseudo-random addresses are accumulated to form the full multidimensional address. The accumulator is limited to the width of the SRAM address (17-bits) and is allowed to continually roll over. The generalization counter is incremented and the process repeats a number of times equal to the generalization. After a couple of stages of pipeline delay, a new memory address is produced on every clock tick.

### 3.4 Receptive fields

The receptive field distances are formed for each input dimension when the weight addresses are aligned along the hyper-diagonals of the state space. Equation 4 can be seen as the subtrahend of Equation 3.

$$Rf_x = ((S_x - C_c) \bmod C) \quad (4)$$

where  $Rf_x$  = receptive field distance for input dimension  $x$ ,  $S_x$  = input state,  $C_c$  = generalization counter and  $C$  = generalization parameter.

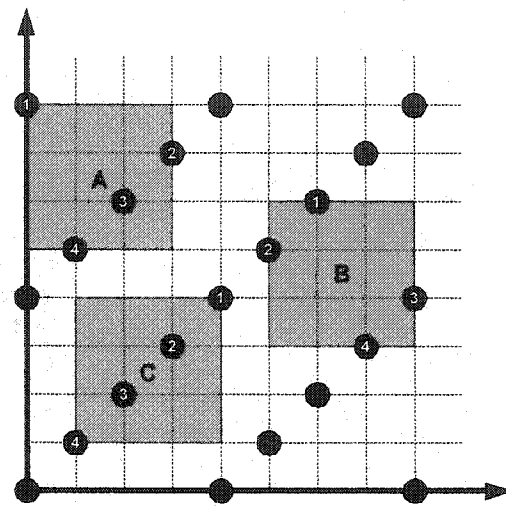


Figure 8. Receptive Field Distances

Each of these distances is summed. Figure 8 shows the three different weight configurations that are possible for a network with the generalization of 4. The input vector maps to the upper right hand corner of the shaded area and the distances are calculated from that point. Tables 1 and 2, below, demonstrate how each shaded box has the same total distance to all of its weights.

Table 1. Receptive Field Weights for Area A

Weight	Distance	Receptive Field Strength
1	3	4
2	1	6
3	3	4
4	5	2
<b>Total</b>	<b>12</b>	<b>16</b>

Table 2. Receptive Field Weights for Area B

Weight	Distance	Receptive Field Strength
1	2	5
2	4	3
3	2	5
4	4	3
Total	12	16

The maximum distance any weight can be from the input in one dimension is one less than the generalization parameter. The overall maximum distance is one minus the generalization times the number of input dimensions. This value determines the bit width of the receptive field strength and distance. The value used for the receptive field strength is the one's complement of the receptive field distance. Table 3 shows all the possible distances and strengths for a network with two inputs and a generalization of 4.

Table 3. Receptive Field Strength ( $C=4$ )

Receptive Field Distance	RF Distance (binary)	Receptive Field Strength	RF Strength (binary)
0	000	7	111
1	001	6	110
2	010	5	101
3	011	4	100
4	100	3	011
5	101	2	010
6	110	1	001

An interesting property arises when this function is used for the receptive strength; the sum total for the receptive strength is a power of two, as long as the input vector and generalization parameter are also a power of two. This is shown in Tables 1 and 2, where the third column sums to 16. The fact that 16 is a power of two simplifies the design. It allows division to be a bit shift and the modulus operator to be a simple truncation.

Figure 9 is a plot of the receptive field shape. Although the plot still appears to be a staircase, the function varies with every change of the least significant data bit. This function minimizes the jagged output, which is common with CMAC.

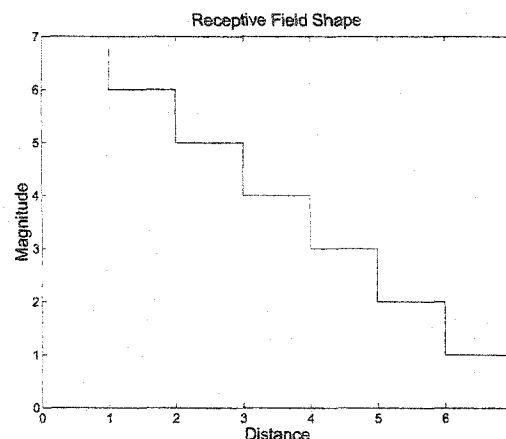


Figure 9. Receptive Field Shape

### 3.5 Caching

There is a well-defined bottleneck in this CMAC implementation and all other CMAC implementations, the SRAM or weight vector. In order to relieve pressure on this interface, all the weights that are used to calculate the current output value are cached. The actual SRAM address and the receptive field distances are also cached. Instead of regenerating these addresses and performing a read-modify-write on each weight, the address and weight are removed from cache. The weight is updated according to its' receptive field strength and written to the SRAM.

