Correlation with the hermite series using artificial neural network technology

Mark Mackenzie
University of Wollongong
CORRELATION WITH THE HERMITE SERIES USING
ARTIFICIAL NEURAL NETWORK TECHNOLOGY

A thesis submitted in fulfillment of the requirements for the
award of the degree

DOCTOR OF PHILOSOPHY

from

UNIVERSITY OF WOLLONGONG

by

MARK MACKENZIE

DEPARTMENT OF MECHANICAL, MATERIALS AND
MECHATRONIC ENGINEERING

2004
DECLARATION

I, Mark Mackenzie, declare that this thesis, submitted in fulfillment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering, University of Wollongong, Australia is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Mark Mackenzie

ACKNOWLEDGEMENTS

I would firstly like to acknowledge my supervisor, Professor Kiet Tieu, for his guidance, wisdom and experience throughout this thesis and in particular his continuous encouragement despite setbacks.

I would also like to acknowledge the anonymous reviewers of the publications and conferences of this thesis, which have significantly improved its content.

Finally I would like to thank the staff of the Faculty of Engineering at the University of Wollongong.
PUBLICATIONS

JOURNAL ARTICLES


SUBMITTED TO JOURNALS


CONFERENCE AND WORKSHOP PROCEEDINGS


This thesis investigates the correlation of Hermite functions in the form of a Hermite neural network. The terminology “Hermite neural network” describes the Hermite series of orthonormal functions, which have been generalized to include numerical algorithms that are often associated with artificial neural networks.

The aim of the investigation was to determine whether there are any benefits over the traditional correlation method using Fourier functions. The relative performance of the Hermite neural network correlator was compared to a Fourier neural network correlator. The main result is that the correlation of a Hermite neural network is a summation of $N \times N$ associated Laguerre functions whereas a Fourier neural network correlation is a summation of $N$ Fourier functions. In this regard the Fourier neural network will be more efficient for the general correlation of functions. An exception occurs for the correlation of the Hermite neural network with a Gaussian function, or with the CHIRP radar signal. For these signals the correlation is also a summation of $N$ terms. In these applications the Hermite correlator proved to be superior to the Fourier correlator for the following reasons.

- It does not suffer from a circular correlation error, which is a characteristic of the Fourier correlator.

- It allows the Gaussian inverse correlation to be computed without the numerical instability that occurs with a Fourier correlator.

- It achieves a more compact signal interpolation for the CHIRP radar signal correlator than is possible with a Fourier correlator.
The relatively good performance of the Hermite correlation, particularly the numerical stability of the inverse correlation, can be expected to be a useful asset for image processing, where the Gaussian function is especially important.

As a digression from the main topic of Hermite neural network correlation, during the investigation a new method of fast training the sigmoid neural network was discovered. The principle of the new training method is that it trains the sigmoid neural network to the rate of change of the unknown non-linear function rather than to the function itself. This allows the sigmoid neural network to be trained with an associated radial basis neural network with the speed that is inherent with this type of network. In tests the associated training method was 100 times faster than the conventional training method for the sigmoid neural network. The new training method can be expected to widen the application of the sigmoid neural network to include applications that have previously not been possible on account of the slow training.
**TABLE OF CONTENTS**

Declaration ........................................ II
Acknowledgements .................................. III
Publications ........................................ IV
Abstract ........................................... VI
Table of Contents ................................ VIII
List of Figures ..................................... XII
List of Tables ..................................... XXI
Nomenclature ...................................... XXII

**CHAPTER 1 INTRODUCTION** .......................... 1

1.1 Motivation, contribution and method .......... 1
1.2 Thesis outline .................................. 2
1.3 Contents of thesis. ............................. 4

**CHAPTER 2 ARTIFICIAL NEURAL NETWORKS** .... 8

2.1 Introduction ................................... 8
2.2 A brief historical summary .................. 8
2.3 Biological inspired artificial neural networks .. 10
2.4 Single layer neural networks for function approximation .. 14
2.5 Sigmoid neural networks and network training .. 19
2.6 Radial basis neural networks .................. 23
2.7 Kernel regression .............................. 25
2.8 Orthonormal neural networks .................. 27
2.9 Multi-layer neural networks ................... 33
2.10 Conclusion .................................... 36
CHAPTER 3 HERMITE NEURAL NETWORK

3.1 Introduction

3.2 Quantum physics solution for probability density

3.3 Hermite functions

3.4 Hermite neural network

3.5 Duration and bandwidth of the Hermite neural network

3.6 Training of the Hermite neural network

3.7 Scaling the Hermite neural network

3.8 Comparison with the Fourier neural network

3.9 Conclusion

CHAPTER 4 HERMITE NEURAL NETWORK CORRELATION

4.1 Introduction

4.2 Correlation using Hermite neural networks

4.3 Convolution of two Hermite neural networks

4.4 Recurrence relations for the Laguerre functions

4.5 Comparison with the Fourier correlator

4.6 Error of the Hermite and Fourier correlators

4.7 Auto-correlation of a finite duration sine wave

4.8 Correlation of rectangular pulses

4.9 Conclusion

CHAPTER 5 GAUSSIAN FILTERS AND FILTER SYNTHESIS

5.1 Introduction

5.2 Gaussian filter

5.3 Interpretation as a Gaussian Radial Basis Neural Network

5.4 Equivalence to kernel regression
5.5 Frequency response 103
5.6 Filter synthesis 104
5.7 Method of application of the Hermite/Laguerre filter 106
5.8 Application to the filtering of noisy sine waves 108
5.9 Signal to noise ratio of Hermite/Laguerre filter 117
5.10 Gaussian inverse filters 120
5.11 Fourier inverse filter 124
5.12 Hermite/Laguerre inverse filter 125
5.13 Training the Hermite/Laguerre inverse filter 126
5.14 Effect of the LMS algorithm on the frequency response of the Hermite/Laguerre inverse filter 127
5.15 Pseudo-Weiner filter 130
5.16 De-blurring of laser Doppler anemometry 132
5.17 Conclusion 137

CHAPTER 6 ERROR ANALYSIS OF HERMITE/LAGUERRE FILTER 140
6.1 Introduction 140
6.2 Equivalence between Kernel regression and the Hermite/Laguerre filter 141
6.3 Error of Hermite/Laguerre filter 143
6.4 Boundary error 147
6.5 Asymmetrical kernel regression 150
6.6 Test results of asymmetric kernel regression 156
6.7 Conclusion 162

CHAPTER 7 APPLICATION OF HERMITE CORRELATION TO RADAR 164
7.1 Introduction 164
7.2 Matched filter detection of radar signals 165
7.3 Doppler radar signals 174
7.4 Frequency modulated radar 182
7.5 Conclusion 192

CHAPTER 8 ASSOCIATED TRAINED SIGMOID NEURAL NETWORK 194
8.1 Introduction 194
8.2 Principle of associated trained sigmoid neural network 195
8.3 Differentiation of the training data 200
8.4 Training of the associated sigmoid neural network 201
8.5 Constant of integration 202
8.6 Network centers and widths 202
8.7 Convolution based notation and Fourier transform 203
8.8 Analytical solution for amplitude weights 205
8.9 Noise filtering properties of associated training 209
8.10 Test results for the associated training algorithm 211
8.11 Conclusion 224

CHAPTER 9 CONCLUSION AND RECOMMENDATION 227
REFERENCES 230
LIST OF FIGURES

Figure 2.1  Schematic diagram of a biological neuron 11
Figure 2.2  Mathematical model of biological neuron 11
Figure 2.3  Schematic layout of a single input/output, single
layer neural network 13
Figure 2.4  Detail of the $m^{th}$ node of the neural network of Figure 2.3. 13
Figure 2.5  Classification of neural networks based on the type
of activation function. 16
Figure 2.6 (a)  Rectangular activation function (local). 16
Figure 2.6 (c)  Unit-step activation function (global). 16
Figure 2.6 (b)  Gaussian activation function (local). 16
Figure 2.6 (d)  Logic sigmoid activation function (global) 16
Figure 2.7  Neural network approximation of $y(t)$ using rectangular
local activation functions. 17
Figure 2.8  Neural network approximation of $y(t)$ using unit-step
global activation functions. 17
Figure 2.9  Schematic layout of a multi-layer artificial neural network 34
Figure 2.10  Detail of the $m^{th}$ node of the neural network of Figure 2.9 34
Figure 3.1  Approximate solution of the Schrodinger equation for
the wave function, $\psi(x)$, of an oscillator. 42
Figure 3.2 (a)  Hermite function of order $\{0\}$, the Gaussian function. 45
Figure 3.2 (b)  Hermite function of order $\{1\}$ 45
Figure 3.2 (c)  Hermite function of order $\{2\}$ 45
Figure 3.2 (d)  Hermite function of order $\{3\}$ 46
Figure 3.2 (e) Hermite function of order {4}  

Figure 3.2 (f) Hermite function of order {5}  

Figure 3.3 Classical and quantum mechanical probability density solutions for the harmonic oscillator ($n=10$).  

Figure 3.4 Hermite function of order {10}, corresponding to figure 3.3.  

Figure 3.5 Hermite neural network interpolation of rectangular function.  

Figure 3.6 Error of Hermite neural network interpolation of rectangular function.  

Figure 3.7 Fourier neural network interpolation of rectangular function.  

Figure 3.8 Error of Fourier neural network interpolation of rectangular function.  

Figure 3.9 Hermite neural network Fourier transform of rectangular function.  

Figure 3.10 Error of Hermite neural network Fourier transform of rectangular function.  

Figure 3.11 Fourier neural network Fourier transform of rectangular function.  

Figure 3.12 Error of Fourier neural network Fourier transform of rectangular function.  

Figure 3.13 Rmse versus the number of activation functions for the rectangular function  

Figure 3.14 Rmse versus the sampling rate (number of samples per unit) for the rectangular function  

Figure 3.15 Training speed using the gradient descent algorithm (random initial weights).
Figure 4.1 (a)  Associated Laguerre function $l_{m}^{n}(t)$, where $(m=0, n=100)$. 76
Figure 4.1 (b)  Associated Laguerre function $l_{m}^{n}(t)$, where $(m=0, n=200)$. 76
Figure 4.1 (c)  Associated Laguerre function $l_{m}^{n}(t)$, where $(m=0, n=300)$. 76
Figure 4.2 (a)  Associated Laguerre function $l_{m}^{n}(t)$, where $(m=100, n=100)$. 77
Figure 4.2 (b)  Associated Laguerre function $l_{m}^{n}(t)$, where $(m=200, n=100)$. 77
Figure 4.2 (c)  Associated Laguerre function $l_{m}^{n}(t)$, where $(m=300, n=100)$. 77
Figure 4.3  Hermite interpolation of $x(t)$. 83
Figure 4.4  Error of Hermite interpolation of $x(t)$. 83
Figure 4.5  Hermite correlation $r(t) = x(t) \ast x(t)$. 84
Figure 4.6  Error of Hermite correlation $r(t) = x(t) \ast x(t)$. 84
Figure 4.7  Fourier interpolation of $x(t)$. 86
Figure 4.8  Error of Fourier interpolation of $x(t)$. 86
Figure 4.9  Fourier correlation $r(t) = x(t) \ast x(t)$. 87
Figure 4.10  Error of Fourier correlation $r(t) = x(t) \ast x(t)$. 87
Figure 4.11 (a)  Hermite interpolation of $x(t)$. 89
Figure 4.11 (b)  Hermite interpolation of $x(t-2.5)$. 89
Figure 4.11 (c)  Hermite correlation $r(t) = x(t) \ast x(t-2.5)$. 89
Figure 4.12  Enlargement of figure 4.11 (c), Hermite correlation $r(t) = x(t) \ast x(t-2.5)$. 90
Figure 4.13  Error of Hermite correlation $r(t) = x(t) \ast x(t-2.5)$. 90
Figure 4.14  Fourier correlation $r(t) = x(t) \ast x(t-2.5)$. 91
Figure 4.15  Error of Fourier correlation $r(t) = x(t) \ast x(t-2.5)$. 91
Figure 4.16  Root mean square error (rmse) versus displacement, tau. 93
Figure 4.17 (a) Hermite correlation $\tau=8.5$.

Figure 4.17 (b) Hermite correlation $\tau=9.0$.

Figure 4.17 (c) Hermite correlation $\tau=9.5$.

Figure 4.18 (a) Fourier correlation $\tau=8.5$.

Figure 4.18 (b) Fourier correlation $\tau=9.0$.

Figure 4.18 (c) Fourier correlation $\tau=9.5$.

Figure 5.1 Gaussian window, moving average filter.

Figure 5.2 Associated Laguerre functions, $l_0^0(t^2/2)$, of increasing order.

Figure 5.3 Comparison of $l_0^0(t^2/2)$ with a Gaussian function.

Figure 5.4 Diagram of Hermite/Laguerre neural network.

Figure 5.5 Cascaded Hermite/Laguerre neural network.

Figure 5.6 Frequency response of the Hermite/Laguerre and Synthesized filters.

Figure 5.7 Close up of the frequency response of the Laguerre and Synthesized outputs.

Figure 5.8 Root mean square error versus number of activation functions.

Figure 5.9 Hermite/Laguerre neural network training speed.

Figure 5.10 Hermite interpolation of sine wave.

Figure 5.11 Hermite error of sine wave interpolation.

Figure 5.12 Laguerre interpolation of sine wave.

Figure 5.13 Laguerre error of sine wave interpolation.

Figure 5.14 Sigmoid neural network training speed.

Figure 5.15 Sigmoid neural network interpolation of sine wave.

Figure 5.16 Sigmoid neural network interpolation of sine wave.
Figure 5.17 (a) Noisy sine wave, period=40.0; SNR=7.2.

Figure 5.17 (b) Hermite output; SNR=22.2.

Figure 5.17 (c) Laguerre output; SNR=149.7.

Figure 5.18 Laguerre and Hermite output SNR compared with theory.

Figure 5.19 Comparison of SNR with kernel regression

Figure 5.20 Comparison of SNR with synthesized filter

Figure 5.21 Comparison with the Weiner filter.

Figure 5.22 Comparison of SNR with sigmoid neural network

Figure 5.23 Inverse Gaussian filter composed of Hermite and Laguerre neural networks.

Figure 5.24 Frequency response of Laguerre input of the inverse filter as a function of the number of training cycles of the LMS algorithm from 0 to 100 in steps of 10 cycles.

Figure 5.25 Frequency response of Hermite output of inverse filter as a function of the number of training cycles of the LMS algorithm.

Figure 5.26 Cut-off frequency of the Hermite/Laguerre inverse filter versus the number of LMS training cycles.

Figure 5.27 Comparison of Hermite/Laguerre inverse filter with the pseudo-Weiner inverse filter.

Figure 5.28 Distorted velocity profile from laser Doppler anemometer.

Figure 5.29 Displacement of velocity profile from data center line

Figure 5.30 (a)-(i) Numerical stability of inverse filters as the network size is increased.

Figure 5.31 (a)-(i) Off-axis numerical stability of inverse filters.
| Figure 6.1 | Bias error of Hermite/Laguerre filter. | 145 |
| Figure 6.2 | Gaussian kernel regression showing truncation of the kernel at the boundary. | 148 |
| Figure 6.3 | Correction of boundary bias error by reduction of the variance of the Gaussian kernel close to the boundary. | 148 |
| Figure 6.4 | Gamma kernel functions. | 151 |
| Figure 6.5 | Beta kernel functions. | 151 |
| Figure 6.6 | Gamma and Gaussian kernel regression of sine wave. | 157 |
| Figure 6.7 | Gamma and Gaussian kernel regression bias error of sine wave. | 157 |
| Figure 6.8 | Comparison of Gamma and Beta kernel regression of sine wave. | 158 |
| Figure 6.9 | Gamma and Beta kernel regression bias error of sine wave. | 158 |
| Figure 6.10 | Gamma and variable bandwidth Gaussian kernel regression of noisy sine wave. | 160 |
| Figure 6.11 | Output noise of Gamma and variable bandwidth Gaussian kernel regression of noisy sine wave. | 160 |
| Figure 6.12 | Gamma, Beta and variable bandwidth Gaussian kernel regression of noisy sine wave. | 161 |
| Figure 6.13 | Output noise of Gamma, Beta and variable bandwidth Gaussian kernel regression of noisy sine wave. | 161 |
| Figure 7.1 | Simplified schematic diagram of typical radar system | 166 |
| Figure 7.2 | Simplified schematic diagram of radar system with the addition of a frequency modulator. | 166 |
| Figure 7.3 (a) | Noisy returned radar signal. | 170 |
| Figure 7.3 (b) | Hermite output of interpolation of noisy radar signal | 170 |
| Figure 7.3 (c) | Laguerre output (matched filter) of noisy radar signal. | 170 |
Figure 7.4  Signal to noise ratio of demodulated signal by HNN and FNN.  

Figure 7.5  Signal to noise ratio of demodulated signal by HNN and conventional sigmoidal neural network (SNN) 

Figure 7.6 (a)  Fourier correlator, $\tau = 5.0$. 

Figure 7.6 (a)  Fourier correlator, $\tau = 7.0$. 

Figure 7.6 (c)  Fourier correlator, $\tau = 9.0$. 

Figure 7.6 (a)  Hermite correlator, $\tau = 5.0$. 

Figure 7.6 (b)  Hermite correlator, $\tau = 7.0$. 

Figure 7.6 (c)  Hermite correlator, $\tau = 9.0$. 

Figure 7.7 (a)  HNN approximation of noisy Doppler radar signal. 

Figure 7.7 (b)  HNN demodulation of noisy Doppler radar signal. 

Figure 7.7 (c)  HNN Fourier transform of noisy Doppler radar signal. 

Figure 7.8  Root mean square error (rmse) versus number of Hermite functions. 

Figure 7.9  Root mean square error versus number of Hermite-Rodriguez functions. 

Figure 7.10  Frequency response of HNN of size $n=70$ 

Figure 7.11  Frequency response of HRNN of size $n=70$ and $n=1600$. 

Figure 7.12  Signal to noise ratio of demodulated signal by HNN and HRNN. 

Figure 7.13  Hermite signal of order 18. 

Figure 7.14  CHIRP signal matched to Hermite function of order 18. 

Figure 7.15  Correlated Hermite signal of order 18. 

Figure 7.16  Correlated CHIRP signal with the width of the first peak matched to the Hermite correlated signal of Figure 7.16 

Figure 7.17  Root mean square error of FNN approximation of Hermite
signal versus number of Fourier functions.

Figure 7.18 Root mean square error of FNN approximation of Hermite signal versus sampling rate.

Figure 7.19 SNR of Hermite signal using HNN and FNN correlators

Figure 7.20 SNR of Hermite signal and CHIRP signal with FNN correlators.

Figure 7.21 SNR of Hermite signal with HNN correlator and CHIRP signal with FNN correlator.

Figure 8.1 Standard trained sigmoid neural network

Figure 8.2 Associated trained sigmoid neural network

Figure 8.3 Associated sigmoid kernel function.

Figure 8.4 Fourier transform of associated sigmoid kernel function.

Figure 8.5 Effect of sampling rate on rmse of a sigmoid neural network with associated training and 20 activation functions.

Figure 8.6 Frequency response of sigmoid neural network with associated training and 20 activation functions.

Figure 8.7 Associated training without LMS ($\alpha = 0.5$) and 3 activation functions.

Figure 8.8 Associated training with 50 training cycles of LMS ($\alpha = 0.5$) and 3 activation functions.

Figure 8.9 Associated training with 100 training cycles of LMS ($\alpha = 0.5$) and 3 activation functions.

Figure 8.10 Error of amplitude weights with respect to the LMS optimum solution (6 activation functions).

Figure 8.11 Error of amplitude weights with respect to the LMS
solution (9 activation functions). 216

Figure 8.12 Error of amplitude weights with respect to the LMS solution (18 activation functions). 217

Figure 8.13 Associated training curves ($\alpha = 0.1$) with approximate solution of equation (29). 220

Figure 8.14 Associated training curves ($\alpha = 0.1$) with approximate analytical solution of equation (41). 220

Figure 8.15 Conventional neural network training curves ($\alpha = 0.1$). 221

Figure 8.16 Comparison of theoretical estimation of SNR with actual SNR for an associated trained sigmoid neural network with 20 activation functions. 221

Figure 8.17 SNR of Associated trained sigmoid neural network versus conventionally trained network (6 activation functions). 223

Figure 8.18 SNR of Associated trained sigmoid neural network (6 activation functions) versus Gaussian kernel regression. 223
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 4.1</td>
<td>Location of the numerical instability for the recurrence relation for the generation of the associated Laguerre functions</td>
<td>78</td>
</tr>
<tr>
<td>Table 8.1</td>
<td>Weights for network with 6 activation functions</td>
<td>215</td>
</tr>
<tr>
<td>Table 8.2</td>
<td>Weights for network with 9 activation functions</td>
<td>216</td>
</tr>
<tr>
<td>Table 8.3</td>
<td>Weights for network with 18 activation functions</td>
<td>217</td>
</tr>
</tbody>
</table>
NOMENCLATURE

\( a_j \) \( j \) th amplitude weight of neural network

\( a(t) \) amplitude weight function

\( A(\omega) \) Fourier transform of amplitude weight function

\( b_j \) \( j \) th bias weight of neural network, \( j \) th amplitude weight of neural network

\( B \) bandwidth of function

\( c_j \) \( j \) th amplitude weight of Fourier neural network.

\( c \) neural network constant

\( D \) duration of function

\( e(t) \) online error of neural network

\( f \) frequency, Hertz.

\( f(x) \) probability density

\( FNN \) Fourier Neural Network

\( g(t) \) Gamma density

\( h_j(t) \) \( j \) th orthonormal Hermite function

\( H_j(t) \) \( j \) th Hermite polynomial

\( HNN \) Hermite Neural Network

\( hr_j(t) \) \( j \) th Hermite-Rodriguez function

\( HRNN \) Hermite-Rodriguez Neural Network

\( i \) integer
\( j \) integer

\( l_i^j(t) \) \( i \), \( j \) th associated Laguerre orthonormal function

\( L_i^j(t) \) \( i \), \( j \) th associated Laguerre polynomial

\( m \) mass of particle; integer

\( m(t) \) noise at filter output

\( M \) largest integer of summation

\( mse \) mean square error

\( n \) integer

\( N \) largest integer of summation

\( n(t) \) noise

\( r(t) \) correlation function.

\( r_h(t) \) Hermite neural network approximation of correlation function.

\( r_\Theta(t) \) Fourier neural network approximation of correlation function.

\( RBNN \) Radial Basis Neural Network

\( rmse \) root mean square error

\( s(t) \) Sigmoid function

\( SNN \) Sigmoid Neural Network

\( t \) input variable, time variable.

\( T \) period of Fourier function.

\( u(t) \) ideal filter

\( U(\omega) \) Fourier transform of ideal filter

\( u_j \) \( j \) th input weight of neural network.

\( v_j \) \( j \) th output weight of neural network.
$w_j$ \hspace{1em} j\text{th weight of artificial neural network} \equiv u_j, v_j, b_j \text{ or } c.

$x(t)$ \hspace{1em} training function

$X(\omega)$ \hspace{1em} Fourier transform of $x(t)$.

$x_\phi(t)$ \hspace{1em} artificial neural network approximation of the training function

$x_s(t)$ \hspace{1em} SNN interpolation of the training function $x(t)$

$x_h(t)$ \hspace{1em} HNN interpolation of the training function $x(t)$

$x_\Theta(t)$ \hspace{1em} FNN interpolation of the training function $x(t)$

$y(t)$ \hspace{1em} training function

$y_h(t)$ \hspace{1em} HNN interpolation of the training function $y(t)$

$y_\Theta(t)$ \hspace{1em} FNN interpolation of the training function $y(t)$

$\omega$ \hspace{1em} frequency, radians per second.

$\alpha$ \hspace{1em} feedback constant of the gradient descent algorithm; scale factor of Hermite-Rodriguez function; Gamma function parameter.

$\beta$ \hspace{1em} scale factor of Hermite-Rodriguez function; Gamma function parameter.

$\delta(t)$ \hspace{1em} delta function

$\Delta t$ \hspace{1em} interval between samples

$\Delta T$ \hspace{1em} interval between network centres

$\varepsilon$ \hspace{1em} accuracy of neural network

$\phi(t)$ \hspace{1em} activation function of neural network; can be a sigmoid function, radial basis function or orthogonal function.

$\gamma$ \hspace{1em} scale factor of Hermite-Rodriguez function or associated Hermite function.

$\eta$ \hspace{1em} mean value

$\lambda$ \hspace{1em} width parameter weight of sigmoid neural network
\( \lambda_j \) \hspace{1cm} j \text{ th width parameter weight of sigmoid neural network}

\( \theta(t) \) \hspace{1cm} \text{orthonormal Fourier function}

\( \Theta(t) \) \hspace{1cm} \text{complex orthonormal Fourier function}

\( \sigma \) \hspace{1cm} \text{standard deviation of Gaussian function}

\( \psi(x) \) \hspace{1cm} \text{wave function of quantum mechanics}
1.1 MOTIVATION, CONTRIBUTION AND METHOD

The chief method of computing a correlation based filter and inverse filter is to use the Fourier orthogonal series. This has remained so despite the introduction of new technologies such as the wavelet transform. The reason for its popularity is that it is very efficient in performing these types of computations. To calculate the correlation directly with another type of orthogonal series would generally require a summation of $N^2$ terms, but the Fourier series does it with a summation of only $N$ terms.

The correlation of other types of orthogonal functions is less well known. It is of interest to investigate the correlation of these functions to determine whether there are any possible benefits. The particular orthogonal functions investigated in this thesis are the Hermite functions. The reason for choosing the Hermite functions is that they are based on the Gaussian function, which is important in filtering and inverse filtering applications in engineering. To the best knowledge of the author, the correlation of two arbitrary functions using the Hermite functions has not yet been investigated anywhere.

The method chosen in this thesis to investigate the Hermite series correlation is with artificial neural network technology. The application of artificial neural network technology to orthogonal functions is a recent development, following the introduction
of the sigmoid neural network. In these applications the sigmoid activation function of
the neural network was replaced with an orthogonal activation function. A key
component of the artificial neural network technology is the gradient descent algorithm,
which is applied throughout this thesis to simplify the numerical calculation of the
Hermite and Laguerre series weights.