## 4 Performance

### 4.1 Previous Hardware Implementations

There have been two previous implementations of CMAC in hardware: a version that used one FPGA for address and control, and another FPGA for weight accumulation, and a second design that was a replica done in VLSI [14][15]. Both of these designs had a hardwired receptive field size of 512. They could use any size for the generalization. This causes a very jagged output for networks with a generalization of less than 512. The VLSI version allowed for receptive field shape control. However, it is important to state that only one of the two chips in the VLSI design was actually manufactured and the authors are unclear how the individual weights would have been scaled during the error correction cycle. The new design forces the generalization and input vector to be powers of two. This eliminates any difficulties with scaling. The power of two restriction on the number of inputs can be overcome by mapping the same state to multiple inputs on the

same network and forcing the input to map directly onto the hyper-diagonal.

Advances in FPGA technology have allowed the authors to double the clock speed of the design. Both of the previous implementations were designed to run at 16 MHz. The current implementation operates at 33 MHz. Another large gain in performance is the result of the new address generation sequence. Addresses for each input state are calculated in parallel with a couple of stages of pipeline used to accumulate them into the final SRAM address. The previous implementations calculated the SRAM address by serially processing the input vector. Consequently, the new implementation has a performance increase equal to twice the number of states in the input vector. For example, a network with 4 input states operates 8 times faster for the output formulation cycle. Furthermore, the learning cycle, which previously regenerated the address, fetched the weight and corrected it, was twice as long as the output formulation cycle. The use of cached information has also reduced this cycle time. Using the same example, a network with four inputs is 16 times faster. The processing times for all implementations are still dependent on the generalization parameter.

#### 4.2 Learning Ability

Figure 10 shows an individual network learning a sinusoid function with a single input state. The plot shows three learning passes of the network. The network was given one fifth of the waveform for training data and then reconstructed the entire waveform. The learning rate was 0.5 and the generalization was 8. It can be seen that the network converges on sinusoid after each pass.

Figure 11 demonstrates the network's ability to learn multidimensional functions. In this case, the CMAC learns a two-dimensional function derived from the sinusoids. The plot shows the network recreating the entire function after three learning passes. Once again, the network was given one fifth of the waveform for training data and then reconstructed the entire waveform. The learning rate was 0.5 and the generalization was 8.

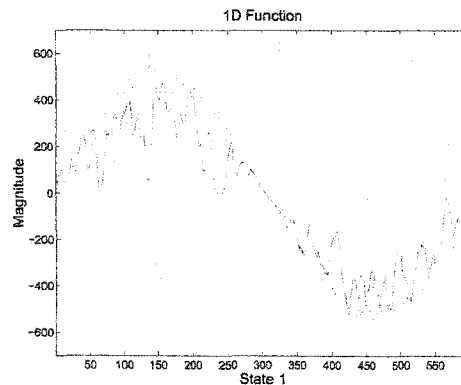


Figure 10. Learning Convergence

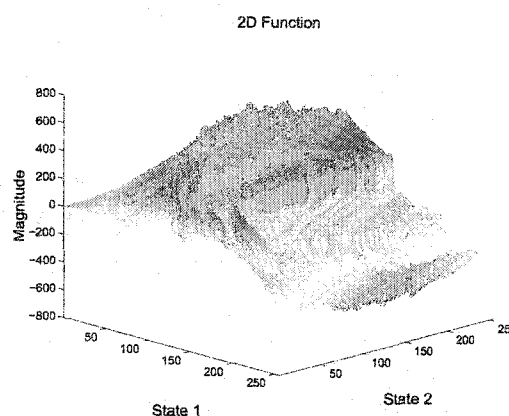


Figure 11. 2-D Function

#### 4.3 Simulated Vibration Control System

The network has been used to simulate the following three configurations: a cantilever beam, a floating beam and a floating table. The cantilever beam has a single sensor and CMAC controller. The floating beam has two sensors and two CMAC controllers. Finally, the table has four sensors and four controllers. The systems are assumed to have one source for the disturbance.

Tables 4 and 5 show the runtime for a set of experiments comparing the previous software, which was run on a 400 MHz Pentium II, to the same system with the new CMAC processor added. The experiments used the function in Figure 11 and each system made 9 learning passes where they trained from 25,000 random samples. Tables 4 and 5 have generalizations of 8 and 16, respectively.

Table 4. Performance Comparison (C=8)

Number of CMACs	Software Runtime	Hardware Runtime	Percent Decrease
1	7.768 s	1.781 s	436%
2	15.844 s	1.797 s	881%
4	31.109 s	3.516 s	886%

With just one CMAC, the new implementation is over four times faster. If another CMAC is added to the system, the inputs to the second CMAC can be written, while the first one is still calculating the output. By interleaving the operation of the two networks, the performance is increased by another factor of two. However, the performance is not increased for the four CMAC case at this generalization. In this case, the CMAC processor is not the limiting case and it completes a cycle before the CPU has time to deliver new inputs. The pure software solution increases linearly in runtime, but more processors could be added. Clearly, this would help the software solution but synchronizing of the CMAC controllers would have to be done and the cost of the system could rise dramatically. For the software solution to meet the results of the CMAC processor, the microprocessor would have to operate at 1.6 GHz, which is currently not available.