1.2 THESIS OUTLINE

The correlation of two functions $x(t)$ and $y(t)$ is defined as the integration, $r(t)$, given
by:

$$
r(t) = x(t) \star y(t) = \int_{-\infty}^{+\infty} x^*(\tau)y(\tau + t) d\tau
$$

(1.1)

where $x^*(t)$ is the complex conjugate of $x(t)$. An alternative equivalent form is

$$
r(t) = x(t) \star y(t) = \int_{-\infty}^{+\infty} x^*(\tau - t)y(\tau) d\tau
$$

(1.2)

Correlation can be computed by expanding $x(t)$ and $y(t)$ as Hermite neural networks.
The Hermite neural network interpolation of $x(t)$ and $y(t)$ is (Chapter 3):

$$
x(t) \equiv \sum_{n=0}^{N} a_n h_n(t)
$$

(1.3)

and

$$
y(t) \equiv \sum_{n=0}^{N} b_n h_n(t)
$$

(1.4)
The weights \( a_k \) and \( b_k \), which minimize the mean square error of the interpolation, were initially guessed using Monte Carlo integration of \( x(t) \) and \( y(t) \) using the discrete data \( \{t_i, x(t_i), y(t_i) : i = 0, 1, 2, ..., I\} \) on the interval \( \{-T/2 \leq t \leq T/2\} \). This initial guess is

\[
a_k \approx \frac{T}{I} \sum_{i=0}^{I} x(t_i) h_k(t_i) \tag{1.5}
\]

and

\[
b_k \approx \frac{T}{I} \sum_{i=0}^{I} y(t_i) h_k(t_i) \tag{1.5}
\]

The optimum weights were then found by successive application of the gradient descent algorithm as

\[
a_k(t_{i+1}) = a_k(t_i) + \alpha \left\{ \sum_{n=0}^{N} a_n(t_i) h_n(t_i) \right\} a_k(t_i) \tag{1.7}
\]

and

\[
b_k(t_{i+1}) = b_k(t_i) + \alpha \left\{ \sum_{n=0}^{N} b_n(t_i) h_n(t_i) \right\} b_k(t_i) \tag{1.8}
\]

where \( \alpha \) is the learning rate of the algorithm.

Using these network weights, the correlation of \( a(t) \) and \( b(t) \) is given as a summation of correlated Hermite functions by:

\[
x(t) * y(t) \approx \sum_{m=0}^{N} \sum_{n=0}^{N} a_m b_n \{ h_m(t) * h_n(t) \} \tag{1.9}
\]

The correlation of two Hermite functions is an associated Laguerre function (Chapter 4). That is
Therefore the correlation of two Hermite neural networks becomes an associated Laguerre neural network containing $N \times N$ terms:

$$r(t) \equiv \sum_{m=0}^{N} \sum_{n=m}^{N} (a_m b_m (-1)^{n+m} \delta_{n,m} + a_n b_n) l_m^{n-m}(t^2/2) \quad \text{for} \quad t \geq 0$$

(1.11)

where

$$\delta_{n,m} = \begin{cases} 1 & \text{for} \quad n \neq m \\ 0 & \text{for} \quad n = m \end{cases}$$

(1.12)

with a similar summation for $t < 0$.

Due to the $N \times N$ terms required by the Hermite correlation, the correlation of $x(t)$ and $y(t)$ will generally be more efficiently calculated with a Fourier neural network, which only requires $N$ terms. However if either $x(t)$ or $y(t)$ is the Gaussian function then the Hermite correlation is also a summation of only $N$ terms, which is competitive with the Fourier correlation.

**1.3 CONTENTS OF THESIS**

Following the introduction, Chapters 2 and 3 of this thesis review neural network technology and introduce the Hermite orthogonal series respectively. Then in Chapters 4 to 8 the new research work conducted in this thesis is presented. To the best knowledge of the author, this research work has not been given before. The research work presented in these chapters has now been published in several international scientific journals by the author (see publication list for this thesis).
CHAPTER 1 INTRODUCTION

Chapter 2 is intended to provide a general review of artificial neural network technology. It describes the operation and mechanisms of neural networks with an emphasis on the application to the modeling of non-linear functions.

Chapter 3 describes the Hermite neural network. Hermite functions achieved fame as the solution of the Quantum mechanic harmonic oscillator. A fact exploited in this chapter to derive many of the properties of the Hermite neural network. A method of training the Hermite neural networks is proposed. Comparison with the Fourier neural network is made.

Chapter 4 describes the correlation of two Hermite neural networks given by equation (1.10). The correlation of two Hermite functions of the same order is a Laguerre function, which is given in Klauder’s [1960] paper. In this chapter this work is extended to show that the correlation of two Hermite functions of different order is an associated Laguerre function. This allows the computation of the correlation of two arbitrary functions expanded with the Hermite series to be accomplished, which has not been given before.

Associated Laguerre functions are related to the Laguerre functions by differentiation. A numerical scheme of generating the associated Laguerre functions is proposed. The Hermite correlation is compared and discussed in relation to the well-known Fourier correlation.

Chapter 5 describes the application of the correlation to the development of a new type of Gaussian filter and inverse filter using a Hermite neural network.

The name Hermite/Laguerre filter was coined to distinguish this new type of filter from all previous types of filter based on the Gaussian function. The noise filtering properties
of the Hermite/Laguerre filter are quantified by comparison with theory. A comparison is also made with other types of filters including the sigmoid neural network. Filtering of noisy signals is an area where the sigmoid neural network is still not fully developed despite considerable research effort.

In addition to the Gaussian filter it is shown in Chapter 5 how to synthesize different types of filters with the Hermite neural network. A fourth order Gaussian filter is demonstrated as an example.

Application of the Hermite/Laguerre correlation as a new type of Gaussian inverse filter is also described. A comparison is made with a Fourier series based inverse filter. It is shown that the Hermite/Laguerre series has a distinct advantage over the Fourier series in regard to numerical stability. The application of the Hermite/Laguerre inverse filter to the recovery of distorted laser Doppler anemometer measurement data is presented.

Chapter 6 analyses the errors associated with the Hermite/Laguerre filter using statistics. This is done by showing that the Hermite/Laguerre filter is equivalent to Gaussian kernel regression. A boundary error is shown to occur at the edges of the data. This is a well-known problem associated with kernel regression. To correct the boundary error, a new type of kernel regression with asymmetric functions is developed in this chapter.

Chapter 7 applies the Hermite neural network correlation to the processing of simulated radar signals. Several types of radar were investigated including Doppler and CHIRP radar. The aim of this chapter is to present further results in addition to those given in Chapter 5, which compare the new type of correlation with existing methods. Therefore in these applications the Hermite neural network was compared with a Fourier correlator and a correlator based on Hermite-Rodriguez functions.
Chapter 8 develops an improved training algorithm for the sigmoid neural network interpolation of a non-linear function. This allows for the future development of a hybrid neural network consisting of a Hermite neural network with a sigmoid neural network. The improved algorithm for the sigmoid neural network is based on training with respect to the rate of change of the non-linear function rather than to the function itself.

Chapter 9 summarizes the main findings of this thesis and suggests further research work.
CHAPTER 2

ARTIFICIAL NEURAL NETWORKS

2.1 INTRODUCTION

This chapter reviews neural network technology, particularly those aspects of the technology that are relevant to this thesis.

In Section 2.2 the history of neural networks is briefly reviewed. Section 2.3 describes the operation of the biological neural network, which inspired many of the current networks. Section 2.4 explains the mathematical model of the biological neuron. In this section it is explained how to train neural networks with logistic activation functions. The topic of training is extended to the sigmoid activation functions in Section 2.5. Radial basis neural networks and the closely related kernel regression are covered in Section 2.6 and 2.7 respectively. Further details on kernel regression are given in Chapter 6. Orthonormal activation functions play an important role in this thesis. The operation of these types of neural networks and their important properties are given in Section 2.8. Although not applied in this thesis, for completeness multi-layer neural networks are described in Section 2.9.

2.2 A BRIEF HISTORICAL SUMMARY

Artificial neural networks originated from research work on the operation and
mechanism of biological nervous systems, particularly the brain. The first mathematical model of a neuron was described by McCulloch and Pitts in their 1943 paper “A logical calculus of ideas imminent in nervous activity”. This work not only led to the development of artificial neural networks but it is also credited with inspiring the digital computer and expert systems.

Ten years later Rosenblatt applied a modified McCulloch-Pitt neural network model to visual perception. The name “perceptrons” is derived from this application. A feature of the perceptron was that the connections between neurons could be adjusted so that it could be trained to model different patterns.

In 1960 Widrow and Hoff proposed a neuron called “adaline” an abbreviation taken from the full description ADAptive LINEar neuron. Adaline was similar to Rosenblatt’s perceptron but differed in the training algorithm. An extension to a network composed of many Adelines was called “madaline”, which stands for Multiple Adeline [Widrow, 1990].

Following the publication of the book “Perceptrons” by Minsky and Papert in 1969, work on neural networks entered a period of stagnation. In their book, Minsky and Papert showed that the perceptron could not distinguish between certain simple patterns. This led to a lack of funding in neural network research. It was not until 1985 that training procedures designed to overcome this problem were devised.

Renewed interest in neural networks occurred after the 1982 paper by John Hopfield. In this paper he described a neural network, which was based on research undertaken on the olfactory system of a garden slug. The increased attention generated by Hopfiled’s work exploded in the late 1980’s when it was realized that neural networks have many
important characteristics. The important characteristics of neural networks, which have led to this renewed interest, are:

- Learning and adaptation
- Generalization
- Massive parallelism
- Robustness
- Associative storage of information
- Spatio-temporal information processing

Unfortunately, not all is rosy. In practice, neural networks are often implemented with software where parallel operation is not possible. These neural networks suffer from very slow training, a serious disadvantage in many applications.

2.3 BIOLOGICALLY INSPIRED ARTIFICIAL NEURAL NETWORKS

[NEURALWARE, 1993]

A direct analogy with the biological system is a useful aid in describing the principles and properties of artificial neural networks. Much of the terminology of artificial neural networks is taken directly from the biological system.

The biological nervous system consists of billions of individual processing units, called neurons (Figure 2.1), densely interconnected to each other. Each neuron behaves like a simple micro-processing unit. Input signals enter the neuron through channels called dendrites and the processed output signal leaves through a channel called axon. The neuron sums all of the input signals and produces an output signal if the combined signal is strong enough.
Figure 2.1 Schematic diagram of a biological neuron (Gupta and Rao)

Figure 2.2 Mathematical model of biological neuron.
The junction between the axon output of one neuron and the input dendrite of another neuron is called a synapse. The amount of signal transmitted across the junction is a variable, called the synaptic efficiency, which is modified when the brain learns.

In an artificial neural network the neuron is called a processing element (Figure 2.2); the inputs are also called dendrites. The processing element sums all the inputs and the combined input is transferred to the output. The magnitude of the combined input that reaches the output is determined by a transfer function. Each input to the artificial neuron is assigned a weight, which corresponds to the synaptic strength. The weight determines the relative importance of the input. The assignment of weights to each dendrite of the neural network is called training.

By itself, a single neuron is not very useful, but a large number of neurons interconnected to form a network, enables complex problems to be solved. The potential benefits of such a network are summarized as follows.

Neural networks have many elements connected in a parallel structure. Due to this parallel structure, the failure of a few neurons does not significantly affect the overall performance. They are capable of adaptation in response to changes in the environment allowing ill-defined and imprecise data to be successfully modeled. This is also beneficial to adaptive control systems.

In modeling applications, neural networks can approximate a non-linear function to the desired degree of accuracy and can easily be configured to have as many inputs and as many outputs as one desires. Multi-layer neural networks are able to perform a multi-resolution approximation of a non-linear function similar to that achieved with a wavelet expansion.
Figure 2.3 Schematic layout of a single input/output, single layer neural network.

Figure 2.4 Detail of the $n^\text{th}$ node of the neural network of Figure 2.3.
2.4 SINGLE LAYER NEURAL NETWORKS FOR FUNCTION APPROXIMATION

Many practical problems can be solved with neural networks; among these is the modeling of non-linear functions. The simplest network capable of modeling a non-linear function of one dimension is shown in Figure 2.3. It consists of a single input-processing element followed by a single row or layer of processing elements called the hidden layer, and a single output-processing element. The input signal corresponds to the independent variable of the non-linear function; the output signal is the neural network approximation of the non-linear function. The input-processing element splits the input signal into a multitude of dendrites, each with its own individual weight, for distribution to the hidden layer. The output-processing element collects and sums the individual outputs from each of the hidden layer processing elements. The operation of a single processing element of the hidden layer, the $n^{th}$ element, is shown in detail in Figure 2.4.

In order to model non-linear functions, additional fixed inputs are generally required at the input and output of the network. These inputs bias the transfer functions of each of the processing elements of the hidden layer and also provide the output with a constant bias term. Without these inputs the network will not converge to a solution.

Mathematically the single layer network is described as

$$y_\phi(t) = \sum_{n=0}^{N} v_n \phi (u_n t - b_n) + c$$

(2.1)

$u_n$ are the weights on the input dendrites to each processing element of the hidden layer; $v_n$ are the weights on the output dendrites of each processing element of the
hidden layer; $b_n$ are the weights of the input bias; $c$ is the constant output bias weight; and the activation function is $\phi(u_n t - b_n)$.

The activation functions of the early neural networks, such as those of Rosenblatt [1959] and Widrow and Hoff [1960], were logistic functions. Later Hopfield [1982] replaced the logistic function with the sigmoid function. To improve the training speed of software implemented neural networks, Gaussian activation functions were developed [Moody and Darken, 1989]. Networks with sigmoid functions and Gaussian functions are now the main type of neural networks encountered.

A convenient way of classifying neural networks is based on the type of activation function (Figure 2.5). Activation functions can be divided into two broad classes: local and global. Local functions only respond across a very short range of the input variable. Examples of local functions are the Gaussian function and the rectangular function, Figure 2.6(a) and 2.6(b) respectively. Global activation functions respond across the full range of the input variable. Typical global functions include sigmoid functions such as the unit step function and logistic function, and orthonormal functions. Figures 2.6(c) and 2.6 (d) show the rectangular and logistic function respectively.

To apply a neural network to the modeling of a non-linear function, the weights of the network must be determined. The simplest types of neural networks for function approximation have activation functions of the local rectangular and global unit step type. The weights of these networks are particularly easy to obtain. Figure 2.7 shows the modeling of a function, $y(t)$, using local rectangular activation functions, $u_n = 1$. 
Figure 2.5 Classification of neural networks based on the type of activation function.

Figure 2.6(a) Rectangular activation function (local).

Figure 2.6(b) Gaussian activation function (local).

Figure 2.6(c) Unit-step activation function (global).

Figure 2.6(d) Logic sigmoid activation function (global).
Figure 2.7 Neural network approximation of $y(t)$ using rectangular local activation functions.

Figure 2.8 Neural network approximation of $y(t)$ using unit-step global activation functions.
The activation function is

\[
\phi(t) = \begin{cases} 
1 & \text{for } -\Delta t/2 \leq t \leq \Delta t/2 \\
0 & \text{for } t > \Delta t/2 \\
0 & \text{for } t < -\Delta t/2 
\end{cases}
\]  

(2.2)

where $\Delta t$ is the width of the rectangular function. By inspection, the weights of the network are given by

\[ b_n = t_n \]  

(2.3)

and

\[ v_n = y(t_n) \]  

(2.4)

Figure 2.8 shows the modeling of a function, $y(t)$, using global unit step activation functions. The activation function is

\[
\phi(t) = \begin{cases} 
1 & \text{for } t \geq 0 \\
0 & \text{for } t < 0 
\end{cases}
\]  

(2.5)

The weights of the network are

\[ b_n = t_n \]  

(2.6)

and

\[ v_n = y(t_n) - y(t_{n-1}) \]  

(2.7)

The determination of the network weights is generally considerably more complicated than the simple networks with rectangular and unit step activation functions would suggest.
The determination of the network weights for more general types of neural networks is considered in the next two sections.

### 2.5 SIGMOID NEURAL NETWORKS AND NETWORK TRAINING

A particular class of global activation function is the sigmoid function. A function is defined as being a sigmoid if it satisfies the conditions

\[
\lim_{t \to -\infty} \phi(t) = 0 \quad (2.9)
\]

and

\[
\lim_{t \to \infty} \phi(t) = 1 \quad (2.8)
\]

The unit-step sigmoid function has already been described. Although simple to apply, it produces a discontinuous, bumpy and irregular interpolation of a function. In order to obtain a smoother interpolation, the logistic sigmoid function is in widespread use; this function (Figure 2.6(d)) is defined as

\[
\phi(t) = \frac{1}{1 + e^{-t}} \quad (2.10)
\]

The penalty for the smoother interpolation is that it is difficult to find the network weights.

“Training”, is the name that has been adopted to describe the procedure of finding the weights of a neural network. The early neural networks of Widrow [1990] had training algorithms based on the least mean square algorithm, which is suitable for the logistic function. When the sigmoid function became the main type of activation function, the least mean square algorithm was not suitable. In these types of neural networks training
is usually accomplished using the gradient descent algorithm [Widrow, 1990]. The term “back-propagation” is often associated with neural network training, particularly with the gradient descent algorithm [Widrow, 1990]. Back-propagation comes from the application of the gradient descent algorithm to multi-layer neural networks. In a multi-layer network the weights are updated at the output of the network first then to the preceding layers in a backward direction, hence the name back-propagation.

Training adjusts the weights of a neural network to those values that minimize the error between a data set and the neural network approximation. In order to train the neural network the error must first be defined. In widespread use is the mean square error ($mse$), which is defined for the training data set $\{t_i, y(t_i) : i = 0,1,2,\cdots, I - 1\}$, as

$$mse = \frac{1}{I} \sum_{i=0}^{I-1} (y(t_i) - y_\phi(t_i))^2 .$$

(2.11)

The principal algorithm for training the logistic sigmoid neural network is the gradient descent algorithm [Widrow, 1990]. Gradient descent minimizes the $mse$ by changing the weights in a series of steps; with each step the weights are changed according to the gradient of the error function with respect to the weight. A zero gradient with respect to the weights occurs at the optimum value of the weights. If $w_n \equiv u_n, v_n, b_n, c$, then gradient descent updates the weights according to

$$w_n(t_{i+1}) = w_n(t_i) - \alpha \frac{\partial e(t_i)}{\partial w_n} .$$

(2.12)

where $e(t_i)$ is the on-line error defined as

$$e(t_i) = (y(t_i) - y_\phi(t_i))^2 .$$

(2.13)
From Equations (2.1), (2.12) and (2.13) the appropriate mathematical equations for gradient descent at time step $t_i$, are given by

$$u_n(t_{i+1}) = u_n(t_i) + \alpha \{y(t_i) - y_\phi(t_i)\} \frac{\partial y_\phi(t_i)}{\partial u_n}$$  \hspace{1cm} (2.14)$$

$$v_n(t_{i+1}) = v_n(t_i) + \alpha \{y(t_i) - y_\phi(t_i)\} \frac{\partial y_\phi(t_i)}{\partial v_n}$$  \hspace{1cm} (2.15)$$

$$b_n(t_{i+1}) = b_n(t_i) + \alpha \{y(t_i) - y_\phi(t_i)\} \frac{\partial y_\phi(t_i)}{\partial b_n}$$  \hspace{1cm} (2.16)$$

$$c(t_{i+1}) = c(t_i) + \alpha \{y(t_i) - y_\phi(t_i)\} \frac{\partial y_\phi(t_i)}{\partial c}$$  \hspace{1cm} (2.17)$$

$\alpha$ is a constant which controls the rate at which the weights converge to a solution. The weights are updated for each individual pair of training data $\{t_i, y(t_i)\}$ from $i = 0$ to $I$. Repetition of the training data is required until a satisfactory solution is found.

Evaluating the partial derivatives leads to the following set of equations

$$u_n(t_{i+1}) = u_n(t_i) + \alpha \{y(t_i) - y_\phi(t_i)\} y_\phi'(t_i) t_i$$  \hspace{1cm} (2.18)$$

$$v_n(t_{i+1}) = v_n(t_i) + \alpha \{y(t_i) - y_\phi(t_i)\} \phi(u_n t_i - b_n)$$  \hspace{1cm} (2.19)$$

$$b_n(t_{i+1}) = b_n(t_i) - \alpha \{y(t_i) - y_\phi(t_i)\} y_\phi'(t_i)$$  \hspace{1cm} (2.20)$$

$$c(t_{i+1}) = c(t_i) + \alpha \{y(t_i) - y_\phi(t_i)\}$$  \hspace{1cm} (2.21)$$

where the derivative of the sigmoid neural network interpolation, $y_\phi'(t)$, is
In this equation, the derivative of the logistic sigmoid function, has been conveniently expressed as

\[ \phi'(t) = \frac{e^{-t}}{1 + e^{-t}} = \phi(t) \cdot (1 - \phi(t)) \]  

(2.23)

The simple form of the gradient descent algorithm given by Equation (2.19) and (2.20), which occurs when the gradient of the error is independent of the derivative of the activation function, is referred to as the least means square algorithm.

The gradient descent algorithm requires an initial guess of the solution. In view of the simple solution for the unit-step sigmoid functions, it would seem appropriate to use the solution given by Equations (2.6) and (2.7) as an initial guess. However in practice this does not lead to good results. This is because the curvature of the logistic sigmoid function allows a good interpolation to be achieved with much fewer activation functions than with the unit-step function. As far as the author is aware, a good initial guess to the optimum weights of the logistic sigmoid neural network is not known. The initial guess usually adopted is to randomly choose the weights.

The lack of a close approximation to the optimum weights of the logistic sigmoid neural network means that many iterations of the gradient descent algorithm must be undertaken to reach the optimum solution. This leads to very slow training. A further problem is that it is possible for the \(mse\) of a logistic sigmoid neural network to have more than one location in which the gradient is zero which can cause the algorithm to finish early whilst the \(mse\) is still quite large. Unfortunately a fast and reliable training
algorithm for the sigmoid neural networks has not yet been found. For practical
application it is useful to know what types of functions can actually be approximated by
the sigmoid neural network. Cybenko [1989] proved mathematically that a neural
network summation of sigmoid functions, \( y_\phi(t) \), given by Equation (2.1) exists which
allows any function, \( y(t) \), to be approximated to a degree of accuracy, \( \varepsilon \), defined as

\[
|y_\phi(t) - y(t)| < \varepsilon
\]  

(2.24)

Unless stated otherwise, in this thesis the term sigmoid neural network will be taken to
mean a network with the logistic type of activation functions described by Equation
(2.10).

2.6 RADIAL BASIS NEURAL NETWORKS

The radial basis neural network was introduced by Moody and Darken [1988], and
Broomhead and Lowe [1989]. Radial basis neural networks have activation functions of
the local type. The most commonly encountered activation function of a radial basis
neural network is the Gaussian function defined as

\[
\phi(t) = r(t) = \exp(-t^2)
\]  

(2.25)

Gaussian radial basis neural networks are rarely written in the general form given by
Equation (2.1). They are more often written in full as

\[
y_\phi(t) = \sum_{i=0}^{N} a_i \exp\left(-\frac{(t-b_i)^2}{\sigma_i^2}\right)
\]  

(2.26)

\( b_i \) is the centre of the Gaussian processing element; \( \sigma_i \) is a scaling parameter; \( a_i \) is the
amplitude of the Gaussian function. The scaling parameter determines over what
distance the processing element will have a significant effect.

Although all of the weights of the radial basis neural network can be trained with gradient descent, faster training algorithms have been developed which are more popular. A common method [Leonard and Kramer, 1991] combines the k-means clustering algorithm for the selection of the centres with a P-nearest neighbor heuristic for choosing the appropriate scaling factors and gradient descent for determining the amplitude weights. An ordered approach is required; first the centres are determined and then the scaling factors are chosen; finally the amplitude weights are solved.

k-means clustering determines the location of the processing element centres by finding a set of k cluster centres such that the sum of all the distances between each element of the training data, \( t_i \), and its closest cluster centre, \( b_n \), is a minimum. The adaptive k-means clustering algorithm begins with a random selection for the cluster centres. The training data, is input sequentially and the cluster centres are updated by changing only the cluster centre, \( b_n \), which is closest to the training data, \( t_i \), by the gradient descent step

\[
b_n(t_{i+1}) = b_n(t_i) + \alpha(t_i - b_n(t_i))
\]  

\( \alpha \) is a learning rate coefficient.

The aim of the procedure to determine the scaling factors, \( \sigma_n \), is to fit the network output smoothly to the training data. If the scaling factors are too small then a bumpy output will be obtained and the network is said to over fit the training data. Conversely if the scaling factors are too large then the neural network excessively dampens the training data resulting in under fit of the training data. The P-nearest neighbor heuristic
selects the scaling factor $\sigma_n$ for the Gaussian function with centre, $b_n$, from the average mean square distance of the nearest neighbors

$$\sigma_n = \frac{1}{P} \sum_{m=1}^{P} (b_m - b_n)^2$$  \hspace{1cm} (2.28)$$

where $b_1$, $b_2$, ..., $b_P$ are the P-nearest neighbors of $\sigma_n$. A suggested value for the integer, $P$, is 2.

After the centers and scaling factors have been determined, the amplitude weights, $a_n$, are solved with gradient descent, which has the Least Means Square (LMS) form of

$$a_n(t_{i+1}) = a_n(t_i) + \alpha \left[ y(t_i) - y_p(t_i) \right] \exp \left( -\frac{(t_i - b_n)^2}{(\sigma_n)^2} \right)$$  \hspace{1cm} (2.29)$$

In the last decade, radial basis neural networks and sigmoid neural networks have emerged as the two principle types of neural network. Their properties are generally complementary to each other. Compared to the sigmoid neural network, radial basis neural networks train more quickly, but have a slower response on recall. Their interpolation of non-linear functions is not as smooth as that obtained with a sigmoid neural network and is often more bumpy in comparison. Slow recall response is due to the larger number of activation functions required to obtain a smooth approximation. The fast training is due to the localized response of the activation function combined with the simpler training of the least means square algorithm.

### 2.7 KERNEL REGRESSION

In their introductory paper on the radial basis neural network, Moody and Darken [1989] seem to be the first to associate the Radial Basis Neural Networks (RBNN) with Kernel regression. The distinction between each is not clearly defined, but kernel
regression has a statistical foundation whereas RBNNs are a natural development of the sigmoid neural network. Due to the close relationship with kernel regression, many of the important results originally developed for kernel regression can be directly applied to the RBNN and visa-versa.

Kernel regression was derived independently in 1964 by Nadaraya and Watson with a mathematical foundation from Parzen’s [1962] earlier work on kernel density estimation. Wand and Jones [1995] have given a thorough, extensive review of the development of kernel density estimation since its invention. In 1991 Specht and Zaknich et al independently introduced it to the neural network community. Zaknich [1998] provides a comparison with the sigmoid neural network as well as other types of filters.

An extensive amount of research has been undertaken in the field of statistics on kernel regression. A number of different types of kernel regression have been developed. Among these, probably the easiest to understand is the Priestley and Chao [1972] kernel regression. The Priestley and Chao [1972] regression approximates the convolution integral of the moving average filter by the Monte-Carlo integration.

\[
y_k(t) = \int_{-\infty}^{+\infty} y(\tau) k(t - \tau) d\tau \approx \frac{T}{I} \sum_{i=0}^{I-l} y(t_i) k(t - t_i)
\]  

(2.30)

where the training data \( \{t_i, y(t_i); i = 0, 1, 2, \ldots, I - l\} \) on the interval \( \{-T/2 \leq t \leq T/2\} \) may be ordered or random. For equally spaced, ordered training data the Monte-Carlo integration is equivalent to a numerical integration.

The function \( k(t) \) is the kernel function, which is traditionally chosen from a wide variety of symmetric functions [Parzen, 1962]; the Gaussian function is a popular
choice. The kernel regression averages all the data contained within the range of the kernel function centered at \( t = t_i \). The effective range of the kernel function is called its window width.

Another popular type of kernel regression is the Narayada [1964] and Watson [1964] kernel regression. This estimates \( y(t) \) with the estimator \( y_k(t) \) given by

\[
y_k(t) = \frac{\sum_{i=0}^{l-1} y_i k(t-t_i)}{\sum_{i=0}^{l-1} k(t-t_i)}
\]  

(2.31)

Statistical analysis of the Narayada/Watson kernel regression is difficult because it is defined as the ratio of two random variables [Hall, 1984]. The Priestley and Chao [1972] kernel regression does not suffer from this problem. For this reason, only the Priestley and Chao [1972] type of regression is considered in this thesis.

### 2.8 ORTHONORMAL NEURAL NETWORKS

If \( \phi_n(t) \) and \( \phi_m(t) \) are any two functions from a set of functions \( \{\phi_0(t), \phi_1(t), \ldots, \phi_N(t)\} \) defined on an interval \( \{t, t\} \), then the set of functions are defined as being orthogonal if they satisfy the condition

\[
\int_{t}^{t} \phi_n(t) \phi_m(t) dt = 0 \quad \text{when} \quad m \neq n
\]  

(2.32)

They are said to be orthonormal if they satisfy the additional condition that

\[
\int_{t}^{t} \phi_n(t) \phi_m(t) dt = 1 \quad \text{when} \quad m = n
\]  

(2.33)
The application of neural network technology to Orthogonal functions was first proposed in the early nineties [Osowski, 1992], [Cichocki and Lobos, 1992], [Zhu and Paul, 1994], [Ulug, 1992]. In an orthogonal neural network the hidden layer of processing elements consists of a set of functions, which form an orthogonal basis.

The approximation of a function by an orthogonal series is not new and dates back to the early days of mathematics. However, orthogonal series benefit from the application of neural network technology. This is nicely illustrated in the research work of Osowski [1992]. Osowski [1992] simulated an electronic neural network consisting of Fourier functions, with which he estimated the Harmonic components in a power system. The advantage of the neural network, which was demonstrated in the simulation, was speed. In fact it allowed a solution to be obtained in real time. The high speed was achieved using a parallel structure of neural network components.

Zhu and Paul [1994], Pelagotti and Piuri [1997], Ulug [1992] have also demonstrated neural networks with Fourier activation functions. These applications were all software neural networks. The main theme of these papers was the speed advantage offered by Fourier activation functions compared to the sigmoid function.

The generalization of neural network technology to other types of orthogonal activation functions followed shortly after the application to the Fourier series. Yang and Tseng [1996] proposed a neural network of Legendre functions and more recently Lee and Jeng [1998] investigated the Tchebychev neural network.

An important point mentioned by Yang and Tseng [1996] was that the orthogonal activation functions could be generated recursively, increasing the response time of the network.
The sigmoid neural network has several advantages that have so far not been mentioned in the neural network literature. The advantage of a sigmoid neural network is that it is able to approximate a wide variety of functions very well using much fewer processing elements than an orthogonal neural network. Also sigmoid neural networks do not suffer from the ringing effect known as the Gibbs phenomena that occurs in the approximation of functions, which have a discontinuity.

Practical applications of orthogonal networks are varied; Jeng et al [1998] applied a Tchebychev orthogonal neural network to the control of a magnetic bearing; Shulka et al [1999] demonstrated an orthonormal neural network for the control of dry or static friction dynamical systems.

An arbitrary function, \( y(t) \), defined on the interval \( \{t_1, t_2\} \), can be represented as a series consisting of the orthonormal functions as the network

\[
y(t) \equiv y_n(t) = \sum_{n=0}^{N} a_n \phi_n(t)
\]  

(2.34)

This is equivalent in form to the general definition of a single layer neural network given by Equation (2.1) of Section 2.4; the input weights, \( u_n \), are all equal to one and the output weights, \( v_n \), correspond to the coefficients \( a_n \). Each processing element has a different activation function corresponding to a particular function from the orthogonal set of functions.

One of the advantages of the Fourier orthogonal networks is that they also form an orthogonal basis for uniformly distributed discrete data, that is

\[
\sum_{i} \phi_n(t_i) \phi_m(t_i) = \begin{cases} 1 & m = n \\ 0 & m \neq n \end{cases}
\]  

(2.35)
Orthogonal neural networks are trained with the same gradient decent algorithm described for the sigmoid neural network of Section 2.4. However for the idealized case of continuous data a unique, optimum solution exists for the weights of an orthogonal neural network. This solution can be derived from the mean square error for continuous data. The procedure is illustrated for the particular case of an orthonormal neural network. A continuous definition of the mean square error is defined in an analogous way to the discrete mean square error of Equation (2.11). The continuous mean square error, $mse$, is

$$mse = \int_{t_i}^{t_f} \left( y(t) - y_\phi(t) \right)^2 dt$$  \hspace{1cm} (2.36)$$

where we have used the same name as for the discrete $mse$; the form of the $mse$ will be implied by the application. For the orthonormal neural network

$$mse = \int_{t_i}^{t_f} \left( y(t) - \sum_{n=0}^{N} a_n \phi_n(t) \right)^2 dt$$  \hspace{1cm} (2.37)$$

The optimum weights for the approximation of a continuous function by an orthonormal neural network are obtained by differentiating the $mse$ with respect to the weight, $a_m$, to obtain the gradient

$$\frac{\partial mse}{\partial a_m} = -2 \int_{t_i}^{t_f} \left( y(t) - \sum_{n=0}^{N} a_n \phi_n(t) \right) \phi_m(t) dt$$  \hspace{1cm} (2.38)$$
Then using the definition of orthonormal functions of Equation (2.32) and (2.33) this simplifies to

\[ \frac{\partial \text{mse}}{\partial a_m} = -2 \cdot \int_{t_i}^{t_f} y(t)\phi_m(t) \, dt + 2a_m \] (2.39)

At the location of the minimum \( \text{mse} \), the gradient \( \frac{\partial \text{mse}}{\partial a_m} = 0 \), from which we obtain the optimum weight

\[ a_m = \int_{t_i}^{t_f} y(t)\phi_m(t) \, dt \] (2.40)

To prove that this is the only optimum solution, consider an alternative approximation of the function by a different set of weights, \( b_n \), such that

\[ y(t) \equiv y_{b_n}(t) = \sum_{n=0}^{N} b_n \phi_n(t) \] (2.41)

The \( \text{mse} \) for this series with the weights, \( b_n \), is by definition

\[ \text{mse}(b_n) = \int_{t_i}^{t_f} \left( y(t) - \sum_{n=0}^{N} b_n \phi_n(t) \right)^2 \, dt \] (2.42)

Expanding the square and using Equation (2.40) we obtain the expression

\[ \text{mse}(b_n) = \int_{t_i}^{t_f} \{y(t)\}^2 \, dt - 2 \sum_{n=0}^{N} a_n b_n + \sum_{n=0}^{N} \{b_n\}^2 \] (2.43)

The \( \text{mse} \) for the series with the weights, \( a_n \), is from Equation (2.37) and (2.40)

\[ \text{mse}(a_n) = \int_{t_i}^{t_f} \{y(t)\}^2 \, dt - \sum_{n=0}^{N} \{a_n\}^2 \] (2.44)
Subtracting Equation (2.44) from (2.43) and collecting terms gives the result

\[ mse(b_n) - mse(a_n) = \sum_{n=0}^{N} (a_n - b_n)^2 \]  

(2.45)

Since the right hand side of this equation is the sum of a series of positive numbers, which is always greater than zero, it follows that the weights \( a_n \) are the only optimum \( mse \) weights.

In practical problems in which the neural network is applied to, the function, \( y(t) \), is not available as a continuous function but is instead defined as a discrete set of data, \( \{ t_i, y(t_i) : i = 0, 1, 2, \ldots, I \} \). The weights \( a_n \) which minimize the \( mse \) for the continuous function may be slightly different from those which are optimum for the discrete function, denoted by \( \tilde{a}_n \). This is especially true if the data is corrupted by noise. The optimum weights for the discrete function are then found using the gradient descent algorithm, which is for the orthogonal functions

\[ \tilde{a}_m(t_{i+1}) = \tilde{a}_m(t_i) + \alpha [y(t_i) - y(t_i)\phi_m(t_i)] \]  

(2.46)

On its own, the gradient descent algorithm is slow. Considerably faster training can be achieved by first estimating the weights \( \tilde{a}_m \) by Equation (2.40)

\[ \tilde{a}_m \approx a_m = \int_{t_i}^{t_f} y(t)\phi_m(t) dt \]  

(2.47)

and using a numerical integration of the training data to approximate the integration. With this initial estimate, the weights converge in only a few iterations of the gradient descent algorithm, depending on the quality and extent of training data available.
2.9 MULTI-LAYER NEURAL NETWORKS

A single layer neural network is capable of modeling any function provided the number of processing elements is sufficiently large. However, a single layer neural network converges globally. In some applications, local, or point wise, convergence is required; these applications require multi-resolution systems. For example, it may be required to model an amplitude-modulated sine wave. A single layer network may satisfactorily model the slowly changing amplitude signal but may not have sufficient resolution to detect the fast changing, carrier frequency. As another example, a multi-resolution system can pick up better the sharp edge effects of a square wave than is possible with a fixed resolution system.

Multi-resolution capability can be achieved with a neural network by increasing the number of hidden layers. Note that multi-resolution capability is also achieved with wavelet systems.

Figure 2.9 and 2.10 show the notation adopted for a multi-layer neural network consisting of \( H \) hidden layers, each layer containing \( N \) neurons. The network is shown with multiple inputs and outputs for consistency in notation. The mathematical equations describing a multi-layer neural network are written as follows. At the \( m^{th} \) output neuron of the network

\[
y_m^{[H]} = \phi \left( \sum_{n=0}^{N} w_{mn} y_n^{[H-1]} \right)
\]  

(2.48)

For an arbitrary layer \( K \), the output signal from the \( m^{th} \) neuron is the forward propagation of the output signal from the previous layer given by
Figure 2.9 Schematic layout of a multi-layer artificial neural network.

Figure 2.10 Detail of the $m^{th}$ node of the neural network of Figure 2.9.
The sum of all of the inputs to the $m^{th}$ neuron at layer $K$ is denoted by

$$S_m^{[k]} = \sum_{n=0}^{N} w_m[n] y_n^{[k-1]}$$

(2.50)

Due to the many weights, which are hidden within the nested output of the multi-layer network, the application of the gradient descent algorithm is considerably more complex and cumbersome than with the single layer network. A systematic procedure of solving these weights using the gradient descent algorithm in an ordered manner is referred to as back propagation. This procedure was originally proposed by Werbos in his 1971 PhD thesis and also by Parker [1975], but it wasn’t until the much later publication of Rosenblat [1985] that it gained popularity.

If $y_m$ is the desired signal at the output of the network then the error, $E$, is

$$E = \sum_{n=0}^{N} (y_n - y_m^{[k]})^2$$

(2.51)

In application of the gradient decent algorithm, the partial derivatives of $E$ with respect to $w_m^{[k]}$ are required. These are

$$\frac{\partial E}{\partial w_m^{[k]}} = \frac{\partial E}{\partial S_m^{[k]}} \frac{\partial S_m^{[k]}}{\partial w_m^{[k]}} = \frac{\partial E}{\partial S_m^{[k]}} y_n^{[k-1]}$$

(2.52)

Defining $e_m^{[k]}$ as

$$e_m^{[k]} = \frac{\partial E}{\partial S_m^{[k]}}$$

(2.53)
Then Equation (2.52) simplifies to

$$\frac{\partial E}{\partial w_{mn}^{[K]}} = e_m^{[K]} y_n^{[K-1]}$$  \hspace{1cm} (2.54)$$

It is apparent that $e_m^{[K]}$ is a measure of the error at layer $K$. It can be related to the error at the layer above by using the relationship between the inputs of each of these layers given by

$$S_m^{[K+1]} = \sum_{n=0}^{N} w_{mn}^{[K+1]} \phi(S_n^{[K]})$$  \hspace{1cm} (2.55)$$

It then follows that Equation (2.53) may be rewritten as

$$e_m^{[K]} = \sum_{n=0}^{N} \frac{\partial E}{\partial S_n^{[K+1]}} \frac{\partial S_n^{[K+1]}}{\partial S_m^{[K]}} = \phi'(S_m^{[K]}) \sum_{n=0}^{N} e_n^{[K+1]} w_{mn}^{[K+1]}$$  \hspace{1cm} (2.56)$$

Hence the error at layer $K$ is obtained by the back propagation of the error from the layer above.

2.10 CONCLUSION

This completes the brief review of artificial neural network technology. In Chapter 3 the Hermite orthonormal functions will be reviewed.
CHAPTER 3

HERMITE NEURAL NETWORK

3.1 INTRODUCTION

In order to develop a Hermite neural network for practical application, it is useful to relate the Hermite functions to a physical process. This allows the characteristics of these functions to be visualized and understood and the properties to be estimated. Hermite functions are best known as the solution of the harmonic oscillator problem of quantum mechanics. Therefore this chapter begins with the basic theory of quantum mechanics with an emphasis on those features of the theory that are directly applicable to the Hermite functions.

The Hermite neural network is then introduced and it is explained how to apply the network to practical problems. In particular the bandwidth and duration of the Hermite neural network must match the signal being interpolated. These properties of the Hermite neural network are derived from quantum mechanics. Although not required for this thesis, the associated Hermite functions are introduced in Section 3.7 for completeness. Associated Hermite functions are identical to the Hermite functions apart from the addition of a scaling factor to scale the ordinate into an appropriate range.

The procedure adopted in this thesis to train the Hermite neural network is described in Section 3.6.
A comparison with the Fourier neural network is made in Section 3.8, which provides an idea of the performance that can be expected with the Hermite neural network.

### 3.2 Quantum Physics Solution for Probability Density

Quantum physics is a mathematical theory, which describes the location and velocity of particle motion in terms of probability [Richtmyer et al, 1969, page 294-296]. It applies when the dimensions of the coordinates of motion of a particle approach microscopic dimensions.

In quantum physics, the probability of finding a particle at a location, \( x \), is described by a complex variable, \( \psi(x) \), called the wave function, from which the probability density is obtained by

\[
f(x) = \psi(x) \cdot \psi(x)^* \tag{3.1}
\]

where \( \psi(x)^* \) denotes the complex conjugate of \( \psi(x) \).

The wave function is the solution of a fundamental differential equation, defining probability density at the microscopic level, called the time independent Schrodinger equation [Schrodinger, 1926], in honor of E. Schrodinger who first discovered it. This equation is

\[
\psi''(x) = -\frac{m\eta^2}{2} \psi(x) (E - V(x)) \tag{3.2}
\]

where \( m \) is the particle mass, \( \eta \) is a constant called Planck’s constant, \( E \) is the particle energy and \( V(x) \) is the potential energy. Despite many attempts to derive the Schrodinger equation, see for example the classic papers by Feynman [1948] and
Consider the motion of an oscillating particle that obeys the conservation of energy law of classical physics. Then the velocity of the particle, \( v(x) \), satisfies the energy equation of classical physics

\[
\frac{v^2(x)}{m} = \frac{2}{m} (E - V(x))
\]  

(3.3)

where \( x \) is the displacement of the particle. \( E \) is the total energy, \( V(x) \) is the potential energy and \( m \) is the mass of the particle. It is assumed that the particle is confined to the range of position \( -L \leq x \leq L \) and the velocity at the extreme range of the oscillation is \( v(-L) = v(L) = 0 \).

In quantum physics, the particles position and velocity are based on random measurements. These measurements are described using probability densities. The probability density for the position of the particle satisfies Schrodinger Equation (3.2). Since the position is random, it is possible to record a measurement outside the range \( -L \leq x \leq L \), although the probability of recording such a measurement is very low. However because the Schrodinger equation describes probability it must be able to account for these measurements. It achieves this by allowing the potential energy to exceed the total energy, \( V(x) > E \), which occurs when \( x > L \) and \( x < -L \). This causes the wave function to change form into a rapidly decaying function outside of the range \( -L \leq x \leq L \) in agreement with the low probability of a measurement.

Due to the possibility of \( V(x) > E \), the use of a single equation to describe the
probability density is not in agreement with normal dynamics. For this reason, it is preferable to approximate the Schrödinger Equation (3.2) as two separate equations. It is also preferable to use velocity instead of energy, which is easier to visualize.

These two separate equations can be identified according to the nature of the probability density. The first equation accounts for those measurements outside the nominal range of particle oscillation where the probability density, $\psi(x)^2$, decays rapidly as $|x|$ increases:

$$\psi'(x) = \frac{I}{\eta} \psi(x) \sqrt{2L - |x|}^2 \quad \text{for } |x| > L$$

(3.4)

In the second equation the probability density, $\psi(x)^2$, is stable with respect to $|x|$:

$$\psi''(x) = -\frac{I}{\eta^2} \psi(x) \nu(x)^2 \quad \text{for } |x| \leq L$$

(3.5)

The location, $x = L$, is defined as the point where

$$\nu(x) = 0$$

(3.6)

An exact analytical solution of Schrödinger’s equation is very complicated and is available for only a few problems. However, an approximate solution, called the WKB method can be obtained easily from Equations (3.4) and (3.5) [see Park, 1964]. The WKB method is named after Wentzel, Kramers and Brillouin who introduced this approximate solution independently, and almost simultaneously, in 1926. The approximate solution for wave function in the region of $|x| \leq L$ is
\[ \psi(x) = \frac{A \cos\left(\int v(x)dx + B\right)}{\sqrt{v(x)}} \]  \hspace{1cm} (3.7)

where \( A \) and \( B \) are constants. This is a wave of spatially varying amplitude and frequency. It is this wave-like behavior, which has led to Quantum mechanics sometimes being called “Wave Mechanics”. The component of the phase of the wave, which varies with position is

\[ \theta(x) = \int v(x)dx + B \]  \hspace{1cm} (3.8)

From which the instantaneous frequency [Cohen, 1995, page 15] of the wave can be obtained as

\[ \omega(x) = \frac{d\theta(x)}{dx} = v(x) \]  \hspace{1cm} (3.9)

Note the instantaneous frequency is equal to the velocity of the particle at the location \( x \).

The amplitude of the wave is

\[ a(x) = \frac{A}{\sqrt{v(x)}} \]  \hspace{1cm} (3.10)

which is identical to the classical probability density for an oscillator with random phase [Papoulis, 1965, page 133]. The quantum probability density is

\[ \psi(x)\psi(x)^* = \frac{A^2 \cos^2\left(\int v(x)dx + B\right)}{v(x)} \]  \hspace{1cm} (3.11)
Figure 3.1 Approximate solution of the Schrödinger equation for the wave function, $\psi(x)$, of an oscillator.
In the region in which the probability density decays given by $|x| > L$, the approximate solution is

$$\psi(x) = \frac{C}{\sqrt{v(x)}} \exp\left( - \int v(x) dx \right)$$ \hspace{1cm} (3.12)

The constants $A$, $B$ and $C$ are determined at the boundary of the two regions $|x| = L$, where the solutions must match each other. Figure 3.1 shows the form of the approximate solution.

### 3.3 HERMITE FUNCTIONS

The Harmonic oscillator is one of the few dynamical systems, which has an analytical solution of the Schrodinger equation. The energy equation for the harmonic oscillator is

$$E = \frac{1}{2} m v^2 + \frac{1}{2} k x^2$$ \hspace{1cm} (3.13)

where $k$ is the spring constant and $m$ is the particle mass. The location of the point where $v(x) = 0$ is given by

$$x = L = \sqrt{2E/k}$$ \hspace{1cm} (3.14)

Making the velocity the subject of the energy equation gives

$$v = \sqrt{\frac{E - kx^2/2}{m/2}}$$ \hspace{1cm} (3.15)

By substituting this velocity into the Schrodinger equation we obtain

$$\psi''(x) + \frac{2m}{\eta^2} \left( E - \frac{1}{2} kx^2 \right) \psi(x) = 0$$ \hspace{1cm} (3.16)
Then by using the power series method of solving differential equations [Richtmyer et al, 1969, page 351], a solution can be shown to exist only for discrete values of $E$ given by

$$E = \frac{\eta}{2} \sqrt{\frac{k}{m}} (2n + 1)$$

(3.17)

where $n$ is an integer. The function, $\psi(x)$, which satisfies the Schrodinger equation for these discrete energies is

$$\psi(x) = \left( \frac{m}{kn^2} \right)^{\frac{i}{2}} h_n(t)$$

(3.18)

where $t$ is the scaled coordinate

$$t = \left( \frac{m}{kn^2} \right)^{\frac{i}{2}} x$$

(3.19)

and $h_n(t)$ is a Hermite function, which is derived from the Hermite polynomials, $H_n(t)$, by

$$h_n(t) = \frac{H_n(t)e^{-t^2/2}}{\sqrt{2^n n! \sqrt{\pi}}}$$

(3.20)

The Hermite polynomials are themselves derived from the differentiation of the Gaussian function.
**Figure 3.2(a) Hermite function of order \(\{0\}\), the Gaussian function.**

\[
h_0(t) = \frac{\exp\left(-t^2/2\right)}{\sqrt{\pi}}
\]

**Figure 3.2(b) Hermite function of order \(\{1\}\)**

\[
h_1(t) = \frac{(2t)\exp\left(-t^2/2\right)}{\sqrt{2\pi}}
\]

**Figure 3.2(c) Hermite function of order \(\{2\}\)**

\[
h_2(t) = \frac{(4t^2 - 1)\exp\left(-t^2/2\right)}{2\sqrt{2\pi}}
\]
$h_3(t) = \frac{(8t^3 - 12t) \exp(-t^2/2)}{4\sqrt{3\pi}}$

Figure 3.2(d) Hermite function of order \(3\)

$h_4(t) = \frac{(16t^4 - 48t^2 + 12) \exp(-t^2/2)}{8\sqrt{6\pi}}$

Figure 3.2(e) Hermite function of order \(4\)

$h_5(t) = \frac{(32t^5 - 160t^3 + 120t) \exp(-t^2/2)}{16\sqrt{15\pi}}$

Figure 3.2(f) Hermite function of order \(5\)
CHAPTER 3 HERMITE NEURAL NETWORK

\[ H_n(t) = (-1)^n e^{-t^2} \frac{d^n}{dt^n} e^{-t^2} \]  
(3.21)

Some of the Hermite functions are shown in Figure 3.2(a),(b),(c),(d),(e) and (f). As the order of the Hermite functions increases, they begin to display all the characteristics of the approximate solution given in Section 3.2 by Equations (3.7) and (3.12).

Figure 3.3 shows the classical probability density and quantum probability density solutions for the Harmonic oscillator of Equation (3.3). The location where the velocity is equal to zero, \( |x| = L \), given by Equation (3.14), corresponds to the \( t \) coordinate given by

\[ |t| = \sqrt{2n + 1} \]  
(3.22)

This is indicated in Figure 3.3 by the dashed line of the classical probability density at \( t = \sqrt{21} \approx 4.6 \). For \( |t| \leq \sqrt{2n + 1} \), the higher order Hermite functions resemble a cosine or sine wave (Figure 3.4) of a slowly decreasing frequency and increasing amplitude, approximately given by Equations (3.9) and Equations (3.10) respectively. For \( |t| > \sqrt{2n + 1} \), the Hermite function rapidly decays to zero.

The Hermite functions form an orthonormal set of functions in the range \( (-\infty, +\infty) \), that is

\[ \int_{-\infty}^{+\infty} h_n(t) h_m(t) dt = \begin{cases} 1 & \text{for } n = m \\ 0 & \text{for } n \neq m \end{cases} \]  
(3.23)

Therefore they may be used as an orthonormal neural network of the type described in Section 2.6.
classical: \( f(t) = \frac{1}{\pi \sqrt{(2n+1) - t^2}} = \frac{1}{\pi \sqrt{(1 - t^2)}} \)  

quantum: \( f(t) = h_{10}(t)^2 \)

Figure 3.3 Classical and quantum mechanical probability density solutions for the harmonic oscillator \((n=10)\).

Figure 3.4 Hermite function of order \(\{10\}\), corresponding to figure 3.3.
3.4 HERMITE NEURAL NETWORK

The first application of the Hermite functions to practical problems was by Klauder [1960] when he investigated their use as possible modulation functions for CHIRP radar signals [Klauder et al, 1960]. Further application to radar seems to have been limited, but recently Alsup and Whitehouse [2000] applied the functions to design Sonar waveforms.