Table 5. Performance Comparison (C=16)

Number of CMACs	Software Runtime	Hardware Runtime	Percent Decrease
1	15.674 s	3.517 s	445%
2	30.089 s	3.529 s	857%
4	58.907 s	3.569 s	1651%

Table 5 shows the same experiment as Table 4 but the generalization parameter is increased to 16. With a larger generalization, each processor spends more time calculating the output and control of all four CMACs can be interleaved. Thus, the CMAC coprocessor produces an even larger performance increase as seen in Table 5.

## 5 Conclusion

The CMAC network has been successfully implemented on the PCI Pamette board. The pipelined design produces dramatic increases in performance. The authors are attempting to increase the performance of their current PC based vibration control system by a factor of 10.

For a single CMAC with a generalization of 8, the authors were using a cycle time of 10 microseconds as the performance goal. The new CMAC coprocessor is producing cycle times below 4 microseconds for the full cycle of output formulation followed by the error correction. These performance numbers will allow the authors to explore vibration control at frequencies significantly greater than initially planned.

There is still a large amount of work to be completed. The network has proven that it can learn functions of at least two dimensions. This corresponds to the original vibration control using CMAC. The hardware now needs to be tested for higher dimensionality problems. The CMAC will also be adapted to include weight smoothing. The discontinuous weight space in CMAC can cause large changes in weights and this can be detrimental to learning. The addition of weight smoothing and more in-depth study of mapping collisions are currently in progress. A study of more complex learning algorithms for multiple interconnected CMACs is also planned.

With the success of these initial performance numbers, the hardware is being applied to real-time vibration control. The clock speed of the CMAC implementation is also being increased and, eventually, the hardware will also be applied to audio noise cancellation.

## 6 Acknowledgements

This work was partially sponsored by the National Science Foundation, Knowledge Modeling and Computational Intelligence, Director Dr. Paul Werbos. The authors would like to thank Mark Shand and Laurent Moll of Compaq's System Research Center for their help with Pamette.

## References

Authors in bold are associated with the UNH Robotics Lab.

- [1] Albus, J.S., "The Cerebellar Model Articulation Controller", *Trans. ASME Series G*, vol.97, No.3, 1975.
- [2] Albus, J.S., "A New Approach to Manipulator Control: the Cerebellar Model Articulation Controller", *Trans. ASME, J. Dynamic Syst. Meas. Contr.*, vol. 97, pp. 220-227, 1975. (September 1975)

- [3] Kraft, L.G. and Pallotta, J., "Vibration Control Using CMAC Neural Networks with Optimized Weight Smoothing", *Proceedings of 1999 ACC*, San Diego, CA.
- [4] Miller, W.T., Hewes, R.P., Glanz, F.H. and Kraft, L.G., "Real-time Dynamic Control of an Industrial Manipulator Using a Neural Network Based Learning Controller," *IEEE Journal of Robotics and Automation*, February, 1990.
- [5] Kun, Andrew L., "A Sensory-Based Adaptive Walking Control Algorithm for Variable Speed Biped Robot Gaits", Ph.D. Dissertation, University of New Hampshire, Durham, NH, May 1997.
- [6] Woodward, Jeffrey, "Application of CMAC Neural Network Learning to a Stepping Quadruped Robot", Master Thesis, University of New Hampshire, Durham, NH, December 1996.
- [7] Pallotta, J. and Kraft, L.G., "Two Dimensional Function Learning Using CMAC Neural Networks with Optimized Weight Smoothing", *Proceedings of 1999 ACC*, San Diego, CA.
- [8] K.G. Ahn, H.J. Pakk, M.Y. Jung and D.W. Cho, "A Hybrid-Type Active Vibration Isolation System Using Neural Networks." *Journal of Sound and Vibration*, vol. 192, No. 4, pp.793-805, 1996.
- [9] Snyder, Scott D. and Tanaka, Nobou. "Active Control of Vibration using a Neural Network." *IEEE Transactions on Neural Networks*, vol.6, no. 4, pp.819-828, July, 1995.
- [10] Ma, R.P. and Sinha, A. "A Neural Network Based Active Vibration Absorber with State Feedback Control." *Journal of Sound and Vibration*, ver. 190, no.1, pp.121-128, 1996.
- [11] An, Pak-Cheung Edgar, "An Improved Multidimensional CMAC Neural Network: Receptive Field Function and Placement.", Ph.D. Dissertation, University of New Hampshire, Durham, NH, September 1991.
- [12] S.H. Lane, M.G. Flax, D.A. Handelman, and J.J. Gelfand, "Multi-layer perceptrons with B-spline receptive field functions," *Advances in Neural Information Processing Systems*, vol. 3, R.P. Lippman, J.Moody, and D.S. Touretzky, Eds. Morgan Kaufman, 1991.
- [13] Compaq Computer Corp., "PCI Pamette V1," <http://www.research.digital.com/SRC/pamette>.
- [14] Box, Brian, "Hardware implementation of a CMAC neural network using field programmable gate arrays", Master's Thesis, University of New Hampshire, Durham, NH, September 1990.
- [15] Yang, Bing, "A VLSI implementation of CMAC neural network", Master's Thesis, University of New Hampshire, Durham, NH, September 1992.