In 1984, Nandedkar et al applied the Hermite series to the interpolation of the signals recorded from the electrical stimulation of the organs of biological systems. Since then they have become popular in biological engineering, with applications reported in 1994 by Lo Conte et al and also in 1997 by Rasiah et al. The reason for this popularity is due to the fact that the lower order terms of the Hermite series bear some resemblance to the recorded signals from electrically stimulated organs. This allows a compact series to be achieved in the interpolation of these types of signals. Recent research with Hermite functions in biological engineering has been to combine them with a self-organizing neural network [Lagerholm et al, 2000], allowing clustering of signals to be achieved.

In image and vision processing, the Hermite series together with the closely related Hermite-Rodriguez series, have also proved quite popular recently. Hermite-Rodriguez functions are Gaussian windowed Hermite functions, which are also known by the name Gaussian derivatives. Marten [1990] described these types of functions in detail where he argued the potential benefits of the Gaussian window in the filtering of images. Konstantopoulos [1990] demonstrated the deconvolution of images using the Hermite-Rodriguez functions. Konstantopoulos [1990] paper provides a useful deconvolution algorithm, which can be applied to other filter functions besides the Gaussian function. Further deconvolution algorithms have been demonstrated by Nayakkankuppam and
Venkatesh [1994], but this time using the Hermite functions without windowing.

Although the above publications have referred to the Hermite functions as simply an orthogonal series of functions, the name Hermite Neural Network (HNN) used throughout this thesis is justified because of the application of the gradient descent algorithm to train the series and also because of the recent generalization of the orthogonal series as a neural network [Ulug, 1992], [Yang and Tseng, 1996]. Gradient descent has traditionally been associated with the sigmoid neural network.

The single layer HNN approximation for a function, \( x(t) \), is

\[
x_h(t) = \sum_{n=0}^{N} a_n h_n(t)
\]  

(3.24)

where the coefficients, \( \{a_n\} \), are the weights of the network.

For practical application, the derivative of a Gaussian function used to define the Hermite functions, Equation (3.21), must be replaced with a recurrence relation, which is easier to program in computer code. The recurrence relation for the Hermite functions is [Walter, 1994, page 87]

\[
h_{k+1}(t) = t \cdot h_k(t) \sqrt{\frac{2}{k+1}} - h_{k-1}(t) \sqrt{\frac{k}{k+1}}
\]  

(3.25)

with the initial values

\[
h_0(t) = \frac{I}{\sqrt{\pi}} e^{-t^2/2}
\]  

(3.26)

and
As well as this recurrence relation, and the Gaussian derivative Equation (3.21), the following mathematical equation [Walter, 1994, page 22] can also generate the Hermite functions

\[
h_{n+1}(t) = \frac{1}{\sqrt{2n+1}} \left\{ t - \frac{d}{dt} \right\} h_n(t)
\]  

(3.28)

where the initial function to use in Equation (3.28) is the fundamental Hermite function \( h_0(t) \). The remainder Hermite functions are then determined by the successive application of the mathematical operation given in Equation (3.28).

Whilst it is not suitable for computer code, Equation (3.28) is important because it can be used to derive the Fourier transform properties of the whole series by transformation of the equation into the frequency domain [Walter, 1994, page 22]. Using the differentiation property of the Fourier transform [Champeney, 1973, page 17], Equation (3.28) transforms into the frequency domain as

\[
\Psi_{n+1}(\omega) = \frac{-j}{\sqrt{2n+1}} \left( \omega - \frac{d}{d\omega} \right) \Psi_n(\omega)
\]  

(3.29)

where \( \Psi_n(\omega) \) is the Fourier transform of \( h_n(t) \). Apart from the constant \(-j\), Equation (3.29) is identical to Equation (3.28). The appropriate initial function to use is the Fourier transform of the fundamental Hermite function, which is also a Gaussian function given by

\[
\Psi_0(\omega) = \sqrt{2\pi} h_0(\omega)
\]  

(3.30)
The remainder of the Hermite Fourier transforms can then be obtained by successive application of Equation (3.29). Since the initial functions of Equations (3.28) and (3.29) are both the same, and since both equations perform the same mathematical operation, it follows that the remainder of the functions generated by Equation (3.29) are the same as for Equation (3.28), apart from the constant factor \((- j)^n\), and therefore

\[
\Psi_n(\omega) = \sqrt{2\pi}(- j)^n h_n(\omega)
\]

Apart from the complex constant, the Hermite functions are unchanged by the Fourier transform. This important property, called isomorphic Fourier transform, will be used in Chapter 4 to prove a correlation relationship between the Hermite functions.

The isomorphic property allows a neural network interpolation of the Fourier transform of \(x(t)\) to be obtained simultaneously with the time domain neural network interpolation as

\[
X_n(\omega) = \sqrt{2\pi} \sum_{n=0}^{N} a_n (- j)^n h_n(\omega)
\]

In the literature, the isomorphic property is well known being mentioned by both Martens [1990] and Lo Conte et al [1995] and more recently by Rao and Sakar [1999] in their simultaneous time and frequency domain expansion of electromagnetic fields. In a further development of the simultaneous time and frequency domain expansions, Hu et al [2001] optimized the parameters of the HNN including the scale, origin and the order of the network.
3.5 DURATION AND BANDWIDTH OF THE HERMITE NEURAL NETWORK

In order to approximate a function, the size of the network needed must be estimated and the training function scaled into the appropriate range of the network. These factors are determined from the duration and frequency bandwidth of the network. Efficient function approximation involves matching the spatial and frequency bandwidth of the network as closely as possible to that of the function.

Due to the exponential decay of the Hermite functions, the effective range of the functions is considerably less than the infinite interval of the orthonormal condition (Equation (3.23)) would suggest. In Section 3.3 the Hermite function was shown to closely resemble a cosine signal of gradually increasing amplitude and decreasing frequency as it moves further away from the origin. Beyond a certain range, the \( \exp(-t^2/2) \) part of the Hermite function dominates and it drops rapidly to zero. For interpolation purposes, only the central portion of the Hermite function that resembles a cosine is effective. Therefore the duration, \( D \), of the network is the central region of the highest order, \( N \), Hermite function in the network up to the point where the Hermite function begins to decay. This was shown in Section 3.3 to be the location where the velocity of the Harmonic oscillator is zero, that is

\[
D = \sqrt{2N + 1}
\]  

(3.33)

where \( N \) is the order of the Hermite function.

To account for any high frequency components that a function might have, the bandwidth in the frequency domain must also be adequate.
Due to the isomorphic Fourier transform, by similar reasoning to that used to determine the duration, the frequency bandwidth, $B$, is given by

$$B = \sqrt{2N + 1}$$  \hspace{1cm} (3.34)

### 3.6 TRAINING OF THE HNN

The training of the HNN of this thesis follows the procedure discussed in Chapter 2.6 for the general application to orthonormal functions. First the weights were estimated using numerical integration followed by optimization with gradient descent.

An initial estimate of the weights of a HNN is the integration

$$a_m = \int_{-\infty}^{\infty} x(t) h_m(t) dt$$  \hspace{1cm} (3.35)

This integration may be evaluated numerically with Gauss-Jacobian integration [Rasiah et al, 1997]. The difficulty with Gauss-Jacobian numerical integration is that the zeroes of the Hermite functions are required. For an order $n$ Hermite function there are $n$ zeroes. Consequently for a neural network consisting of a total of $N$ Hermite functions, the number of zeroes that must be found will be quite large. In biomedical engineering applications, where only a few Hermite functions are required, this is acceptable but for large networks it places a heavy burden on computational speed and storage space. For this reason a simple summation of the training data, $\{x(t_i) : i = 0,1,2,\ldots,I-1\}$, similar to Euler integration was used, given by

$$a_m = \frac{T}{I} \sum_{i=0}^{I-l} y(t_i) h_m(t_i)$$  \hspace{1cm} (3.36)

where $\{-T/2 \leq t \leq T/2\}$. A feature of this type of integration is that it is also suitable
for randomly sampled data. For random data, this type of numerical integration generalizes to Monte-Carlo integration.

Gradient descent was then applied to determine the optimum weights by successive iterations of the following algorithm

$$a_m(t_{i+1}) = a_m(t_i) - \alpha \left( y(t_i) - \sum_{n=0}^{N} a_n(t_i) h_n(t_i) \right) h_m(t_i)$$  \hspace{1cm} (3.37)

where $\alpha$ is the learning rate constant chosen in the range $0.0$ to $1.0$. Some researchers have also optimized the scale and/or bias of the Hermite network with gradient descent or a similar algorithm [Kwok and Yeung, 1996], [Hu et al, 2001].

Faster training algorithms than gradient descent are available, such as the conjugate gradient algorithm [Battiti, 1992] and the Levenberg-Marquardt algorithm [Kollias and Anastassiou, 1989], which have been compared with gradient descent by Hagan and Menhaj 1994]. Unfortunately these algorithms are complex to apply. In the practical problems investigated in Chapters 5 and 7 of this thesis, the initial solution of the HNN weights by the approximate integration of equation (3.36) was sufficiently close to the optimum solution that training by gradient descent required only a few iterations of the algorithm. In this case application of the faster training algorithms was not warranted due to the fact that they would provide only a marginal improvement in speed but at a cost of increased complexity.

**3.7 SCALING THE HERMITE NEURAL NETWORK**

The Hermite neural network may be scaled into the range of the training data by introducing the scaling factor, $\lambda$. 
In this case, the scaled Hermite functions, which are called associated Hermite functions [Lo Conte et al, 1995], are given by

\[ \tilde{h}_n(t) = \frac{h_n(t/\lambda)}{\sqrt{\lambda}} \]  

(3.38)

The scale factor in the denominator ensures that the scaled Hermite functions are orthonormal. The duration and spectral bandwidths are given by, respectively,

\[ D = \frac{\sqrt{2N+I}}{\lambda} \]  

(3.39)

and

\[ B = \lambda \sqrt{2N+I} \]  

(3.40)

Scaling of the Hermite functions adds an additional variable, \( \lambda \), to the mathematical equations. To avoid using this variable, the scaled Hermite neural network will not be used in this thesis. Instead in the practical applications in this thesis the data has been scaled into the appropriate range of the Hermite neural network.

### 3.8 COMPARISON WITH THE FOURIER NEURAL NETWORK

The Fourier series is the best known of the orthogonal functions. They were also the first orthogonal series in which the gradient descent algorithm was applied to, leading to the general description of this series as a neural network [Ulug, 1992], [Zhu and Paul, 1994], [Osowski, 1992]. In view of the importance of the Fourier series in mathematics, which are well developed in theory, in this section a comparison of the HNN will be made with the Fourier neural network (FNN). This comparison will be extended in the Chapters 4, 5 and 7.
The FNN approximation of \( x(t) \) on the interval \( \{-T/2 \leq t \leq T/2\} \) consists of alternating cosine and sine functions

\[
x(t) \equiv x_d(t) = \sum_{n=0}^{N} b_n \theta_n(t) \tag{3.41}
\]

where \( b_n \) are the weights and \( \theta_n(t) \) is a cosine or sine function depending on \( n \) according to

\[
\theta_n(t) = \begin{cases} 
\frac{1}{\sqrt{T}} & \text{for } n = 0 \\
\frac{1}{\sqrt{2T}} \cos(2\pi n t/T) & \text{for even } n \\
\frac{1}{\sqrt{2T}} \sin(2\pi n t/T) & \text{for odd } n
\end{cases}
\tag{3.42}
\]

This can be conveniently rewritten as an equivalent complex Fourier neural network by replacing \( \theta_n(t) \) by a complex activation function, \( \Theta_n(t) \), given as

\[
\Theta_n(t) = \begin{cases} 
\frac{1}{\sqrt{T}} & \text{for } n = 0 \\
\frac{1}{\sqrt{2T}} e^{j2\pi n t/T} & \text{for } n \neq 0
\end{cases}
\tag{3.43}
\]

so that

\[
x_{\Theta}(t) = \sum_{n=0}^{N} c_n \Theta_n(t) \tag{3.44}
\]

The weights, \( c_n \), are now complex and are related to the weights \( b_n \) by

\[
c_n = \begin{cases} 
b_n & \text{for even } n \\
jb_n & \text{for odd } n
\end{cases}
\tag{3.45}
\]
\( c_n \) can be obtained by the same procedure discussed in Section 3.6 for the HNN and given in the general form shown in Chapter 2.6. In addition, Osowski [1992] has shown how to optimize the frequency, \( f = \frac{2\pi}{T} \), of the network by using gradient descent, which may be necessary for some practical problems.

Note that \( \Theta_n(t) \) form an orthonormal basis on the interval \( \{-T/2 \leq t \leq T/2\} \) and

\[
\int_{-T/2}^{T/2} \Theta_n(t)\Theta_m(t)dt = \frac{1}{2T} \int_{-T/2}^{T/2} e^{i2\pi(n-m)t/T} dt = \begin{cases} 1 & n = m \\ 0 & n \neq m \end{cases} \quad (3.46)
\]

The finite interval of application as opposed to the infinite interval of the Hermite series is one of the fundamental differences between the Fourier and Hermite series. An equivalent duration of the Fourier series is

\[ D = \frac{T}{2} \quad (3.47) \]

The Fourier transform, \( X_\omega(\omega) \), of the FNN, \( x_\omega(t) \), is discontinuous, consisting of a discrete series of equally spaced Dirac impulse functions given by

\[
X_\omega(\omega) = \sqrt{2T} \sum_{n=1}^{N/2} c_n \left( \delta(\omega - 2n\pi/T) + \delta(\omega + 2n\pi/T) \right) + c_0 \sqrt{T} \delta(\omega) \quad (3.48)
\]

The location of the furthest impulse from the origin defines the bandwidth, which is

\[ B = \pi N / T \quad (3.49) \]

where \( N \) is the total number of sine and cosine functions in the network. A continuous interpolation of the Fourier transform for the FNN is given by [Papoulis, 1962, page 52]

\[
X_\omega(\omega) = \sqrt{2T} \sum_{n=1}^{N/2} c_n \left( \frac{\sin(\omega T/2 - n\pi)}{\omega T/2 - n\pi} + \frac{\sin(\omega T/2 + n\pi)}{\omega T/2 + n\pi} \right) + c_0 \sqrt{T} \quad (3.50)
\]
Figure 3.5 Hermite neural network interpolation of rectangular function.

Figure 3.6 Error of Hermite neural network interpolation of rectangular function.
Figure 3.7 Fourier neural network interpolation of rectangular function.

Figure 3.8 Error of Fourier neural network interpolation of rectangular function.
Figure 3.9 Hermite neural network Fourier transform of rectangular function.

Figure 3.10 Error of Hermite neural network Fourier transform of rectangular function.
Figure 3.11 Fourier neural network Fourier transform of rectangular function.

Figure 3.12 Error of Fourier neural network Fourier transform of rectangular function.
For a finite duration signal matched to the period of the Fourier functions, this interpolation uniquely determines the Fourier transform of the signal [Papoulis, 1962, page 52].

It is interesting to compare the spectral bandwidth of the Fourier and Hermite neural networks of order $N$ on a signal of the same duration, $D = \sqrt{2N+1}$. It follows from Equation (3.34) that the Hermite neural network has a bandwidth $B = \sqrt{2N+1} = 1.41\sqrt{N}$. The Fourier neural network has a bandwidth of $B = \pi N/(2\sqrt{2N+1}) = 1.11\sqrt{N}$, which is 20% lower than the Hermite network. If the complexity of a signal is measured by the highest frequency component occurring in the signal, then it can be expected that the Hermite neural network can approximate more complex signals with fewer activation functions.

A rectangular pulse is a simple signal with which to compare the two neural networks. This signal is

$$x(t) = \begin{cases} 
1 & \text{for } -5 \leq t \leq 5 \\
0 & \text{for } t > 5 \\
0 & \text{for } t < -5
\end{cases} \quad (3.51)$$

The theoretical Fourier transform of the rectangular pulse is [Champeney, 1973, page 25]

$$X_\omega(\omega) = \frac{2\sin(5\omega)}{\omega} \quad (3.52)$$

The rectangular pulse was sampled in steps of 0.1 units across an interval $\{-10 \leq t \leq 10\}$ giving a training data set of 200 samples.
Figure 3.13 Rmse versus the number of activation functions for the rectangular function.

Figure 3.14 Rmse versus the sampling rate (number of samples per unit) for the rectangular function.
Figure 3.5 shows the HNN approximation of this pulse, where the HNN consists of fifty Hermite functions trained with twenty cycles of the gradient descent algorithm on these 200 samples. “Ringing” occurs at the discontinuity of the pulse in a similar way to that occurring with a Fourier approximation.

Figure 3.6 shows the error, \( x_h(t) - x(t) \), of the HNN across the range of the data. Figure 3.7 and 3.8 shows the FNN interpolation and the FNN error, \( x_\phi(t) - x(t) \), respectively of the same rectangular function. The FNN consisted of twenty five cosine and twenty five sine functions giving a total of fifty activation functions in the network. The error of each of the interpolations is almost identical.

Figure 3.9 shows the HNN approximation of the Fourier transform of the rectangular function and the theoretical transform given by equation (3.52). Figure 3.10 shows the error, \( X_h(\omega) - X(\omega) \), of the HNN Fourier transform across the range of the data. Figure 3.11 and Figure 3.12 shows the FNN approximation (Equation (3.50)) and FNN error, \( X_\phi(\omega) - X(\omega) \), of the Fourier transform of the rectangular function. The FNN Fourier transform has a much lower error than the HNN Fourier transform across the central band of low to mid frequencies but a greater error at the higher frequencies.

Other features of the networks are described in Figures 3.13 and Figure 3.14. Figure 3.13 indicates that both series require about the same number of activation functions, approximately fifteen, for satisfactory interpolation. Figure 3.14 shows the sampling rate. The rmse of the FNN drops to a satisfactory level slightly before the HNN at a sampling rate of 1.4 compared to 1.6 for the HNN.

Osowski [1992] investigated the FNN with the aim of producing a simulation of an analogue neural network consisting of transistors and other electronic components.
Figure 3.15 Training speed using the gradient descent algorithm (random initial weights) for the rectangular function. Number of cycles is the number of repetitions of the training data set.
In such a network the number of cycles of the gradient descent algorithm provides a measure of the training speed. Given the recent electronic synthesis of Hermite functions by Chesnokov [1999] it is of interest to compare the training speeds of both networks. Figure 3.15 shows the training speed of the HNN and the FNN. Both show nearly identical performance on the interpolation of the rectangular pulse.

3.9 CONCLUSION

This chapter is essentially a review of the Hermite series of functions. The properties of the Hermite functions have been explained from their application in quantum physics where they are the solution of the harmonic oscillator. The chief characteristics are:

- Hermite functions resemble a cosine or sine wave of finite duration.
- The duration of the Hermite function is equal to \( \sqrt{2N+1} \).
- The instantaneous frequency of the Hermite functions decreases and the amplitude increases as the distance from the origin increases.
- The Fourier transform of a Hermite function is also a Hermite function. This allows simultaneous interpolation in the time and frequency domains.

In addition to the review, a method for training the Hermite series to interpolate an arbitrary function has also been given. This method is based on numerical integration followed by optimization with the gradient descent algorithm. The application of the gradient descent algorithm justifies the classification of the Hermite series as a Hermite Neural Network (HNN). A brief comparison has been made with a Fourier Neural Network (FNN). In tests on a rectangular pulse both networks achieved the same level of accuracy and the same training speed using the gradient descent algorithm.
In the next chapter an algorithm for performing the correlation of two arbitrary functions using the HNN will be presented.
CHAPTER 4

HERMITE NEURAL NETWORK

CORRELATION

4.1 INTRODUCTION

Correlation algorithms using Hermite series has been successfully applied by Nayakkankuppam and Venkatesh [1995] and Konstantopoulos et al [1990] to the deconvolution of blurred Gaussian images. Nayakkankuppam and Venkatesh [1995] matched the scale of associated Hermite series in the frequency domain to achieve a linear set of equations in the form of a matrix equation; the limitation is that it requires the solution of a matrix equation. Konstantopoulos et al’s [1990] algorithms were based on the convolution properties of the Hermite-Rodriguez functions [Lo Conte et al, 1995]. These functions involve the product of a Gaussian window with the Hermite functions, complicating the estimation of the scaling constant. In this Chapter we develop a new correlation algorithm for the Hermite functions.

An important property of a Hermite function is that its Fourier transform is also a Hermite function apart from a multiplication factor that is dependent on the order of the Hermite function. Therefore the Fourier transform of a function, approximated by a Hermite series neural network, can be obtained readily. This property is exploited to
develop the algorithm for calculating the correlation using HNNs. This algorithm bypasses the usual Fourier transform approach to calculate the correlation. Instead the correlation is calculated directly from the Hermite series coefficients. The advantage is that for some applications (which are defined in Section 4.2), the direct approach is computationally more efficient.

4.2 CORRELATION USING HERMITE NEURAL NETWORKS

The correlation of two functions $x(t)$ and $y(t)$ is defined as

$$r(t) = x(t) \ast y(t) = \int_{-\infty}^{\infty} x(\tau)y(\tau + t) d\tau$$  \hspace{1cm} (4.1)

Expanding $x(t)$ and $y(t)$ as HNNs

$$x_h(t) = \sum_{n=0}^{N} a_n h_n(t)$$  \hspace{1cm} (4.2)

and

$$y_h(t) = \sum_{n=0}^{N} b_n h_n(t)$$  \hspace{1cm} (4.3)

where $a_n$ and $b_n$ are the HNN weights. The correlation of these two HNNs, which approximates Equation (4.1), is given by the integration

$$r_h(t) = \int_{-\infty}^{\infty} x_h(\tau)y_h(\tau + t) d\tau = \sum_{n,m} a_n b_m \{ h_n(t) \ast h_m(t) \}$$  \hspace{1cm} (4.4)

where $h_n(t) \ast h_m(t)$ is the correlation of two Hermite functions of different order, which is given by
Therefore the correlation of two HNNs reduces to that of a summation of correlated Hermite functions.

Although it is possible to evaluate the correlation of the Hermite functions, $h_n(t) \ast h_m(t)$, in the time domain, a mathematically simpler method is to work in the frequency domain and then take the inverse Fourier transform to obtain the correlation function. In the frequency domain the integration associated with the correlation, becomes a complex conjugate multiplication, which is easier to manipulate. Using this method, the correlation of the Hermite functions from a frequency viewpoint is

$$h_n(t) \ast h_m(t) = (-1)^m \int_{-\infty}^{\infty} h_n(\omega)h_m(\omega)e^{i\omega} d\omega$$

(4.6)

where we have used the Fourier transform, $F\{h_n(t)\}$, of the Hermite function given by

$$F\{h_n(t)\} = (-j)^n \sqrt{2\pi} \cdot h_n(\omega)$$

(4.7)

For the special case of $n = m$, the correlation is a Laguerre function [Klauder, 1960]. The general case of $n \neq m$ is not usually given in the general mathematical literature in the form of Equation (4.5) although the integral in Equation (4.6) can be found listed in integration tables [Prudnikov et al, 1986, page 505, Equation 2.20.15.22] as a cosine and sine integration. It is beneficial to show how it can be evaluated directly because a similar procedure may possibly be applied to other types of orthonormal functions, which are not listed in tables. It can be evaluated directly as follows. The product formula [Prudnikov et al, 1986] for Hermite polynomials is
CHAPTER 4 HERMITE NEURAL NETWORK CORRELATION

\[ H_n(\omega)H_m(\omega) = \sum_{k=0}^{m} \frac{k!}{2^k} \binom{m}{k} \binom{n}{k} H_{m+n-2k}(\omega) \quad (m \leq n) \]  
(4.8)

Substituting Equation (4.8) and (3.20) into Equation (4.6) gives the integration equation

\[ h_n(t) * h_m(t) = \left( \frac{-1}{\sqrt{2^{n+m} \pi n! m!}} \right) \sum_{k=0}^{m} \frac{k!}{2^k} \binom{m}{k} \binom{n}{k} \int_{-\infty}^{\infty} e^{-\omega^2} \left( \omega e^{jt} \right)^{m+n-2k} d\omega \quad (m \leq n) \]  
(4.9)

Equation (4.9) can be rewritten in a more suitable form for integration with the Gaussian derivative definition of the Hermite polynomials (Equation (3.21)). Substituting Equation (3.21) into (4.9) gives

\[ h_n(t) * h_m(t) = \left( \frac{-1}{\sqrt{2^{n+m} \pi n! m!}} \right) \sum_{k=0}^{m} \frac{k!}{2^k} \binom{m}{k} \binom{n}{k} \int_{-\infty}^{\infty} \frac{d^{n+m-2k}}{d\omega^{n+m-2k}} (e^{-\omega^2}) e^{jt} d\omega \quad (m \leq n) \]  
(4.10)

The integration in Equation (4.10) is the inverse Fourier transform (IFT) of the derivative of a Gaussian function. A solution for this integration can be obtained by using the differentiation relation for Fourier transforms [Champeney, 1973, page 17] on the Fourier transform of the Gaussian function [Papoulis, 1962, page 24-25] to give

\[ \int_{-\infty}^{\infty} \frac{d^n}{d\omega^n} (e^{-\omega^2}) e^{jt} d\omega = (-j)^n t^n e^{-t^2/4} \sqrt{\pi} \]  
(4.11)

Using this result we have

\[ h_n(t) * h_m(t) = \left( \frac{-1}{\sqrt{2^{n+m} \pi n! m!}} \right) \sum_{k=0}^{m} \frac{k!}{2^k} \binom{m}{k} \binom{n}{k} (-j)^k t^{n+m-2k} e^{-t^2/4} \quad (m \leq n) \]  
(4.12)

By changing the order of the summation this may be written as an orthonormal associated Laguerre function
where \( I^m_n(t) \) is the orthonormal associated Laguerre function, related to the associated Laguerre polynomials, \( L^m_n(t) \), by

\[
I^m_n(t) = \frac{t^{n/2}e^{-t/2}L^m_n(t)}{\sqrt{(n+m)!/m!}}
\]  

With this result we can now evaluate the correlation of the two HNNs. Substituting Equation (4.13) into the network correlation Equation (4.4) we obtain the final result

\[
r_n(t) = \sum_{m=0}^{N} \sum_{n=m}^{N} (a^*_m b^n_m (-1)^{n+m} \delta_{n,m} + a^n_m b^*_m) I^m_n(t) \frac{t^2}{2} \quad \text{for } t \geq 0
\]  

where

\[
\delta_{n,m} = \begin{cases} 
1 & \text{for } n \neq m \\
0 & \text{for } n = m
\end{cases}
\]  

with a similar summation for \( t < 0 \).

This equation is a summation of \( N^2 \) terms. Stone and Williams [1995] have shown that only the Fourier family of linear transforms can replace the \( N^2 \) summation terms of a convolution integral in the time domain by \( N \) summations in the transform domain. Generally the computation of the correlation will be faster with a Fourier transform network except for the special case that occurs when one of the functions is a Hermite function, \( x(t) = h_k(t) \). In this case the correlation reduces to a summation of \( N \) terms given by
with a similar summation for \( t < 0 \). Functions that can be represented by a single Hermite function are the Gaussian function, \( h_0(t) \), and the rotationally invariant CHIRP radar signal [Klauder, 1960]. The Gaussian function correlation is important in the filtering of noisy signals [Blinchikoff and Zverev, 1976] and also in the previously mentioned image and signal deconvolution [Nayakkankuppam and Venkatesh, 1995], [Konstantopoulos et al 1990].

4.3 CONVOLUTION OF TWO HERMITE NEURAL NETWORKS

The convolution integral is

\[
\int_{-\infty}^{\infty} x(\tau) y(t-\tau) d\tau
\]

which is closely related to the correlation integral. The results presented in this chapter can be applied to the convolution integral by direct multiplication in the frequency domain rather than using the complex conjugate.

4.4 RECURRENCE RELATIONS FOR THE LAGUERRE FUNCTIONS

Application of the correlation algorithm requires the associated Laguerre functions. Associated Laguerre functions are computed numerically by the following recurrence relation [obtained by normalization of Equation (6.15), page 86, from Walter, 1994]

\[
l_{k+1}^n(t) \sqrt{\frac{k + l}{n + k + l}} = \frac{2k + n + 1 - t}{k + 1} \cdot l_k^n(t) - \frac{k + n}{k + 1} \cdot l_{k-1}^n(t)
\]

with the initial values
\[
\begin{align*}
I_0^n(t) &= \frac{t^{n\frac{1}{2}}e^{-\frac{t}{2}}}{\sqrt{n!}} \\
I_1^n(t) &= \frac{(n+1-t)\cdot t^{n\frac{1}{2}}e^{-\frac{t}{2}}}{\sqrt{(n+1)!}}
\end{align*}
\]

(4.20)

Due to the double indices of the associated Laguerre function more than one recurrence relation exists.

Equation (4.19) computes the associated Laguerre function from a recurrence relation using the subscript. Unfortunately for large \( n \) it becomes difficult to apply because of the power term in the initial conditions of Equation (4.20). In this case the following recurrence equation, which sidesteps the power term by using the superscript, was employed (obtained by the summation and then normalization of the second last and third last recurrence equations on page 241 of Magnus et al [1966]):

\[
I_{m}^{k+1}(t) = \frac{k + t}{\sqrt{t} \cdot (m + k + 1)} I_{m}^{k}(t) - I_{m}^{k-1}(t) \sqrt{\frac{m + k}{m + k + 1}}
\]

(4.21)

where the initial values for \( k = 0 \) and \( k = 1 \) were computed from Equation (4.19).

For small \( t \), this recurrence equation is unstable. The instability occurs as a narrow band of oscillations as shown in Figure 4.1(a). As \( t \) increases beyond the instability the recurrence equation becomes stable. The location at which the instability occurs is a function of the order \( (n,m) \) of the associated Laguerre function. Figures 4.1(a) to (c) and Figure 4.2 (a) to (c) shows the location of the instability for increasing \( m \) and \( n \) respectively. In spite of the instability, the recurrence equation can still be applied, but only in the region in which it is stable. Fortunately in the unstable region the magnitude of the Laguerre function is of negligible size and can be ignored.
Figure 4.1 (a) Associated Laguerre function $l_m^n(t)$, where $(m=0, n=100)$.

Figure 4.1 (b) Associated Laguerre function $l_m^n(t)$, where $(m=0, n=200)$.

Figure 4.1 (c) Associated Laguerre function $l_m^n(t)$, where $(m=0, n=300)$.
Figure 4.2 (a) Associated Laguerre function $L_m^n(t)$, where $(m=100, n=100)$.

Figure 4.2 (b) Associated Laguerre function $L_m^n(t)$, where $(m=200, n=100)$.

Figure 4.2 (c) Associated Laguerre function $L_m^n(t)$, where $(m=300, n=100)$. 
Table 4.1 Location of the instability, $t_{ns}$, in the associated Laguerre function generated by the recurrence relation of Equation (4.22) as a function of the order $(m,n)$ of the associated Laguerre function. Note that $t_{ns}$ is calculated from the table value, $d$, by the equation $t_{ns} = 0.5 \times d \times d$. Furthermore a factor of safety has been included to make sure the recurrence equation is only applied well clear of the instability.
In order to apply the recurrence equation, a cut-off point was defined as the location of the transition from the stable to the unstable operation. The recurrence equation was then only applied in the stable region. In the unstable region the magnitude of the associated Laguerre function was set equal to zero. Although the location of the cut-off point can be estimated by analytical methods [Press et al, 1992, page 178-183], this approach was found to be overly cautious. Instead a two dimensional lookup table, which maps the cut off point as a function of \((n,m)\), was found to be more suitable (see Table 4.1). To avoid excessive storage not all \((n,m)\) were tabulated; linear interpolation was used to locate the cut off point between values of \((n,m)\) in table 4.1. Using the table associated Laguerre functions up to a maximum order of (1000,1000).

4.5 COMPARISON WITH THE FOURIER CORRELATOR

It has been stated in the introduction that the standard numerical technique for computing the correlation of two signals is with the Fourier series. Therefore the Fourier correlation, in the form of a FNN, will be used to judge the performance of the HNN correlation. In this section the Fourier correlation will be derived.

The Fourier neural network approximation of the signals \(x(t)\) and \(y(t)\) on the interval \([-T/2 \leq t \leq T/2]\) is from Section 3.8

\[
x_{\Theta}(t) = \sum_{n=0}^{N} c_n \Theta_n(t) \tag{4.22}
\]

and

\[
y_{\Theta}(t) = \sum_{n=0}^{N} d_n \Theta_n(t) \tag{4.23}
\]
where the activation functions are the complex exponential series

\[
\Theta_n(t) = \begin{cases} 
\frac{1}{\sqrt{T}} & \text{for } n = 0 \\
\frac{1}{\sqrt{2T}} e^{j2\pi n t/T} & \text{for } n \neq 0 
\end{cases}
\] (4.24)

\(\Theta_n(t)\) are orthonormal on the interval \([-T/2 \leq t \leq T/2]\). That is they satisfy the integral

\[
\frac{1}{T/2} \int_{-T/2}^{T/2} \Theta_n(t) \Theta_m(t) dt = \frac{1}{2T} \int_{-T/2}^{T/2} e^{j2\pi(n-m)t/T} dt = \begin{cases} 
1 & n = m \\
0 & n \neq m 
\end{cases}
\] (4.25)

The correlation of the Fourier neural networks on the interval \([-T/2 \leq t \leq T/2]\) is

\[
r_\phi(t) = x_\phi(t) \ast y_\phi(t) = \int_{-T/2}^{T/2} \sum_{n=0}^{N} c_n \Theta_n(t) \left(\sum_{m=0}^{N} d_m \Theta_m(t + \tau)\right)^\ast d\tau
\] (4.26)

From their exponential form, it can be seen that \(\Theta_n(t)\), satisfy the condition

\[
\Theta_n(t + \tau) = \Theta_n(t) \Theta_n(\tau)
\] (4.27)

Then using this result with the orthonormal condition of Equation (4.30), the correlation simplifies to

\[
r_\phi(t) = \sum_{n=0}^{N} c_n d_n \Theta_n(t)
\] (4.28)

which is an order (\(N\)) summation of the product of the Fourier weights.

In Section 4.7 and Section 4.8, the performance of the HNN against the FNN will be tested. The main aim of these initial tests is to determine whether the HNN can calculate the correlation with sufficient accuracy and to check the mathematical derivations and algorithms derived in Sections 4.1 to 4.4.
4.6 ERROR OF THE HERMITE AND FOURIER CORRELATORS

Errors occur in both the HNN and FNN correlators due to the fact that the correlation integral is defined across an infinite range of $t$ whereas both the HNN and FNN only operate across a finite range. For convenience, the correlation integral (Equation (4.1)), is repeated again

$$r(t) = x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau) y(\tau + t) d\tau$$  \hspace{1cm} (4.29)

The error of the FNN correlation, called circular correlation [Newland, 1984, page 122], is a direct consequence of the periodic nature of the activation function. The FNN interpolation of $y(t)$ is periodic of period $T$, that is $y_o(t + T) = y_o(t)$. This means that the FNN correlation will not integrate the correct value of $y(t + \tau)$ in Equation (4.29) when $t + \tau > T/2$. Instead it integrates a repetition of $y(t)$.

The error of the HNN correlation is completely different to that of the FNN. The HNN approximates a rectangular window such that $x_h(t) \equiv x(t)u(t)$ and $y_h(t) \equiv y(t)u(t)$, where $u(t)$ is the rectangular window given by

$$u(t) = \begin{cases} 
1 & \text{for } -D \leq t \leq D \\
0 & \text{for } t < -D \\
0 & \text{for } t > D 
\end{cases}$$  \hspace{1cm} (4.30)

The width, $D$, of the rectangular window is given by the duration of the largest Hermite function in the network

$$D = \sqrt{2N + 1}$$  \hspace{1cm} (4.31)

where $N$ is the order of the largest Hermite function in the network. The rectangular
window means that the HNN will interpolate \( y(t + \tau) = 0 \) when \( t + \tau > T/2 \) in the correlation Equation (4.29), producing an error.

### 4.7 AUTO-CORRELATION OF A FINITE DURATION SINE WAVE

In this section we compare the performance of the HNN with a FNN on the auto-correlation of a finite duration sine wave.

The finite duration sine wave is described mathematically as

\[
x(t) = \begin{cases} 
\sin(2\pi f t) & \text{for } -L \leq t \leq L \\
0 & \text{for } t > L \\
0 & \text{for } t < -L 
\end{cases}
\]  

(4.32)

where \( t \) is the time variable measured in seconds, \( L \) is the duration of the sine wave and \( f \) is its frequency. The analytical solution for the auto-correlation of the sine wave can be shown to be a triangular modulated cosine wave

\[
r(t) = x(t) \ast x(t) = \begin{cases} 
\cos(2\pi f t) \left( 1 - \frac{t}{2L} \right) & \text{for } -2L \leq t \leq 2L \\
0 & \text{for } t < -2L \\
0 & \text{for } t > 2L 
\end{cases}
\]  

(4.33)

The particular numerical values chosen for \( f \) and \( L \) were 0.5 Hertz and 6 seconds respectively.

The HNN consisted of 50 Hermite activation functions. With this number of Hermite activation functions the duration of the HNN is \( \pm 10 \) seconds, which is more than adequate for the sine wave. The FNN consisted of 50 cosine and sine functions of period equal to 20 second. This was considered to be the closest match in size and duration to the HNN.
Figure 4.3 Hermite interpolation of $x(t)$.

Figure 4.4 Error of Hermite interpolation of $x(t)$. 
Figure 4.5 Hermite correlation $r(t) = x(t) \ast x(t)$.

Figure 4.6 Error of Hermite correlation $r(t) = x(t) \ast x(t)$. 

The training data was obtained from 200 discrete samples of the sine wave, sampled at a rate of 10 samples per second. Figure 4.3 and Figure 4.4 show the HNN interpolation of the sine wave and the error. The disturbance occurring in the error curve at ±6 seconds is due to the discontinuity in the sine wave at these points caused by the amplitude rapidly dropping to zero. The same effect occurred in the square pulse tested in Section 3.8. The accuracy obtained with the HNN interpolation was nearly identical to the FNN shown in Figure 4.7 and Figure 4.8.

Figure 4.5 shows the HNN autocorrelation of the sine wave. Due to the symmetry of the auto-correlation, only positive \( t \) is shown. Figure 4.6 shows the error with respect to the analytical correlation of Equation (4.33). Given the excellent fit to the analytical solution shown in Figure 4.5, the error is surprisingly large. The reason for the larger than expected error is that it is due to a phase shift of the sine wave, which is not noticeable in Figure 4.5.

Figure 4.9 and Figure 4.10 show the auto-correlation results for the FNN. For small \( t \), the error of the FNN correlation is the same as that of the HNN. However at large \( t \), the FNN suffers from a noticeably greater error. This is due to the circular correlation effect described in Section 4.6. The HNN correlation does not suffer from this error.

4.8 CORRELATION OF RECTANGULAR PULSES

In this section the correlation of two rectangular pulses will be investigated, where the rectangular pulse is defined as

\[
x(t) = \begin{cases} 
1 & \text{if } -1.0 \leq t \leq 1.0 \\ 
0 & \text{if } t > 1.0 \\ 
0 & \text{if } t < -1.0 
\end{cases}
\]  

(4.34)
Figure 4.7 Fourier interpolation of $x(t)$.

Figure 4.8 Error of Fourier interpolation of $x(t)$. 
Figure 4.9 Fourier correlation $r(t) = x(t) * x(t)$.

Figure 4.10 Error of Fourier correlation $r(t) = x(t) * x(t)$.
\( t \) is the time variable, which is assumed to be measured in seconds. The same rectangular pulse, but with its origin shifted by \( \tau \) seconds in the positive direction, is

\[
y(t) = x(t - \tau)
\]

(4.35)

The analytical solution for the correlation of \( x(t) \) with \( y(t) \) is the triangular function

\[
r(t) = \begin{cases} 
2 - t + \tau & \text{for } 0 \leq t - \tau \leq 2.0 \\
t - \tau + 2 & \text{for } -2.0 \leq t - \tau < 0 \\
0 & \text{for } t - \tau > 2.0 \\
0 & \text{for } t - \tau < -2.0 
\end{cases}
\]

(4.36)

Figures 4.11(a), 4.11(b) and 4.11(c) show the HNN approximation of \( x(t) \), \( y(t) \) and \( r(t) \) respectively. Figure 4.12 is an enlargement of \( r(t) \). The ringing, that occurs in the HNN approximation of the pulse \( x(t) \) does not occur in the correlation which tends to smooth the response. Figure 4.13 is the error with respect to the analytical solution for the HNN correlation. Figures 4.14 and 4.15 show the FNN approximation of \( r(t) \) and the error respectively. These figures show that the HNN achieves the same accuracy as the FNN.

Next the root mean square error of the HNN and FNN correlation with variation of the displacement \( \tau \) of \( y(t) \) was investigated (Figure 4.16). For low displacements both networks are comparable in rmse, but at large displacements the rmse of the FNN exceeds that of the HNN. The higher rmse of the FNN is due to the circular correlation mentioned in Section 4.6. This occurs as a disturbance in the left hand side of the FNN correlation at large \( \tau \) as shown in Figure 4.18 (a) to Figure 4.18 (c). The HNN correlation does not have this disturbance (Figure 4.17 (a) to Figure 4.18 (c)).
Figure 4.11 (a) Hermite interpolation of $x(t)$.

Figure 4.11 (b) Hermite interpolation of $x(t - 2.5)$.

Figure 4.11 (c) Hermite correlation $r(t) = x(t) * x(t - 2.5)$.
Figure 4.12 Enlargement of figure 4.11 (c), Hermite correlation \( r(t) = x(t) * x(t - 2.5) \).

Figure 4.13 Error of Hermite correlation \( r(t) = x(t) * x(t - 2.5) \).
Figure 4.14 Fourier correlation $r(t) = x(t) \ast x(t - 2.5)$.

Figure 4.15 Error of Fourier correlation $r(t) = x(t) \ast x(t - 2.5)$. 
4.9 CONCLUSION

The correlation of Hermite neural networks has been investigated. To the best knowledge of the author, with the available literature and reference sources quoted in this chapter and the preceding Chapter 3, this is the first time the correlation has been attempted using the Hermite series of functions. The key results are:

* A computer algorithm for the correlation of two Hermite neural networks has been developed for Hermite functions of order $N$ up to 1000.

* In tests with a FNN of comparable size and order, the HNN achieved the same level of accuracy. However the advantage of the HNN correlation is that it does not suffer from the circular correlation that is inherent in a FNN.

* The FNN correlation is a summation of $N$ terms. The correlation of two HNNs is a summation of $N^2$ terms. A correlation summation of $N$ terms is obtained with a HNN only if one of the signals in the correlation is a Gaussian function or Hermite function. Therefore except for these types of function, or simple combinations of these functions, the FNN will be faster numerically.

Due to the importance of the Gaussian function in signal processing, it is expected that this will be the main area of application of the HNN correlation. In the next chapter the correlation and inverse correlation of the HNN with a Gaussian function will be dealt with in detail.
Figure 4.16 Root mean square error (rmse) versus displacement, \( \tau \).
Figure 4.17 (a) Hermite correlation $\tau = 8.5$.

Figure 4.17 (b) Hermite correlation $\tau = 9.0$.

Figure 4.17 (c) Hermite correlation $\tau = 9.5$. 
Figure 4.18 (a) Fourier correlation $\tau = 8.5$.

Figure 4.18 (b) Fourier correlation $\tau = 9.0$.

Figure 4.18 (c) Fourier correlation $\tau = 9.5$. 
CHAPTER 5

GAUSSIAN FILTERS, INVERSE FILTERS

AND

FILTER SYNTHESIS

5.1 INTRODUCTION

For application in scientific instrumentation and industrial processing noise is often embedded in the signal and this must be removed by filtering. Whilst the sigmoid neural network can be trained to interpolate noisy signals, the filtering properties are still under development [Geman et al, 1992], [Lawrence and Giles, 2000]. This Chapter applies the Hermite correlation algorithm developed in Chapter 4 to the development of a new type of neural network filter with clearly defined parameters. This is achieved by implementing for the first time a Gaussian filter using the Hermite neural network. In contrast to the sigmoid neural network, the properties of the Gaussian filter are well developed and are given in Blinchikoff and Zverev [1976]. The procedure developed for implementing the Gaussian filter is further developed to synthesize other types of moving average filters based on the Hermite functions.

Sometimes filtering occurs as an unwanted distortion or blur of the signal. The process of recovering a signal from the distorted signal is called de-correlation or inverse
filtering. This is a much more difficult task than filtering because it is an ill-conditioned operation that is sensitive to small disturbances in the system. Equally important to the Gaussian filter, is the inverse operation whereby this Gaussian distortion is removed. In addition to the Gaussian filter and filter synthesis, this chapter also develops a new Gaussian inverse filter using the Hermite neural network. This new type of inverse filter is compared with the Weiner inverse filter on the restoration of distorted laser Doppler measurements.

5.2 GAUSSIAN FILTER

Figure 5.1 explains the operation of a Gaussian moving average filter. The filter integrates only the noisy function, \( x(\tau) \), enclosed within the Gaussian window function, \( k(t - \tau) \), located at \( t \). An average value, \( y(t) \), of the noisy function at \( t \) is obtained.

This average value is the correlation of the noisy function with the window function. This is

\[
y(t) = \int_{-\infty}^{\infty} x(\tau) k(t - \tau) d\tau = \frac{1}{\sigma \sqrt{2\pi}} \int_{-\infty}^{\infty} x(\tau) \exp \left( -\frac{(t - \tau)^2}{2\sigma^2} \right) d\tau
\]

(5.1)

where

\[
k(t) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{t^2}{2\sigma^2} \right)
\]

(5.2)

is the Gaussian window function. By using Hermite activation functions, a simple expression can be obtained for the correlation with the Gaussian window function.

Consider the approximation of a function by a Hermite network. The network approximates the function by a finite expansion of Hermite functions on an interval
Figure 5.1 Gaussian window, moving average filter.

Figure 5.2 Associated Laguerre functions, \( l_n^0(t^2/2) \), of increasing order.
\[ x(t) \equiv x_n(t) = \sum_{n=0}^{N} a_n h_n(t) \quad \{ -\infty \leq t \leq \infty \} \quad (5.3) \]

where \( \{ a_n \} \) is a set of suitably chosen weights and \( h_n(t) \) are the Hermite activation functions on the interval \( \{ -\infty, +\infty \} \). Since the fundamental Hermite activation function, \( h_0(t) \), is a Gaussian function with \( \sigma = 1 \), it may be used as the filter function and

\[ k(t) = \frac{h_0(t)}{\sqrt{2\pi}} \quad (5.4) \]

The correlation integration of Equation (5.1) is then approximately equal to the correlation of the Hermite network with the fundamental Hermite function.

\[ y(t) \equiv y_n(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \left( \sum_{n=0}^{N} a_n h_n(\tau) \right) h_0(t - \tau) d\tau \quad (5.5) \]

This expression can be evaluated from the relationship for the correlation between the Hermite functions of different order, Equation (4.13) of Chapter 4, giving

\[ \int_{-\infty}^{+\infty} h_n(\tau) h_0(t - \tau) d\tau = \begin{cases} \frac{I_0^0(t^2/2)}{1} & \text{for } t \geq 0 \\ (-1)^n I_0^n(t^2/2) & \text{for } t < 0 \end{cases} \quad (5.6) \]

where \( I_0^n(t) \) is an orthonormal associated Laguerre function (Equation (4.14)).

Substituting Equation (5.6) into Equation (5.5) we obtain

\[ y_n(t) = \begin{cases} \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N} a_n I_0^0(t^2/2) & \text{for } t \geq 0 \\ \frac{1}{\sqrt{2\pi}} \sum_{n=0}^{N} a_n (-1)^n I_0^n(t^2/2) & \text{for } t < 0 \end{cases} \quad (5.7) \]
CHAPTER 5 GAUSSIAN FILTERS, INVERSE FILTERS AND FILTER SYNTHESIS

Figure 5.3 Comparison of $t^2/(2^n)$ with a Gaussian function.

Figure 5.4 Diagram of Hermite/Laguerre neural network.

Figure 5.4 Diagram of Hermite/Laguerre neural network.
The usefulness of this expansion is that by training the Hermite network to the input function, one also immediately obtains the weights of the Laguerre network, \( \{a_n\} \), which is the correlation of the input with a Gaussian function.

Having both a subscript and a superscript the associated Laguerre functions are more complicated than the Hermite functions. However, for the Gaussian filter, only the associated Laguerre functions of order zero are required, thus greatly simplifying the result. The associated Laguerre functions of subscript zero are

\[
l_n^0(t) = \frac{t^{n/2} \exp(-t/2)}{\sqrt{n!}}
\]  

(5.8)

### 5.3 INTERPRETATION AS GAUSSIAN RADIAL BASIS NEURAL NETWORK

The functions, \( l_n^0(t^2/2) \), are shown in Figure 5.2. For \( n \) larger than 5 they are quite well approximated by the Gaussian function as shown in Figure 5.3 for \( n=8 \), although the Laguerre function is slightly skewed away from the origin.

They shift slowly along the axis and reduce slowly in amplitude as \( n \) increases. The location of the central peak, \( t_n \), and its amplitude, \( A_n \), can be determined by differentiation of \( l_n^0(t^2/2) \) as

\[
t_n = \sqrt{2n}
\]  

(5.9)

and

\[
A_n = \frac{n^{n/2}}{\sqrt{n!}} \exp\left(\frac{-n}{2}\right)
\]  

(5.10)

The variance of the Gaussian approximation remains constant at about \( 1.5/\sqrt{2} \) for all \( n \).
The Gaussian approximation is

\[
I_0\left(\frac{t^2}{2}\right) \cong \frac{n^{n/2}}{\sqrt{n!}} \exp\left(-\frac{n/2}{2.25}\right) \exp\left(-\frac{t - \sqrt{2n}}{2.25}\right) \tag{5.11}
\]

Using the Gaussian approximation, the output Laguerre network is approximately

\[
y_\theta(t) = \sum_{n=0}^{\infty} d_n \frac{n^{n/2}}{\sqrt{n!}} \exp\left(-\frac{n/2}{2.25}\right) \exp\left(-\frac{t - \sqrt{2n}}{2.25}\right) \tag{5.12}
\]

In this form it may be recognized as a Gaussian Radial Base Neural Network (RBNN), though there are some differences. Being slightly skewed, Laguerre functions are not radial symmetric and therefore the network cannot strictly be defined as an RBNN. However the real difference between the present method and an RBNN is in the training of the networks. In an RBNN, the centres of the Gaussian radial functions may be freely chosen and the weights are computed to minimize the mean square error between the network and the training function. With the Laguerre output network, the centres of the radial functions cannot be freely chosen, and are given by Equation (5.9).

Furthermore the weights of the Laguerre network do not minimize the error with the training function (they are optimum with respect to the Hermite input network).

### 5.4 EQUIVALENCE TO KERNEL REGRESSION

The correlation can also be computed using a Monte-Carlo integration given by

\[
y(t) = \frac{T}{I\sigma\sqrt{2\pi}} \sum_{i=0}^{I} x(t_i) \exp\left(-\frac{(t-t_i)^2}{\sigma^2}\right) \tag{5.13}
\]

where \(\{t_i, x(t_i) : i = 0, 1, 2, ..., I\}\) is a discrete data set for the function \(x(t)\) on the interval
\{-2T/2 \leq t \leq 2T/2\}. This summation is the kernel regression in the form given by Priestley and Chao [1972]. In simulations with uniformly sampled data, Figure 5.19, the Hermite/Laguerre neural network achieved an identical signal to noise ratio to the kernel regression of Equation (5.13).

To distinguish the Gaussian filter implemented with the Hermite neural network from kernel regression and all other computational methods of the Gaussian correlation, in the remainder of the thesis it will be called the “Hermite/Laguerre” filter. The difference between kernel regression and the Hermite/Laguerre filter is that the latter has two filtered outputs: the first filtered output is due to the Hermite neural network; the second is due to the Laguerre network. This is shown by Figure 5.4.

5.5 FREQUENCY RESPONSE

In application, the frequency bandwidth of the network is required to ensure that the signal is interpolated correctly without signal loss. The Hermite/Laguerre filter has two filtered outputs. The Hermite output has a bandwidth, \( B \), determined by the number of Hermite functions in the network. From Equation (3.34) this is

\[
B = \sqrt{2N + 1}
\]  

(5.14)

The Laguerre output computes the correlation with the Gaussian function. Its bandwidth is equal to that of the Gaussian function. A Gaussian kernel, \( k(t) \), function has a Fourier transform, \( K(\omega) \), of

\[
K(\omega) = \exp\left(-\frac{\omega^2}{2}\right)
\]

(5.15)

The frequency response of the Gaussian kernel function to a sinusoidal input is determined by approximating \( K(\omega) \) with a Taylor series expansion as follows
\[ K(\omega) \approx \frac{l}{1 + \frac{\omega^2}{2} + 2\omega^4 + \cdots} \approx \frac{l}{(1 + \frac{\omega^2}{2})^2(1 + 2\omega^4 + \cdots)} \]  

(5.16)

where the expansion on the left hand side was obtained using the Taylor series. The Gaussian filter is predominantly 2\textsuperscript{nd} order with a bandwidth, \( B \), given approximately by

\[ B = \sqrt{2} \]  

(5.17)

This approximate bandwidth is obtained by neglecting of powers greater than 2 in the Taylor series expansion.

5.6 FILTER SYNTHESIS

The procedure described to compute the Gaussian filter (Section 5.2) can be extended to include other types of moving average filters besides the Gaussian filter by combining various Hermite functions together. The Hermite functions have a simple Fourier transform which allows the filter to be designed in the frequency domain. From Chapter 3.4 this Fourier Transform (F.T.) is

\[ h_n(t) \xrightarrow{F.T.} \sqrt{2\pi}(-j)^n h_n(t) \]  

(5.18)

This property may be exploited to obtain the time domain coefficients of any filter synthesised in the frequency domain as a summation of Hermite functions. The filtering of a noisy function using the synthesized filter can then be obtained from the correlation of Equation (4.15).

A simple 4\textsuperscript{th} order filter can be designed as follows. The Gaussian filter, \( k(t) \), transforms into the frequency domain as given by Equation 5.16:
CHAPTER 5 GAUSSIAN FILTERS, INVERSE FILTERS AND FILTER SYNTHESIS

\[ K(\omega) = \exp\left(-\frac{\omega^2}{2}\right) = \frac{1}{1 + \frac{\omega^2}{2} + \frac{\omega^4}{4} + \cdots} = \frac{1}{1 + \frac{\omega^2}{2} \left(1 + \frac{\omega^4}{4} + \cdots\right)} \]  

(5.19)

The fourth order filter is derived from the Gaussian filter by multiplication in the frequency domain by the term \((1 + \omega^2/2)\) so that

\[ K_{4th}(\omega) = (1 + \omega^2/2)\exp\left(-\frac{\omega^2}{2}\right) = b_0 h_0(t) + b_2 h_2(t) \approx \frac{1}{1 + \frac{\omega^2}{4}} \]  

(5.20)

where the coefficients \(b_0\) and \(b_2\) of the Hermite functions can be determined by inspection as

\[ b_0 = \frac{5\pi^{3/4}}{4} \]  

(5.21)

and

\[ b_2 = \frac{\sqrt{2\pi^{3/4}}}{4} \]  

(5.22)

The equivalent time domain filter, \(k_{4th}(t)\), using Equation (5.18), is

\[ k_{4th}(t) = \frac{b_0 h_0(t) - b_2 h_2(t)}{\sqrt{2\pi}} \]  

(5.23)

In kernel regression, this filter corresponds to the 4th order Gaussian kernel function described by Wand and Jones [1995, page 32].

The filtered output, \(y_4(t)\), of the noisy function, \(x(t)\), with the 4th order filter is determined from Equation (4.15) as
\[ y_h(t) = \frac{b_0}{\sqrt{2\pi}} \sum_{n=0}^{N} a_n l^n_0 \left( \frac{t^2}{2} \right) - \frac{a_0 b_2}{\sqrt{2\pi}} l^2_0 \left( \frac{t^2}{2} \right) - \frac{b_2}{\sqrt{2\pi}} \sum_{n=2}^{N} a_{n-2} l^n_2 \left( \frac{t^2}{2} \right) \quad (t \geq 0) \]  

with a similar equation for \( t < 0 \). In order to avoid confusion, this 4th order Gaussian filter implemented with the Hermite neural network will be referred to as the synthesized filter.

### 5.7 METHOD OF APPLICATION OF THE HERMITE/LAGUERRE FILTER

A diagram of the neural network, which has a single input and two outputs, is given in Figure 5.4. It consists of two networks, a Hermite network and a Laguerre network. The first output is from the Hermite network. The second output, from the Laguerre network, computes the correlation of the noisy function with the filter function.

The network is trained by minimizing the error between the Hermite network and the training data from the noisy function. The weights of the trained Hermite network are passed directly to the Laguerre network. The purpose of the output Laguerre network is only to provide the correlation. It behaves in the same way as a moving average filter.

Since the weights of the Laguerre network are passed to it from the Hermite network, only the latter needs to be trained.

The weakness of the Hermite/Laguerre filter is that the Gaussian window of the correlation is relatively large compared to the duration of the Hermite interpolation. For example with 50 Hermite activation functions the effective spatial bandwidth of the network is only \( \pm 10 \), whereas the width of the Gaussian window function is \( \pm 3 \).

A method to bypass this weakness, increasing the spatial resolution, is to cascade a series of \( j = 1, 2, \cdots, M \) identical Hermite/Laguerre filter units along the data range by shifting the origin of each filter (Figure 5.5). In this way an acceptably small window
Figure 5.5 Cascaded Hermite/Laguerre neural network
can be achieved. Each individual Hermite/Laguerre filter unit now requires a bias, $\beta_j$, and an output processing element transfer function, $\phi(t)$, given by

$$\beta_j = 10 \ j$$

and

$$\phi(t) = \begin{cases} 
1 & \text{for } -5.0 \leq t \leq 5.0 \\
0 & \text{for } t > 5.0 \\
0 & \text{for } t < -5.0 
\end{cases}$$

The transfer function of the output processing element restricts the output signal of each individual Hermite/Laguerre filter unit so that it does not interfere with the adjacent units. The Hermite and Laguerre cascaded neural network outputs are now, respectively given as

$$x_h(t) = \sum_{j=1}^{M} \sum_{i=1}^{N} a_{ij} \phi(t - \beta_i) h_i(t - \beta_j)$$

and

$$y_h(t) = \begin{cases} 
\sum_{j=1}^{M} \sum_{i=1}^{N} a_{ij} \phi(t - \beta_i) h^n_i(t - \beta_j) & \text{for } t \geq \beta_j \\
\sum_{j=1}^{M} \sum_{i=1}^{N} a_{ij} \phi(t - \beta_i) (-1)^n l^n_i(t - \beta_j) & \text{for } t < \beta_j 
\end{cases}$$

5.8 APPLICATION TO THE FILTERING OF NOISY SINE WAVES

In this section and the next we demonstrate the application of the network to the filtering of noisy sine waves with simulated noise. The sine wave signal used for testing was of unit amplitude and of period, $T=40$, and was sampled at a rate of 10 samples per second. The cascaded network consisted of 5 identical Hermite/Laguerre networks each
Figure 5.6 Frequency response of the Hermite/Laguerre and Synthesized filters.

Figure 5.7 Close up of the frequency response of the Laguerre and Synthesized outputs.
having 50 Hermite and 50 Laguerre activation functions.

Figure 5.6 shows the frequency response of the Hermite/Laguerre filter and the Synthesized filter of Section 5.5. These curves were obtained by inputting a cosine wave of unit amplitude into the network and recording the output amplitude. The frequency bandwidth (3dB) of the Hermite output was in agreement with the theoretical bandwidth given by Equation (5.14). Figure 5.7 shows a close up of the Laguerre and synthesized filter outputs. The bandwidth of the synthesized output is almost double that of the Laguerre output (Gaussian filter) and it has a sharper cut-off. The Laguerre bandwidth, was 40% lower than predicted by Equation (5.17) due to the contribution of the higher order terms in the Taylor expansion. From the graph of Figure 5.10, the actual bandwidth of the Laguerre output is

$$B \cong 0.63\sqrt{2}$$  \hspace{1cm} (5.29)

In order to apply the network to practical problems, it is useful to know the number of activation functions and the number of training cycles required. Figure 5.8 shows the root mean square error of the Hermite and Laguerre networks as a function of the number of activation functions. As the number of activation functions is increased the error of each output decreases and eventually becomes constant. The figure indicates that thirty activation functions are required to ensure that both the Hermite and Laguerre outputs have reached their optimum root mean square error. Figure 5.9 shows the root mean square error of the Hermite and Laguerre networks versus the number of training cycles. The figure shows that more than ten training cycles are required to ensure that the weights have reached their optimum value.
Figure 5.8 Root mean square error versus number of activation functions.

Figure 5.9 Hermite/Laguerre neural network training speed.
Figure 5.10 to Figure 5.13 demonstrate the performance of the Hermite/Laguerre filter on the sine wave without noise. Figure 5.10 is the Hermite output interpolation of the sine wave. The error of the Hermite output, Figure 5.11, occurs in bursts of high frequency oscillations across alternate cascaded elements of the network. Each burst occurs when an asymmetrical portion of the sine wave is interpolated and may be related to the Gibbs phenomenon [Papoulis, 1962, page 30].

Figure 5.12 is the Laguerre output interpolation of the sine wave. Figure 5.13 is the error of the Laguerre output. The small “blips” occurring at regular intervals are caused by the discontinuity at the joints between each cascaded network. At the edges of the data, the error increases sharply. This is due to truncation of the Gaussian filter by the boundary of the data, which is a well-known problem reported with kernel regression [Wand and Jones, 1995, page 46-49]. Due to the sharper cutoff of the Hermite output, it does not suffer from a boundary error to the same extent as the Laguerre output.

For comparison, a sigmoid neural network was also applied to the sine wave. Successful interpolation of the sine wave was possible with a sigmoid neural network consisting of a single layer of five activation functions. This is much smaller in size compared to the Hermite/Laguerre neural network. Training of all the sigmoid neural network weights was with the gradient descent algorithm. The Sigmoidal neural network was considerably slower in training, taking 42 minutes on a Pentium II processor compared to the 3 minutes training time of the Hermite/Laguerre filter. The slow training is due to the considerably larger number of training cycles required by sigmoidal neural network to achieve a satisfactory fit to the data. Figure 5.14 shows the number of training cycles required by the sigmoidal neural network, which is of the order of a 500 cycles.
Figure 5.10 Hermite interpolation of sine wave

Figure 5.11 Hermite error of sine wave interpolation
Figure 5.12 Laguerre interpolation of sine wave

Figure 5.13 Laguerre error of sine wave interpolation
Figure 5.14 Sigmoid neural network training speed.
Figure 5.15 Sigmoid neural network interpolation of sine wave.

Figure 5.16 Sigmoid neural network interpolation of sine wave.
The sigmoid neural network achieved a lower error than both the Hermite and Laguerre networks and did not suffer from a boundary error effect as shown in Figure 5.15 and Figure 5.16. The error of the sigmoid neural network is slightly asymmetrically distributed across the domain, presumably due to its reliance on the gradient descent algorithm, which operates from left to right.

5.9 SIGNAL TO NOISE RATIO OF HERMITE/LAGUERRE FILTER

To investigate the filtering properties of the Hermite/Laguerre filter simulated random noise, of uniform density with a mean value of zero, was added to the sine wave. The random noise was generated artificially from a pseudo-random sequence. Figure 5.17(a) to 5.17(c) demonstrates the operation of the network on the noisy signal. The figure shows the two outputs: the Hermite output (Figure 5.17(b) and the filtered Laguerre network output (Figure 5.17(c)).

The signal to noise ratio ($SNR$) was calculated on the sine wave at the input and output of the network. The input signal to noise ratio, $SNR_{in}$, is defined as the ratio of the signal power without noise to the noise power at the input. The output signal to noise ratio, $SNR_{out}$, is the ratio of the interpolated signal without noise to the noise power at the output.

A theoretical $SNR$ can be derived from the bandwidth of the Hermite/Laguerre filter. Since the simulated random noise is a form of white noise, it is uniformly distributed across the frequency domain. The noise power at the input and output is then proportional to the bandwidth at the input, $\omega_{in}$, and output, $\omega_{out}$, respectively [Lathi, 1983, page 109] and it follows that
Figure 5.17 (a) Noisy sine wave, period=40.0; SNR=7.2.

Figure 5.17 (b) Hermite output; SNR=22.2.

Figure 5.17 (c) Laguerre output; SNR=149.7.
An equivalent noise bandwidth at the input is given in terms of the sampling rate as

\[
\frac{SNR_{out}}{SNR_m} = \frac{\omega_m}{\omega_{out}}
\]  \hspace{1cm} (5.30)

where \( s \) is the sampling rate. At the output, the bandwidth is equal to that of the filter. Since the Laguerre output approximates a Gaussian filter it has the same bandwidth as that of the Gaussian filter given by Equation (5.29). The Hermite output has the bandwidth given by Equation (5.14). Figure 5.18 compares the SNR of the Laguerre output and the Hermite output for different input noise power. A reasonable fit to the theoretical SNR of Equation (5.30) is achieved in both cases.

The Hermite output deviates from a straight line to a lesser extent than the Laguerre output, probably due to the sharper cut off of the Hermite network (Figure 5.10). An improvement in SNR by a factor of ten is achieved with the Laguerre output.

Figure 5.19 shows the SNR obtained using the kernel regression of Section 2.3 compared to that of the Hermite/Laguerre network. The identical noise performance to the Laguerre output confirms the equivalence between each network.

Figure 5.20 shows the SNR obtained with the synthesized filter compared with the Laguerre filter. The reduction in the signal to noise ratio is due to the wider bandwidth of the synthesized filter.

An upper limit to the filtering performance of the Laguerre output can be obtained by comparison with a Weiner filter (Figure 5.21). The Weiner filter [Champeney, 1975,
page 131-135] is the theoretical optimum filter for a sine wave, which is achieved when the filter function is itself a sine wave. The Weiner filter shown in Figure 5.21 consisted of a sine wave of the same frequency as the noisy sine wave but with a phase and amplitude determined by numerical integration.

The noise present in the Weiner filter is due to phase noise, amplitude noise and numerical computational errors associated with the integration. Although the Weiner filter has a higher SNR than the Laguerre output, it requires the signal to be known prior to filtering.

Figure 5.22 shows the SNR obtained with the sigmoidal neural network compared to that of the Hermite/Laguerre filter. The Laguerre output achieved approximately a 50\% higher SNR compared to the sigmoid neural network. The results concerning the sigmoid neural network are not unique because there are many different possible ways of applying this type of network. For example a multi-layer sigmoid neural network is likely to have a lower SNR. The results for the sigmoid neural network are the author’s attempts to achieve the typical performance with this type of network.

5.10 GAUSSIAN INVERSE FILTERS

In the remainder of this chapter the Hermite/Laguerre neural network will be investigated as an inverse Gaussian filter. The Gaussian filter has already been defined in Section 5.1. The Gaussian filter operating on a signal \( x(t) \), produces the blurred or distorted signal \( y(t) \) by the correlation

\[
y(t) = x(t) * k(t) \tag{5.32}
\]
Figure 5.18 Laguerre and Hermite output SNR compared with theory.

Figure 5.19 Comparison of SNR with kernel regression
Figure 5.20 Comparison of SNR with synthesized filter

Figure 5.21 Comparison with the Weiner filter.
Figure 5.22 Comparison of SNR with sigmoid neural network

Figure 5.23 Inverse Gaussian filter composed of Hermite/Laguerre neural networks.
where $k(t)$ is the Gaussian function

$$k(t) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{t^2}{2\sigma^2}\right)$$  \hspace{1cm} (5.33)$$

An inverse Gaussian filter is defined to be a filter, which recovers the signal $x(t)$ from the blurred signal $y(t)$. In order to simplify the mathematics, it will be assumed that the data has been scaled such that $\sigma = 1$. The same assumption was made in the application as a Gaussian filter (Section 5.1).

5.11 FOURIER INVERSE FILTER

The Fourier series expansion of the blurred signal and the Gaussian blur function using complex notation is respectively

$$y_\Theta(t) = \sum_{n=0}^{N} c_n e^{j2\pi n/T}$$  \hspace{1cm} (5.33)$$

and

$$k_\Theta(t) = \sum_{n=0}^{N} d_n e^{j2\pi n/T}$$  \hspace{1cm} (5.35)$$

where $c_n$, are the weights of the blurred signal. $d_n$ are the weights of the Gaussian blur function, which are real on account of its symmetry. The de-blurred signal is obtained from $y_\Theta(t)$ by dividing by the Gaussian blur function, $k_\Theta(t)$, in the frequency domain and taking the inverse Fourier transform. The resulting Fourier series is

$$x_\Theta(t) = \sum_{n=0}^{N} \left(\frac{c_n}{d_n}\right) e^{j2\pi n/T}$$  \hspace{1cm} (5.36)$$
Because the Gaussian function, $k(t)$, is isomorphic with respect to the Fourier transform, its weights, $d_n$, are themselves the discrete samples of a Gaussian function given by

$$d_n = \exp\left(-\frac{\sigma^2}{2} \left(\frac{m}{T}\right)^2\right) \quad (5.37)$$

They rapidly decrease in magnitude as $n$ increases. Consequently a point is reached in the Fourier expansion, $x_\omega(t)$, where it becomes unstable due to numerical error present in the computation.

A different situation occurs if instead of using the Fourier series we choose an orthogonal series in which the blur function itself is a member of the series. In this case the de-blurring operation does not involve a division by a series of decreasing weights. This is the principle behind the numerical stability of the Hermite/Laguerre inverse filter described in the next section.

### 5.12 HERMITE/LAGUERRE INVERSE FILTER

Consider the blurring of the signal, $x(t)$, by the Gaussian function with both functions expanded as Hermite series. The Gaussian blur function is represented by itself in the Hermite series as

$$k(t) = a_0 h_0(t) \quad (5.38)$$

where the constant $a_0 = 1/\sqrt{2\pi}$. The Hermite series expansion of $x(t)$ is

$$x_n(t) = \sum_{n=0}^{N} a_n h_n(t) \quad (5.39)$$
where \( a_n \) are the network weights, which will be derived later. Gaussian blurring produces the blurred signal, \( y(t) \), by the correlation

\[
y(t) = x(t) * k(t) \equiv a_0 \sum_{n=0}^{N} a_n h_n(t) * h_0(t)
\]

This equation can be re-written in a much more simplified form by replacing the Hermite correlation terms with their direct mathematical form (Equation (5.6)). \( y(t) \) then simplifies to a summation of associated Laguerre functions

\[
y(t) \equiv y'_t(t) = \begin{cases} 
  a_0 \sum_{n=0}^{N} a_n I^n(t^2/2) & \text{for } t \geq 0 \\
  a_0 \sum_{n=0}^{N} a_n (-1)^n I^n(t^2/2) & \text{for } t < 0 
\end{cases}
\]

Apart from the constant, \( a_0 \), the weights of the blurred signal expanded as a Laguerre series are the same as the Hermite series expansion of the de-blurred signal of Equation (5.39). The procedure of obtaining the de-blurred signal is then to fit the Laguerre series to the blurred signal to obtain the weights \( a_n \). The de-blurred signal, \( x(t) \), is obtained from these weights via the Hermite series. It is evident that the procedure of recovering the de-blurred signal does not involve a division from a series of successively smaller factors. Therefore it is numerically stable.

### 5.13 TRAINING THE HERMITE/LAGUERRE INVERSE FILTER

In order to apply the inverse filter, the weights \( a_n \) of the associated Laguerre network must be fitted to the blurred signal. Associated Laguerre functions are not orthogonal with respect to the superscript. In this respect, at first glance, it would seem to be a difficult task to obtain these weights. Fortunately associated Laguerre functions are...
approximately orthogonal, which can be deduced from their approximate Gaussian form of Equation (5.11). The orthogonal condition can be stated as

\[ \int I_n^m(t^2/2)I^m_n(t^2/2)dt \equiv 0 \quad \text{for} \ n \neq m \] (5.42)

Using this orthogonal condition, a simple approximate solution can be derived, which can then be optimized with respect to the mse using the LMS algorithm. Multiplying both sides of Equation (5.41) by \( I_n^m(t^2/2) \), integrating with respect to \( t \) and re-arranging terms gives the approximate solution for the amplitudes as

\[ a_m \approx \frac{\int y(t)I^m_n(t^2/2)dt}{\int I^m_n(t^2/2)^2 dt} \] (5.43)

The LMS algorithm to train these weights takes the form

\[ a_m(t_{i+1}) = a_m(t_i) - \alpha \left( y(t_i) - \sum_{n=0}^{N} a_n(t_i)I^n_0(t_i^2/2) \right) I^m_0(t_i^2/2) \] (5.44)

5.14 EFFECT OF THE LMS ALGORITHM ON THE FREQUENCY RESPONSE OF THE HERMITE/LAGUERRE INVERSE FILTER

Figure 5.23 shows a schematic drawing of the inverse filter. At the input of the filter, the frequency response is due to the Laguerre network. Consider first the frequency response without the LMS algorithm. Using the Gaussian approximation of Equation (5.11) for the Laguerre function, the Fourier transform of the associated Laguerre function, \( F\{I^n_0(t^2/2)\} \), is approximately a Gaussian amplitude modulated cosine or sine wave given by

\[ F\{I^n_0(t^2/2)\} \approx e^{-\omega^2/2} \cdot e^{-j\omega t} \] (5.45)
Figure 5.24 Frequency response of Laguerre input of the inverse filter as a function of the number of training cycles of the LMS algorithm from 0 to 100 in steps of 10 cycles.

Figure 5.25 Frequency response of Hermite output of inverse filter as a function of the number of training cycles of the LMS algorithm.
The Gaussian envelope restricts the response of the Laguerre network to within the envelope. Therefore at the input the bandwidth is approximately the width of this envelope, which is

\[ B_{in} = \sqrt{2.25} \quad (5.46) \]

The frequency response at the output is due to the Hermite network alone. Therefore the bandwidth at the output is fixed by the duration of the highest order Hermite function in the series, according to

\[ B_{out} \approx \sqrt{2N + 1} \quad (5.47) \]

For networks that are practically useful \( N \) is quite large. Hence the bandwidth at the output is much greater than at the input and the Hermite output has a negligible effect on the filter.

The application of the LMS training algorithm to the Laguerre network considerably alters the frequency response characteristics of the inverse filter. The LMS algorithm extends the frequency response of the Laguerre network by increasing the amplitudes \( a_n \), of the Laguerre functions at high frequency to offset the drop in response of the Laguerre functions at these frequencies.

Figure 5.24 shows the frequency response of the Laguerre network as a function of the number of LMS training cycles. As the LMS algorithm is applied, the bandwidth of the Laguerre network gradually increases.

At the Hermite output of the inverse filter, the overall effect of the LMS algorithm is that the Hermite output gradually takes on the form of a truncated inverse Gaussian function (Figure 5.25). In mathematical form, the inverse filter can then be regarded as
being roughly equivalent to an ideal filter in series with a Gaussian inverse function. The bandwidth of the ideal filter component at which the truncation occurs is dependent on the number of LMS training cycles. We define the cut-off point as the frequency at which the peak amplitude is obtained.

The dependence of the bandwidth of the ideal filter component on the number of LMS training cycles provides a convenient method of adjusting the filter to cope with different amounts of signal noise. For low noise signals, little or no filtering is required which is achieved when a large number of LMS training cycles are applied. Conversely for more noisy signals, a filter with a low bandwidth is required, which is achieved with a low number of training cycles. Figure 5.26 shows the cut-off frequency as a function of the number of LMS training cycles from which an appropriate filter bandwidth may be chosen.

5.15 PSEUDO-WEINER FILTER

A benchmark for the inverse filtering of noisy signals is the classical Weiner filter applied in conjunction with the Fourier series [Pratt, 2001, page 329-333]. It is therefore appropriate to compare the Hermite/Laguerre filter with the Weiner filter. The Weiner filter is the theoretical optimum filter for the minimization of the \( \text{mse} \) between the de-blurred signal and a linear inverse filter of the blurred signal. In the frequency domain, the Weiner filter is

\[
W(\omega) = \frac{K(\omega)}{K(\omega)^2 + N(\omega)/X(\omega)}
\] (5.48)

where \( W(\omega) \), \( K(\omega) \) and \( N(\omega) \) are the Fourier transforms of the Weiner filter, Gaussian filter and noise respectively.
Figure 5.26 Cut-off frequency of the Hermite/Laguerre inverse filter versus the number of LMS training cycles.

Figure 5.27 Comparison of Hermite/Laguerre inverse filter with the pseudo-Weiner inverse filter.
To achieve the Weiner filter requires the original signal to be known prior to filtering. Obviously in general application it will be unknown until after filtering. To deal with this, the pseudo-Weiner filter has been developed [Petrou and Bosdogianni, 1999, page 219]. The pseudo-Weiner filter replaces the unknown signal by a constant, $\gamma$, called the noise desensitizing factor. The Weiner filter then becomes

$$W(\omega) = \frac{K(\omega)}{K(\omega)^2 + \gamma}$$

It should be emphasized that the pseudo-Weiner filter is not the true optimum filter but merely an approximation. However it does produce acceptable results in application and it is simple to apply and understand.

Figure 5.27 shows the pseudo-Weiner filter compared with the Hermite/Laguerre inverse filter. Although the Hermite/Laguerre inverse filter resembles the pseudo-Weiner filter, there are some notable differences. The Hermite/Laguerre inverse filter follows the Gaussian inverse filter up to a point just prior to the cut-off where it drops off sharply. The pseudo-Weiner filter begins to drop away from the Gaussian inverse filter much further away from the cut-off point.

### 5.16 DE-BLURRING OF LASER DOPPLER ANEMOMETRY

The Hermite/Laguerre inverse filter was tested against the pseudo-Weiner and Fourier inverse filters by application to the de-blurring of a laser Doppler anemometer. Laser Doppler anemometers are an important instrument for the non-invasive measurement of fluid flow; see Crossway and Hornkohl [1973] for a simplified explanation of their operation and [Edwards, et al., 1971] for a detailed analysis. In some applications, measurements are required in flow systems of extremely small dimensions, of the order
Figure 5.28 Distorted velocity profile from laser Doppler anemometer.

Figure 5.29 Displacement of velocity profile from data center line
of a few hundred micrometers [Cochrane and Earnshaw, 1978]. Examples of some of these applications include boundary layer flow [Mazumder et al., 1981], in-vitro vascular flow studies [Mishina et al., 1975] and hydrodynamic bearings [Tieu et al., 1994]. At these small dimensions, the finite parameters of the laser optical system introduce a significant Gaussian distortion into the data, which needs to be corrected. In view of the simple mathematical form of the measurement data obtained from the laser Doppler anemometer and the fact that it is well understood theoretically, it is an ideal system with which to test the hybrid filter.

Laminar fluid flow in a small tube is parabolic, described by the mathematical equation [Currie, 1974, pp. 31-32]

\[
y(t) = \begin{cases} 
(1 - 4t^2/L^2) & \text{for} \ |t| \leq L \\
0 & \text{for} \ |t| > L
\end{cases}
\]  

(5.50)

For fluid flow, \( t \) describes the location from the centerline of the tube, \( y(t) \) is the velocity at this location and \( L \) is the diameter of the tube. A discontinuity occurs at the wall of the tube where the velocity abruptly changes in magnitude. The distortion introduced by the laser assumes the form of Equation (5.1), where \( x(t) \) is the distorted velocity. Figure 5.28 shows the velocity profile and the distorted profile. The parameters chosen for the investigation were \( \sigma = 1 \text{mm} \) and \( L = 20 \text{mm} \). The distorted velocity measurement data was obtained by a direct numerical correlation with the Gaussian distortion. Unless stated otherwise, the Hermite/Laguerre inverse filter was chosen to have 150 activation functions.

Fourier, Weiner and Hermite/Laguerre inverse filters of different network size were applied to obtain the de-blurred velocity. Figure 5.30 shows the numerical stability of
Figure 5.30 (a) to (i) Numerical stability of inverse filters as the network size is increased.
Figure 5.31 (a) to (i) Off-axis numerical stability of inverse filters.
each inverse filter, as the network size is increased. With a network consisting of 30 activation functions, all inverse filters were stable. At 60 activation functions, the Fourier inverse filter shows a pronounced fluctuating error that completely masks the velocity. For network of all size, the Hermite/Laguerre inverse filter achieved the same numerical stability as the Weiner inverse filter.

Training of the Laguerre network consisted of 100 cycles of the LMS algorithm. Figure 5.31 shows the numerical stability of the inverse filters as a function of the displacement of the velocity profile from the center of the inverse filter (Figure 5.29). All filters were stable when the inverse filter was aligned with the origin of the velocity profile. As the off-axis displacement was increased, the Fourier inverse filter became unstable even for moderate displacements. At large off-axis displacements, the Hermite/Laguerre filter achieved greater numerical stability than the Weiner filter.

The effect of noise, artificially generated noise from a pseudo-random sequence, for the Hermite/Laguerre and Weiner inverse filters is currently under investigation by the author. Preliminary results indicate that the pseudo-Weiner inverse filter suffers from distortion at high noise levels where a low cut-off frequency must be used. The Hermite/Laguerre inverse filter does not suffer from this distortion. However it is of limited bandwidth compared to the pseudo-Weiner inverse filter.

5.17 CONCLUSION

A neural network was developed for correlation with a Gaussian function using the Hermite orthonormal functions. The correlation of a Hermite function with a Gaussian function results in a zero order associated Laguerre function (Section 5.1). Consequently by training a Hermite network to an input function the correlation output
is also immediately available via the Laguerre network. The name “Hermite/Laguerre” filter was coined to describe this method of computation of the Gaussian filter. In relation to other types of neural networks, the Hermite/Laguerre filter may be interpreted as a specialized type of RBNN with training by Hermite orthonormal functions.

The network behaves as a Gaussian filter. It has two outputs: the first output approximates the noisy signal; the second output represents the filtered signal. The filtering provided by the Laguerre output produces a factor of ten greater SNR compared to the Hermite output. Good agreement with theoretical SNR was achieved (Figure 5.19).

Besides the Gaussian filter, a 4\textsuperscript{th} order filter was synthesized using a combination of two Hermite functions. This type of filter, which was called the synthesized filter, offers greater frequency response and a sharper cut-off than the Gaussian filter at the expense of a lower SNR.

When compared with a sigmoid neural network, the Hermite/Laguerre Gaussian filter achieves an approximately 50% higher SNR for noisy sine waves. The disadvantages are that it suffers from a boundary error and is not as compact as a sigmoid neural network.

The boundary error is noticeable in Figure 5.13 where there is a rapid increase in the error at the boundaries. The sigmoid neural network does not appear to be effected by the boundary error (Figure 5.16). The boundary error of the Hermite/Laguerre Gaussian filter will be investigated further in the next chapter, where a method of correcting the boundary error will be developed using asymmetrical kernel regression (Chapter 6).

Although the sigmoid neural network has a clear advantage over the Hermite/Laguerre
neural network in that it is of a very compact size it proved to be extremely slow in training. In Chapter 8, a method of training the sigmoid neural network at competitive speeds is developed. The idea being to combine both the sigmoid neural network and the Hermite/Laguerre neural network together. The Hermite/Laguerre neural network would provide the filtering followed by the sigmoid neural network to interpolate the filtered output into a compact form.

This chapter also developed an inverse Gaussian filter based on the Hermite/Laguerre neural network for the removal of Gaussian blur from a signal. The effect of the LMS algorithm on the frequency response characteristics of the inverse filter was established. It was found that by varying the number of LMS training cycles the bandwidth of the inverse filter could be controlled. The LMS algorithm was then applied in such a way that the bandwidth could be adjusted to suit different noise levels. It was demonstrated that the proposed Hermite/Laguerre inverse filter offered greater numerical stability than both the Fourier inverse filter and the pseudo-Weiner filter.
CHAPTER 6

ERROR ANALYSIS

OF THE

HERMITE/LAGUERRE FILTER

6.1 INTRODUCTION

This chapter describes a statistical model for predicting the error associated with the new type of Gaussian filter, the Hermite/Laguerre filter, which was developed in Chapter 5. The notation and other relevant statistical results are taken directly from [Papoulis, 1965].

It is shown that the Hermite/Laguerre filter is equivalent to kernel regression. Based on this result, kernel regression analysis is directly applied to predict the error of the Hermite/Laguerre filter. Kernel regression itself has been extensively studied in the past providing a simple well developed model for the Hermite/Laguerre filter.

At the edges of the data the error requires special attention. In this region the kernel function is truncated by the boundary complicating the analysis. The term, “boundary error” will be used to describe this type of error. A new method of correcting this boundary error for kernel regression is developed based on an asymmetric kernel function. This is the first step towards the development of a boundary error correction
6.2 EQUIVALENCE BETWEEN KERNEL REGRESSION AND THE

HERMITE/LAGUERRE FILTER

In this section it will be proven that when the Hermite/Laguerre filter, \( y_h(t) \), operates on discrete training data it is approximately equivalent to a kernel regression, \( y_k(t) \). That is, it will be shown that

\[
y_h(t) \approx y_k(t) = \frac{T}{I} \sum_{i=0}^{T-I} x_i k(t_i - \tau)
\]

where the Hermite/Laguerre output, Equation (5.5), is

\[
y_h(t) = x_h(t) \ast k(t) = \int_{-\infty}^{+\infty} x_h(\tau) k(t - \tau) d\tau
\]

and

\[
k(t) = \frac{\exp\left(-\frac{t^2}{2}\right)}{\sqrt{2\pi}}
\]

The discrete training data given in Equation (6.1) is \( \{t_i, x_i : i = 0, 1, 2, ..., I - 1\} \) where \( x_i = x(t_i) \) and \( t_i \) is defined on the interval \( \{-T/2 \leq t \leq T/2\} \). The kernel regression summation, \( y_k(t) \), on the left-hand-side of Equation (6.1) is in the form first proposed by Priestley and Chao [1972] (see Section 2.5).

Consider the Hermite interpolation of \( x_h(t) \), which is

\[
x_h(t) = \sum_{j \in d} a_j h_j(t)
\]
where the weights, $a_n$, of the Hermite neural network are obtained from the discrete training data by Equation (3.24)

$$a_n = \frac{T}{I} \sum_{i=0}^{I-1} x_i h_n(t_i). \quad (6.5)$$

Substituting Equation (6.5) into Equation (6.4) leads to the double summation

$$x_h(t) = \frac{T}{M} \sum_{n=0}^{M-1} \sum_{i=0}^{I-1} x_i h_n(t_i) h_n(t) \quad (6.6)$$

This can be simplified by using the Christoffel-Darboux [Abramowitz and Stegun, 1965] formula for the summation of Hermite functions, which is

$$\sum_{n=0}^{N} h_n(t_i) h_n(t) = \frac{h_{N+1}(t_i) h_N(t) - h_N(t_i) h_{N+1}(t)}{t_i - t} \quad (6.7)$$

For large $N$, the terms on the right-hand-side of the Christoffel-Darboux formula approximate a delta function, that is

$$\sum_{n=0}^{N} h_n(t_i) h_n(t) \xrightarrow{N \gg 0} \delta(t - t_i) \quad (6.8)$$

Substituting this delta approximation into Equation (6.6) reduces the double summation to

$$x_h(t) = \frac{T}{I} \sum_{i=0}^{I-1} x_i \delta(t - t_i) \quad (6.9)$$

Using this expression for $x_h(t)$ in the integral Equation (6.2) for the output of the Hermite/Laguerre filter removes the Hermite function from the integral so that
\[ y_h(t) = \frac{T}{I} \sum_{i=0}^{I} x_i \delta(t-t_i)k(t-t_i)dt \]  

(6.10)

By evaluating the integration it can be seen that

\[ y_h(t) = \frac{T}{I} \sum_{i=0}^{I-1} x_i k(t-t_i) \]

(6.11)

which completes the proof.

### 6.3 ERROR OF HERMITE/LAGUERRE FILTER

The equivalence of the Hermite/Laguerre filter with kernel regression will be applied in this section to estimate the mean square error (\(mse\)) of the Hermite/Laguerre filter. The Hermite/Laguerre filter interpolates the data \{\(t_i, x_i; i = 0, I, 2, ..., I - I\}\}. The \(mse\) of the Hermite/Laguerre filter is defined to be the error between the actual function \(x(t)\) and the Hermite/Laguerre filter \(y_h(t)\):

\[ mse[y_h(t)] = E[(x(t) - y_h(t))^2] \]

(6.12)

From the equivalence with kernel regression shown in Section 6.2, this \(mse\) is

\[ mse[y_h(t)] \equiv mse[y_k(t)] = E[(x(t) - y_k(t))^2] \]

(6.13)

where \(y_k(t)\) is the kernel regression output

\[ y_k(t) = \frac{T}{I} \sum_{i=0}^{I-1} x_i k(t-t_i) \]

(6.14)

The \(mse\) can be separated into a bias error and variance error as
CHAPTER 6 ERROR ANALYSIS OF THE HERMITE/LAGUERRE FILTER

\[ mse[y_k(t)] = E[(x(t) - y_k(t))^2] = var[y_k(t)] + (bias[y_k(t)])^2 \]  
\[ (6.15) \]

where the first term is the variance error which is

\[ var[y_k(t)] = E[(y_k(t) - E[y_k(t)])^2] \]  
\[ (6.16) \]

and the second term is the bias error which is

\[ bias[y_k(t)] = E[x(t) - y_k(t)] \]  
\[ (6.17) \]

The bias and variance error will now be examined in more detail. We first consider the case when the data is free of added noise. Since there is no added noise, \( E[y_k(t)] = y_k(t) \) and there is no variance error. The bias error is

\[ bias[y_k(t)] = x(t) - \frac{T}{I} \sum_{i=0}^{I-1} x_k(t_i - t) \cong x(t) - \int_{-\infty}^{\infty} x(t_k) k(t_i - t) dt_i \]  
\[ (6.18) \]

The magnitude of the bias error can be estimated by using a Taylor series expansion for \( x(t_i) \) about \( t_i = t \) in the integration so that

\[ bias[y_k(t)] \cong \frac{\sigma^2}{2} x''(t) + \frac{\sigma^4}{8} x'''(t) \]  
\[ (6.19) \]

Note that the odd terms of the Taylor series disappear due to the symmetric nature of the kernel function and do not contribute.

To check the validity of the bias error of Equation (6.19), the sine wave of Section 5.8 was again investigated using the same Hermite/Laguerre filter. Figure 6.1 shows the bias error of Equation (6.19) and the actual error of the Hermite/Laguerre filter. Excellent agreement is obtained except towards the edges of the data \( t = \pm 20 \) where it is inaccurate. The error at the edges of the data will be dealt with in Section 6.4. The
Figure 6.1 Bias error of Hermite/Laguerre filter compared with theory.
kinks at \( t = -5 \) and \( t = 15 \) are due to a slight discontinuity at the junction between the cascaded Hermite/Laguerre network components of the filter.

The effect of noise on the \( mse \) of the Hermite/Laguerre filter will now be considered. Two types of noise occur: phase noise and amplitude noise. Phase noise occurs when the variables \( t_i \) are random. An example being the noise that is generated when the random variables \( t_i \) are equally likely to occur anywhere across the data range \( \{-T \leq t_i \leq T\} \). Amplitude noise occurs when a random variable \( n \) is added to \( x_i \) such that

\[
x_i \rightarrow x_i + n
\]  

(6.20)

Only amplitude noise will be considered in this thesis. Furthermore it will be assumed that the amplitude noise has a mean value \( E[n] = 0 \).

With \( E[n] = 0 \) the bias error for amplitude noise remains unchanged from that given in Equation (6.26). The variance error is

\[
var[y_k(t)] = E[n^2] \frac{T^2}{I^2} \sum_{i=0}^{I-1} k(t - t_i)^2
\]  

(6.21)

Approximating the summation by an integration gives

\[
var[y_k(t)] = \frac{E[n^2]T^2}{I^2} \int_{-\infty}^{\infty} k^2(t)dt
\]  

(6.22)

Evaluating the integral for a Gaussian kernel function gives

\[
var[y_k(t)] = \frac{TE[n^2]}{I^2\sigma\sqrt{\pi}}
\]  

(6.23)
The noise performance of the Hermite/Laguerre filter has already been shown in tests to be identical to kernel regression (Figure 5.19). This confirms the validity of Equation (6.23).

### 6.4 BOUNDARY ERROR

The boundary error theory of kernel regression also applies directly to the Hermite/Laguerre filter. Boundary error occurs when the kernel function is truncated by the edge of the data (Figure 6.1). This section summarizes the boundary error and boundary error correction techniques developed for kernel regression.

Although unnoticeable in Parzen’s [1962] original paper, which describes the kernel function estimation of density, the method is defined as being applicable to data on the interval \( \{-\infty < t < \infty\} \). In practical application of kernel regression the available data imitates a causal function since it only occurs across a finite interval and is usually taken to be zero for \( t<0 \). The boundary is defined as \( t=0 \). Significant bias error occurs in the vicinity of \( t=0 \) because the kernel function odd moments are no longer zero and these must now be included in the bias error Equation (6.19). The bias error close to \( t=0 \) is:

\[
\text{bias}[y, \tau (t)] = x(t) \int_0^\infty k(\tau - t) d\tau + x'(t) \int_0^\infty (\tau - t) k(\tau - t) d\tau + \\
\frac{1}{2} x''(t) \int_0^\infty (\tau - t)^2 k(\tau - t) d\tau + \cdots
\]

(6.24)

Extensive studies have been undertaken in statistics regarding the boundary error for kernel regression. A number of different numerical schemes have been developed to cope with the boundary error.
Figure 6.2 Gaussian kernel regression showing truncation of the kernel at the boundary.

Figure 6.3 Correction of boundary bias error by reduction of the variance of the Gaussian kernel close to the boundary.
These can be found in the work on kernel density estimation where many have been summarized by Zhang et al [1999]. Probably the simplest method (mentioned in the paper of Hall and Wehrly [1991]) is to reduce the window width of the Kernel function in proportion to its distance from the boundary (Figure 6.3). Although it is very effective in reducing the bias error it can cause a significant increase in the random noise, which is inversely proportional to the window width of the kernel function. For this reason it is probably not practical; but it highlights the fundamental problem with boundary bias correction, which is to correct the bias error without adversely affecting the noise level performance.

Other boundary error correction techniques include, using specialized boundary kernels at the boundary, reflection of the data about the boundary, and transformation of the data. The boundary kernel method uses linear multiples of the Kernel function in the vicinity of the boundary, which are specially tailored to reduce the bias error (Gasser and Muller [1979]). The locally linear kernel regression estimator of Fan and Gijbels [1992] behaves in a similar way. It has since been reported that the boundary kernel method also suffers from an increase in the variance error. The reflection method of Hall and Wehrly [1991] provides reduced boundary bias error with a low variance error. Data is reflected about the boundary to prevent the Kernel from being truncated by the boundary. Whilst boundary bias error is improved the bias error will still be larger than in the interior when the data has a non-zero first derivative at the boundary. Marron and Ruppert [1994] apply a transformation to the data so that it has a first derivative equal to zero prior to applying the reflection method.

In an effort to provide boundary error correction without compromising the variance error, Zhang et al [1999] combine the transformation method and reflection method
with the pseudodata method of Cowling and Hall [1996]. Further practical application will determine the usefulness and weaknesses of this combined technique.

### 6.5 ASYMMETRICAL KERNEL REGRESSION

In this section a method of correcting the boundary error for kernel regression is proposed and investigated using asymmetrical kernel regression. In future work the results of this study will be incorporated into the design of a Hermite/Laguerre filter with boundary error correction.

The Gaussian function is well known in statistics as the limiting probability density of a sum of samples which each have identical probability densities defined on the interval \(-\infty < t < \infty\). For samples, which have probability densities defined on the interval \(0 < t < \infty\), called causal probability densities, the limiting probability density is the Gamma density [Papoulis, 1962, page 235]. The Gamma density is naturally asymmetric to cope with the discontinuity at \(t=0\). In view of the causal nature of the data we replace the symmetric Gaussian function by a Gamma density in the vicinity of \(t=0\).

The Gamma density (Figure 6.4) is defined as

$$g(t) = \frac{t^{\alpha} \exp(-t/\beta)}{\beta^\alpha \Gamma(\alpha + 1)}$$  \hspace{1cm} (6.25)

where \(\Gamma(\alpha + 1)\) is the factorial function and \(\alpha, \beta \geq 0\). The parameters \(\alpha\) and \(\beta\) define the shape and location from the origin of the Gamma density. The mean value, variance and location of the peak of the Gamma density are defined in terms of \(\alpha\) and \(\beta\) respectively by
Figure 6.4 Gamma kernel functions.

Figure 6.5 Beta kernel functions.
Due to the asymmetry, the mean value is shifted to the right of the peak value of the Gamma density.

For the Gaussian kernel function, the window width is equal to its variance and the mean value centered at $t = \eta$ (Figure 6.2) defines the location of the estimate $y_g(\eta)$ of the function $x(\eta)$. In the same way we may define the window width of the Gamma kernel function to be equal to its variance and the mean value, $\eta$, of the Gamma function as the location of the estimate $y_g(\eta)$ of the function $x(\eta)$. Therefore for a particular $\eta$ and $\sigma$ (defined by Equations (6.26) and (6.27) in terms of $\alpha$ and $\beta$) the asymmetric kernel regression using the discrete data $\{t_i, x_i : i = 0,1,2..., I - 1\}$ of $x(t)$ on the interval $\{0 \leq t \leq T\}$ is

$$y_g(\eta) = \frac{T}{I} \sum_{i=0}^{I-1} x_i g(t_i, \eta, \sigma)$$

(6.29)

where $x_i = x(t_i)$. In order to preserve the noise performance close to the boundary, the variance, $\sigma$, of the Gamma density must be kept constant for different locations of the mean value, $\eta$. This can be achieved by forcing $\alpha$ and $\beta$ to satisfy the following relations:
\[ \beta = \frac{\sigma^2}{\eta} \] (6.30)

and

\[ \alpha = \frac{\eta^2}{\sigma^2} - 1 \] (6.31)

Figure 4(a) shows the Gamma density with \( \sigma = 0.1 \) for various values of the mean value, \( \eta \). As \( \eta \) increases, the asymmetry of the Gamma density slowly disappears and with \( \eta \) large enough, the Gamma density can be approximated by the Gaussian function.

As \( \eta \) decreases the Gamma density becomes more and more asymmetric until eventually at \( \eta = \sigma \) (corresponding to \( \alpha = 0 \)), the Gamma density becomes an exponential. This is the closest distance to the boundary that is possible with the variance held constant.

The bias error of the Gamma kernel regression is

\[
\text{bias}[y_s(\eta)] = x(\eta) + \frac{\sigma^2}{2} x''(\eta) + \frac{\sigma^4}{3\eta} x''(\eta) + \frac{\sigma^4}{8} \left( 1 + 2 \left( \frac{\sigma}{\eta} \right)^2 \right) x''(\eta) + \cdots
\] (6.32)

Immediately apparent is the absence of a boundary error term. Therefore the bias error, is independent of the distance from the boundary. Also apparent is that the odd terms of order 3 of the Taylor series must now be included due to the skewed nature of the asymmetric Gamma density. Consequently, at interior points the bias error will be slightly larger than the Gaussian regression. For large \( \eta \), the asymmetry disappears and the bias error will approach the error of the Gaussian function.
The variance error of the Gamma density is
\[
\text{var}[y_g(\eta)] \equiv \frac{TE[n^2]}{2\sigma t \sqrt{\pi}} \quad (6.33)
\]

Because the variance of the Gamma density has been held fixed, this error is also independent of the distance from the boundary.

In the limit of large \(I\), the asymmetric Gamma kernel regression approaches the integral
\[
y_g(\eta) \equiv \lim_{I \to \infty} \frac{T}{I} \sum_{i=0}^{I} x_i g(t_i, \eta, \sigma) = \int_0^T x(t)g(t, \eta, \sigma)dt \quad (6.34)
\]

which is not a convolution operator. However for large \(\eta\), the Gamma density is approximately equal to a Gaussian function and the bandwidth of Equation (5.18) is appropriate.

The procedure described so far is suitable for a boundary on the left at \(t = 0\). For a boundary on the right at \(t = T\), the Gamma density can still be applied by change of variables to
\[
t \to T - t \quad (6.35)
\]

Alternatively the Beta density can be used. The Beta density (Figure 6.5) is the limiting probability density for a sum of samples each having identical probability densities on a finite interval [Papoulis, 1962, page 238]. The Beta density, \(b(t)\), on the interval \(\{0 \leq t \leq I\}\) is
The variance and mean value of the Beta density, for a particular $\alpha$ and $\beta$, are obtained by the solution of the simultaneous equations

$$\alpha + 1 = \frac{\eta(\eta - \eta^2 - \sigma^2)}{\sigma^2}$$  \hspace{1cm} (6.37)$$

and

$$\beta + 1 = \frac{(1-\eta)(\eta - \eta^2 - \sigma^2)}{\sigma^2}$$  \hspace{1cm} (6.38)$$

Beta kernel regression is applied in a similar way to the Gamma kernel regression. An appropriate variance, $\sigma$, is chosen based on the interior data at points well away from the boundaries. To preserve the performance against noisy data this is kept constant for each location, $\eta$, across the full range of the data. The particular Beta function to use at a location, $\eta$, is obtained from Equations (6.37) and (6.38). Data on a different interval to $\{0 \leq t \leq 1\}$ can be scaled so that it spans this interval.

Figure 6.5 shows the Beta function with $\sigma = 0.1$ at different locations across the interval $\{0 \leq t \leq 1\}$. It is similar in appearance to the Gamma function although not as peaked. The variance can be held constant while $\alpha$ and $\beta \geq 0$. For small $\sigma$, the points at which $\alpha$ and $\beta = 0$ may be shown to occur when $\eta \equiv \sigma$ and $\eta \equiv 1 - \sigma$ respectively.
6.6 TEST RESULTS OF ASYMMETRIC KERNEL REGRESSION

Both the Gamma and Beta asymmetric kernel regression methods were tested on a noisy sine wave. Simulated data consisted of a unit amplitude sine wave of frequency 0.5 (arbitrary units). The nominal standard deviation of the kernel regression estimators was set at \( \sigma = 0.05 \). With this standard deviation the frequency bandwidth of the kernel regression is greater than that of the sine wave allowing for a low bias error at interior points. All the results are shown normalized with respect to this variance.

Figure 6.6 and 6.7 shows the Gaussian kernel regression and the Gamma kernel regression, as a function of the distance, \( t \), from the boundary which is at \( t=0 \). In this comparison, no noise has been added to the signal. Consequently the error shown is due to the bias error alone. The right hand side (RHS) of the sine wave, which is far away from the boundary and free from boundary error, provides a useful reference. Ideally, if it were free of boundary error, the left hand side (LHS) would be the same as the RHS. As shown in Figure 6.7, the Gaussian regression mirrors the RHS until \( \sigma < 2.5\sigma \) where it begins to deviate from the RHS, indicating the boundary error begins at this point. At closer distances to the boundary it increases rapidly. This is to be expected because the Gaussian function will begin to be truncated by the boundary for \( t < 3\sigma \). The Gaussian bias error passes through the axes at about \( t \approx 1.5\sigma \), which is due to the positive boundary error canceling the negative interior bias error. The Gamma regression allows a much closer approach to the boundary; it mirrors the RHS until \( t = \sigma \). In the interior away from the influence of the boundary, for \( t > 3\sigma \), the error of the Gamma regression is almost the same as the Gaussian regression. This indicates that the third order skew term of the Gamma bias error (Equation (6.32)) does not significantly degrade the performance of the Gamma regression.
Figure 6.6 Gamma and Gaussian kernel regression of sine wave.

Figure 6.7 Gamma and Gaussian kernel regression bias error of sine wave.
Figure 6.8 Comparison of Gamma and Beta kernel regression of sine wave.

Figure 6.9 Gamma and Beta kernel regression bias error of sine wave.
In the interior there is no advantage in using the Gamma function.

For distances $t < \sigma$ the variance of the Gamma regression cannot be kept constant. In this region we fixed $\alpha = 0$, an exponential function, and reduced the variance according to

$$\sigma = \eta = t$$  \hspace{1cm} (6.40)

where $t$ is the distance from the boundary. The reduction of variance for $t < \sigma$ causes the discontinuity in the Gamma curve at $t = \sigma$ shown in Figure 6.7. Although this method removes the boundary error, the noise performance in this region will be degraded as the boundary is approached due to the decreasing variance.

In order to investigate the noise performance, simulated uniform density random noise with amplitude in the range 0 to 0.25 was added to the sine wave signal (Figure 6.10). To provide a quantitative measure it was compared with a boundary error corrected Gaussian regression using the simple method described in Hall and Wehrley [1991]. For this type of boundary error correction, the variance of the Gaussian function is varied according to the distance from the wall in such a way that

$$\sigma = t/3$$  \hspace{1cm} (6.41)

The output noise shown in Figure 6.11 was the ensemble average from 200 records, each record containing 100 data points. A single record is shown in Figure 6.10. The output noise is the square of the difference between the regression operating on the noisy signal and the regression operating on the signal without noise

$$output \ noise(\eta) = (y_{g}(\eta; t_{i}, y_{i} + n_{i}) - y_{g}(\eta; t_{i}, y_{i}))^{2}$$  \hspace{1cm} (6.42)
6.10 Gamma and variable bandwidth Gaussian kernel regression of noisy sine wave.

6.11 Output noise of Gamma and variable bandwidth Gaussian kernel regression of noisy sine wave.
6.12 Gamma, Beta and variable bandwidth Gaussian kernel regression of noisy sine wave.

6.13 Output noise of Gamma, Beta and variable bandwidth Gaussian kernel regression of noisy sine wave.
As shown in Figure 6.11, the output noise of the Gamma regression remains flat as the boundary is approached until $t \equiv \sigma$. Beyond this point it increases inversely as the distance from the boundary decreases. The increase in this region is due to the gradual reduction of the variance, which from Equation (6.40) is proportional to the distance from the boundary. This result is in agreement with that predicted theoretically (Equation (6.33)). The boundary corrected Gaussian regression behaves in a similar way except that the noise begins to increase at a greater distance from the boundary at $t \equiv 3\sigma$, in accordance with Equation (6.41). This leads to a much higher output noise close to the boundary than the Gamma regression.

The Beta kernel regression was investigated by adding another boundary on the right at $t = 1.0$ ($t/\sigma = 20$). Figure 6.8, 6.9, 6.12 and 6.13 shows the results. Its performance was indistinguishable from the Gamma density where, for the Gamma density, the change of variables given by Equation (6.35) was used to correct the RHS boundary. At very large $\sigma$ with respect to the interval $\{0 \leq t \leq 1.0\}$, of the order $\sigma \approx 0.2$, neither performed very well due to the poor frequency response.

### 6.7 CONCLUSION

In this chapter a statistical model based on kernel regression was developed to explain the errors occurring in the Hermite/Laguerre filter. Good agreement was obtained between the model and test results on a sine wave.

Using the same kernel regression statistical model, the error of the Hermite/Laguerre filter close to the boundary of the data was also investigated. As a first step towards correcting the boundary error in the Hermite/Laguerre filter, a method based on an asymmetrical kernel function was proposed and studied. The principal was
demonstrated with a Gamma kernel function. The main problem with correcting the boundary error is to avoid degrading the noise performance. Theoretical arguments supported by numerical simulation indicate that the Gamma kernel is capable of correcting the boundary error, without sacrificing the noise performance, at distances from the boundary as close as $\eta = \sigma$. A Beta kernel function was also investigated. Numerical simulations indicate that its performance is similar to the Gamma kernel regression.

In future work it is hoped that the asymmetrical kernel function can be incorporated into the Hermite/Laguerre filter. One possibility is a hybrid filter with the asymmetrical kernel regression replacing the Hermite/Laguerre filter in the vicinity of the boundary. However time does not permit this work to be done for this thesis. Instead in the next chapter the HNN correlation will be applied to radar signals, where tests will be conducted against the FNN correlation and the Hermite-Rodriguez correlation. The Hermite-Rodriguez correlation was briefly mentioned in Chapter 4 as an alternative correlation algorithm for Hermite functions but it has not yet been compared with the algorithm proposed in this thesis.
CHAPTER 7

APPLICATION OF HERMITE CORRELATION TO RADAR

7.1 INTRODUCTION

The aim of this chapter is to present further results, in addition to those given in Chapters 4 and 5, which compare the new Hermite neural network correlation of this thesis with existing methods. These comparisons are based on the processing of classical radar signals [Cook and Bernfeld, 1967], [Skolnik, 1980] for which correlation plays an important role. Included within this chapter is a brief summary of the development of classical radar systems.

The chapter begins in Section 7.2 with a simple pulsed radar system where the performance of the Hermite correlator is compared with a Fourier correlator and a conventional Sigmoidal neural network. Following this application, radar signals of increasing complexity are investigated. In Section 7.3 Doppler radar signals are considered where a comparison with a Hermite-Rodriguez correlator is made. In Section 7.4 we compare the performance of the Hermite correlator with a Fourier correlator on Frequency Modulated (FM) radar signals. The Hermite-Rodriguez neural networks are defined in the same way as the Hermite neural network and Fourier neural network of Chapter 3 except that a Hermite-Rodriguez function replaces the Hermite or Fourier
function in the network.

Following the same procedure as previous chapters, all the tests are based on simulated data. In particular, the comparisons make use of a pseudo-random sequence for the simulation of random noise.

7.2 MATCHED FILTER DETECTION OF RADAR SIGNALS

Figure 7.1 shows a schematic diagram of a typical, simple radar system. The transmitter consists of a microwave oscillator, modulator and antenna. The oscillator frequency is called the carrier frequency. The oscillator is pulsed by the modulator to produce a short duration signal, which is radiated from the antenna. The pulsed signal is then transmitted as an electromagnetic wave by an antenna. If a target is present, an echo of the transmitted signal is received through the antenna. The carrier frequency is removed by a local oscillator operating at the same frequency as the microwave oscillator in the transmitter to produce the demodulated pulse signal. In some systems the same antenna is used for both the transmission and reception of the radar signal with additional electronics to ensure that simultaneous transmission and reception cannot occur.

A basic requirement in the design of radar receivers is the optimization of the receiver for the detection of the target echo in the presence of noise. For additive, white Gaussian noise the optimum receiver for detecting the target echo is the matched filter receiver. Matched filter is the term used to describe a filter whose transfer function, except for a delay factor, is the complex conjugate of the Fourier transform of the input signal. North [1963] first derived the matched filter in a classified report (RCA Laboratories Report PTR-6C, June 1943) by maximizing the signal-to-noise ratio of the filter of a pulsed radar system using the calculus of variations.
Figure 7.1 Simplified schematic diagram of typical radar system

Figure 7.2 Simplified schematic diagram of radar system with the addition of a frequency modulator.
The actual name “matched filter” was coined by Van Vleck and Middleton [1946] who obtained the same result as North by using the Schwartz inequality.

At almost the same time as the matched filter was developed, the correlator detector was being developed. Lee et al [1950] first applied correlation analysis for the detection of periodic signals in noise. By applying probability theory to radar detection, Woodward and Davies [1950], [1952] showed that in the case of a receiver limited by additive white Gaussian noise, the correlator receiver is optimum in the sense that it extracts all the available information about the target’s presence, range, or velocity. The connection between correlation and the matched filter was made by Woodward [1951] when he demonstrated, that the matched filter and the correlator detector are equivalent.

In view of the importance of the matched filter in the work of this chapter, a mathematical argument that proves it is optimum will now be given using the Schwartz inequality approach of Van Vleck and Middleton [1946]. Let $x(t)$ be the noisy returned echo which is composed of the clean signal, $s(t)$, and the noise $n(t)$:

$$x(t) = s(t) + n(t) \quad (7.1)$$

Let $y(t)$ be the output from the filtering of the noisy signal, $x(t)$, by a filter, $k(t)$. Then the output of the filter is

$$y(t) = \int_{-\infty}^{\infty} x(\tau)k(t - \tau)d\tau \quad (7.2)$$

In this case the filter function, $k(t)$, is to be determined so that it maximizes the signal to noise ratio. The noise at the filter output is
\[ m(t) = \int_{-\infty}^{+\infty} n(\tau) k(t - \tau) d\tau \] (7.3)

The signal to noise ratio at the output is then

\[
SNR = \frac{(y(t) - m(t))^2}{(m(t))^2} = \frac{\left( \int_{-\infty}^{+\infty} s(\tau) k(t - \tau) d\tau \right)^2}{\left( \int_{-\infty}^{+\infty} n(\tau) k(t - \tau) d\tau \right)^2} \]

It is assumed that the noise is white noise; therefore the Fourier transform of the noise is

\[
\int_{-\infty}^{+\infty} n(t) e^{-j\omega t} dt = \lambda \]

(7.4)

where \( \lambda \) is a constant. The signal to noise ratio expressed in terms of the Fourier transform is

\[
SNR = \frac{\left( \int_{-\infty}^{+\infty} S(\omega) K(\omega) e^{j\omega t} d\omega \right)^2}{\lambda^2 \left( \int_{-\infty}^{+\infty} K(\omega) e^{j\omega t} d\omega \right)^2} \]

(7.5)

Applying the Schwartz inequality to the \( SNR \) gives

\[
SNR = \frac{\left( \int_{-\infty}^{+\infty} S(\omega) K(\omega) e^{j\omega t} d\omega \right)^2}{\lambda^2 \left( \int_{-\infty}^{+\infty} K(\omega) e^{j\omega t} d\omega \right)^2} \leq \frac{\left( \int_{-\infty}^{+\infty} S(\omega) e^{j\omega t} d\omega \right)^2}{\lambda^2 \left( \int_{-\infty}^{+\infty} K(\omega) e^{j\omega t} d\omega \right)^2} \]

(7.6)

The equality in Equation (7.6) occurs when \( K(\omega) = S(\omega) \) and therefore it follows that the \( SNR \) must be a maximum when the filter function, \( k(t) \), is equal to the clean signal,
s(t). This completes the proof.

In this section the transmitted pulse will be assumed to be a Gaussian pulse that is

\[ s(t) = \exp\left(-\frac{t^2}{2\sigma^2}\right) \tag{7.7} \]

where the carrier wave has been omitted for convenience. The optimum filter is then

\[ k(t) = \exp\left(-\frac{t^2}{2\sigma^2}\right) \tag{7.8} \]

The Hermite neural network can be configured as the matched filter by choosing \( k(t) = h_n(t) \). The noisy returned signal, \( x(t) \), represents the input training function to the network and is approximated by the Hermite network as \( x_n(t) \):

\[ x(t) \cong x_n(t) = \sum_{n=0}^{N} a_n h_n(t) \tag{7.9} \]

The correlation of the Hermite neural network with the filter is a Laguerre network, \( y_n(t) \). The Laguerre network approximates the correlation of the returned signal with the filter \( k(t) \). From Equation (4.15) of Chapter 4

\[ y(t) \equiv y_n(t) = \begin{cases} \sum_{n=0}^{N} a_n l_n^m(t^2/2) & \text{for } t \geq 0 \\ \sum_{n=0}^{N} a_n (-1)^n l_n^m(t^2/2) & \text{for } t < 0 \end{cases} \tag{5.7} \]

Figure 7.3 shows the effectiveness of the matched filter in a simulation using a Hermite neural network. Figure 7.3 (a) is the returned pulse buried in noise, Figure 7.3 (b) is the Hermite output fitted to the noisy returned signal and Figure 7.3 (c) is the Laguerre network output. Note that the Hermite output itself also provides some filtering. The Hermite correlator consisted of 50 Hermite functions and 50 Laguerre functions.
Figure 7.3 (a) Noisy returned radar signal.

Figure 7.3 (b) Hermite output of interpolation of noisy radar signal

Figure 7.3 (c) Laguerre output (matched filter) of noisy radar signal.
With a network of 50 functions, the width of the Hermite interpolation was 10.0 seconds. This width defines the maximum phase difference between the echo and transmitted pulse that can be detected by the correlator as $\pm 10$ seconds.

In order to assess the performance of the Hermite correlator it was compared in simulation to a standard Fourier correlator (Section 4.5). The Fourier correlator consisted of 50 cosine and sine functions with a fundamental period equal to 20 seconds. With a period of 20 seconds the maximum phase difference between the echo and transmitted signal that can be detected is $\pm 10$ seconds, the same as the Hermite correlator. Figure 7.4 shows the output signal to noise ratio of the Hermite correlator compared to the Fourier correlator as a function of the input signal to noise ratio. The phase difference between the echo and transmitted pulse was $\tau = 0$. The Hermite correlator achieved the same \textit{SNR} to that of the Fourier correlator and there is no advantage in this regard.

The \textit{SNR} test results of Figure 7.4 were based on the assumption of no phase difference occurring between the echo and transmitted signal. When a phase difference occurs, the Fourier correlator is at a disadvantage compared to the Hermite correlator. The Fourier correlator suffers from a circular correlation error (Section 4.6) which causes a phantom pulse to occur in the output at large phase differences as shown in Figures 7.6 (c). The phantom pulse begins to occur at about $\tau = 7$. This phantom pulse is within the range of the correlator and could result in an ambiguity in the target range. The circular correlation error is a result of the periodic nature of the Fourier series and can be rectified by zero padding the signal at the expense of reduced resolution. The Hermite correlator does not suffer from this effect as shown in Figures 7.6 (a), 7.6 (b) and 7.6 (c).
Figure 7.4 Signal to noise ratio of demodulated signal by HNN and FNN.

Figure 7.5 Signal to noise ratio of demodulated signal by HNN and conventional sigmoidal neural network (SNN).
Figure 7.6 (a) Fourier correlator, $\tau = 5.0$.

Figure 7.6 (b) Hermite correlator, $\tau = 5.0$.

Figure 7.6 (c) Fourier correlator, $\tau = 7.0$.

Figure 7.6 (b) Hermite correlator, $\tau = 7.0$.

Figure 7.6 (c) Fourier correlator, $\tau = 9.0$.

Figure 7.6 (c) Hermite correlator, $\tau = 9.0$. 
Although incapable of performing as a matched filter, the sigmoid neural network is still capable of filtering the radar signal. This is achieved by forcing the network to respond only to the pulsed part of the signal by deliberately choosing a network with only a few elements. For the radar signal, a single layer network of three sigmoid functions performed satisfactorily. Initial weights of the network were randomly chosen, followed by gradient descent to interpolate the Doppler signal. Training was slow with computational time at least ten times greater than the Hermite correlator, though the final signal to noise ratio was slightly greater than that achieved with the Hermite correlator (Figure 7.5) Faster training algorithms are available for the Sigmoid neural network but recent research indicates that these may degrade the signal to noise ratio [Caruana et al, 2000].

7.3 DOPPLER RADAR SIGNALS

Doppler radar systems are used to measure the velocity of moving targets [Skolnik, pp. 101-151]. When the target is moving, the mixer in the receiver will not remove the carrier wave. The output of the mixer will contain an oscillation at the Doppler frequency, which is proportional to the target’s velocity and at a much lower frequency than the microwave oscillator. The simple radar system described in Section 7.2 can determine a target’s velocity as well as range by adding instrumentation to measure this frequency difference.

In the previous section it was shown how to detect the target range using a Hermite neural network by fitting the weights of the Hermite network, \( \{a_n\} \), to the radar signal and taking the correlation with the fundamental Hermite function.
Figure 7.7 (a) HNN approximation of noisy Doppler radar signal.

Figure 7.7 (b) HNN demodulation of noisy Doppler radar signal.

Figure 7.7 (c) HNN Fourier transform of noisy Doppler radar signal.
The Hermite correlator can also determine the target velocity using these same weights by replacing the Hermite functions in the Hermite correlator interpolation of the Doppler signal by their Fourier transform. This gives the frequency spectrum of the Doppler signal, from which the velocity may be determined. From Equation (3.32), the frequency spectrum is

$$F\{x_n(t)\} = \sqrt{2\pi} \sum_{n=0}^{N} (-j)^n a_n h_n(\omega) \quad (7.9)$$

The following mathematical model of a Doppler signal was investigated

$$x(t) = (1 + \cos(2\pi f t)) e^{-t^2/2\sigma^2} \quad (7.10)$$

where the frequency, $f = 1.25 \text{Hz}$, and for convenience the Gaussian width was again chosen to be $\sigma = 1.0$. Simulated random noise, with a uniform probability density, of varying strength was added. A Hermite correlator of 70 elements has a frequency response bandwidth of $1.9 \text{Hz}$, which is sufficient to accurately interpolate the Doppler signal. Figure 7.7 (a), 7.7 (b), and 7.7 (c) show the signal, matched filter output and frequency spectrum respectively from the Hermite correlator interpolation containing 70 elements. The root mean square error (rmse) of the Hermite correlator interpolation of the Doppler signal versus the number of Hermite functions in the network is shown in Figure 7.8.

Although the Hermite-Rodriguez functions were mentioned in Section 3.3, they were not described in detail or compared with the Hermite functions. Hermite-Rodriguez functions are Gaussian windowed Hermite functions defined as

$$h_{R_n}(t) = \frac{h_n(t)e^{-t^2/2}}{\pi^{1/4}} \quad (7.11)$$
where $h_n(t)$ is an orthonormal Hermite function. Like the Hermite series, simple expressions also occur for the correlation and Fourier transform of the Hermite-Rodriguez functions. Therefore they are also suitable for processing the Doppler radar signal. The relevant mathematical equations given below are taken from Lo Conte et al [1995]. The correlation of two Hermite-Rodriguez functions is given by

$$h_r(t) * h_r(t) = \sqrt{\frac{(n+m)!}{2^{n+m} n! m!}} hr_{n+m}(t/\sqrt{2})$$ (7.12)

While the Fourier transform of the Hermite-Rodriguez function is an associated Laguerre function

$$F\{hr_n(t)\} = (-j)^n t_0^n (\omega^2/2)$$ (7.13)

Note that the scale is reduced and the order of the Hermite-Rodriguez function increased by the correlation operation.

Application requires the duration and bandwidth of the Hermite-Rodriguez neural network (HRNN) to be matched to the Doppler signal. Unlike the Hermite neural network, which increases in duration with the order $N$ of the function, the HRNN is independent of $N$. Instead it is limited by the duration of the Gaussian window

$$D = 3/\sqrt{2}$$ (7.14)

However the bandwidth in the frequency domain increases with the order $N$ of the function according to

$$B = \sqrt{2N + 1}$$ (7.15)

which is the same as for the Hermite function (Equation (3.34)).
Figure 7.8 Root mean square error (rmse) versus number of Hermite functions.

Figure 7.9 Root mean square error versus number of Hermite-Rodriguez functions.
Figure 7.10 Frequency response of HNN of size $n=70$

Figure 7.11 Frequency response of HRNN of size $n=70$ and $n=1600$. 
To compare the Hermite-Rodriguez series with the Hermite series, requires the Hermite-Rodriguez functions to be scaled by introducing the variable

$$ t \rightarrow t/\alpha \quad (7.16) $$

Scaling the HRNN changes the duration and bandwidth of the network to, respectively,

$$ D = 3\alpha/\sqrt{2} \quad (7.17) $$

and

$$ B = \frac{\sqrt{2N + 1}}{\alpha} \quad (7.18) $$

Using scaled Hermite-Rodriguez functions, the correlation with the Gaussian function is

$$ hr_n(t/\alpha) * hr_n(t/\beta) = \left(\frac{\alpha}{\gamma}\right)^n hr_n(t/\gamma) \quad (7.19) $$

where $\alpha$ and $\beta$ are the scaling factors and $\gamma^2 = \alpha^2 + \beta^2$.

For the purpose of comparison, the scaling parameters were chosen so that the Hermite-Rodriguez series matches as closely as possible the Hermite series. With $\beta = \sqrt{2}$, the order $\{n = 0\}$, Hermite-Rodriguez function is matched exactly to the order $\{n = 0\}$, Hermite function, that is $hr_0(t/2) = h_0(t)$. Similarly, with $\alpha = 10\sqrt{2}/3$, the Hermite-Rodriguez series has a width of 10 seconds. Figure 7.9 shows the root mean square error of the HRNN interpolation as a function of the number of elements of the series. More than 1500 Hermite-Rodriguez functions are needed before the error drops to a sufficiently small value. The reason for this is because the frequency response of the scaled HRNN is much smaller than a Hermite neural network of the same duration.
Figure 7.12 Signal to noise ratio of demodulated signal by HNN and HRNN.
Figure 7.10 shows the frequency response of the Hermite neural network on a unit amplitude cosine. Figure 7.11 shows the response of the HRNN with 70 and 1600 Hermite-Rodriguez functions. The poor frequency response is unavoidable because a sufficiently large $\alpha$ must be chosen to ensure the series has sufficient duration to interpolate the Doppler signal.

Figure 7.12 shows the output signal to noise ratio of the Hermite correlator, $N = 70$, and the Hermite-Rodriguez correlator, $N = 1600$, versus the input signal to noise ratio. Nearly identical signal to noise ratios are achieved, which is to be expected since an identical Gaussian correlation function was used, but the Hermite correlator is considerably more efficient since it achieves the same result with much fewer terms in the series. The Hermite-Rodriguez functions are more specialized than the Hermite functions. In view of their Gaussian window, the HRNN may prove more suitable for correctly analyzing the frequency spectrum of signals subject to glitches.

### 7.4 FREQUENCY MODULATED RADAR

For targets at long ranges the simple radar system described in Section 7.2 and 7.3 is not suitable. This is because of a range/accuracy tradeoff. A wide pulse is required to ensure that the transmitted signal has sufficient energy to produce a returned signal of sufficient strength that it can be detected; but wide pulses result in poor range resolution. To overcome this problem, frequency modulated radar was developed (first patented by Dicke [1953]). Frequency modulated radar has a wide pulse without degrading range resolution. By linearly sweeping the carrier frequency across the width of the pulse, the wide pulse (Figure 7.14) correlates with itself into a short narrow pulse (Figure 7.16) possessing good range resolution. This process is referred to as pulse compression [Skolink, 1985, page 420].
To see how pulse compression occurs, consider a square radar pulse of linear swept frequency defined as

\[
s(t) = \begin{cases} 
\cos(2\pi f t) & -L \leq t \leq L \\
0 & t > L \\
0 & t < -L 
\end{cases}
\]  
(7.20)

where the frequency, \( f \) is

\[ f = \gamma(t + L) / 2L \]  
(7.21)

The transmitted signal may be approximated as a summation of \( N \) rectangular subsections of equal width and different frequency:

\[
s(t) = s_0(t) + s_1(t) + s_2(t) + \cdots
\]  
(7.22)

where each element is a rectangular windowed cosine function of slightly different frequency to the preceding function and is defined mathematically as

\[
s_n(t) = \begin{cases} 
\cos(2\pi f_n t) & \frac{(n-1)L}{N} \leq t \leq \frac{nL}{N} \\
0 & t > \frac{nL}{N} \\
0 & t < \frac{(n-1)L}{N} 
\end{cases}
\]  
(7.23)

The frequencies of each rectangular element are

\[ f_n = n\Delta\omega \]  
(7.24)

where

\[ \Delta\omega = \gamma / N \]  
(7.25)
The echo from a target will be of the same form but shifted in phase according to the range of the target and also corrupted with noise. At the detector output the correlated signal is

\[ s(t) \ast s(t) = \{s_0(t) + s_1(t) + s_2(t) + \cdots\} \ast \{s_0(t) + s_1(t) + s_2(t) + \cdots\} = \sum_{m,n} \{s_n(t) \ast s_m(t)\} \]

(7.26)

Cross terms in the summation do not contribute because rectangular subsections of different frequency do not correlate with each other. Then (see section 4.7) it follows that

\[ s_n(t) \ast s_m(t) = \begin{cases} [L - t/2L] \cos(2nf \pi t) & m = n \\ 0 & m \neq n \end{cases} \]

(7.27)

Therefore at the detector output the correlated signal is a summation of cosine functions

\[ s(t) \ast s(t) = \begin{cases} [L - t/2L] \left[ \cos(2nf \pi t) + \cos(2nf \pi t) + \cdots + \cos(2nf \pi t) \right] & -L \leq t \leq L \\ 0 & t > L \\ 0 & t < -L \end{cases} \]

(7.28)

This summation is approximately equal to the integration of a cosine function with respect to frequency, which is a delta function:

\[ \cos(2nf \pi t) + \cos(2nf \pi t) + \cdots + \cos(2nf \pi t) = \delta(t) \]

(7.29)

Consequently
A more accurate mathematical solution than the simple model described above is given in the report of Klauder et al [1960]. In the same report, design guidelines for frequency modulated radar are given. Klauder et al [1960] used the name “CHIRP” to describe this type of signal and it is now in widespread use.

Although an improvement over a non-frequency modulated signal, the CHIRP signal is still not ideal. If the target is moving, several measurements are required to resolve the velocity and range of the target. In an effort to develop an improved CHIRP signal, Klauder [1960] suggested the Hermite function (Figure 7.13) as a possible modulation signal. Compared to the conventional CHIRP signal, it has a circular ambiguity function enabling the velocity and range of a target to be resolved from a single measurement. The disadvantage is that the Hermite signal has larger side lobes than the CHIRP signal reducing its performance against multiple targets (compare Figure 7.15 with Figure 7.16). Since the Hermite correlation algorithm is based on the Hermite series it is ideally suited to processing Hermite signals. To ascertain the performance of the Hermite correlation algorithm, it was compared with a Fourier correlator on simulated, noisy Hermite radar signals.

The Hermite function used for the transmitted/received signal was of order \( \{n = 18\} \). This signal has a duration \( t = 7 \) seconds (Figure 7.13) and the full width of the first peak of the correlated output is \( t = 0.8 \) seconds (Figure 7.15). Simulated random noise, with a uniform probability density, of varying strength was added to the received Hermite signal. The Hermite correlator consisted of two networks.
Figure 7.13 Hermite signal of order 18.

Figure 7.14 CHIRP signal matched to Hermite function of order 18.
Figure 7.15 Correlated Hermite signal of order 18.

Figure 7.16 Correlated CHIRP signal with the width of the first peak matched to the Hermite correlated signal of Figure 7.15
Figure 7.17 Root mean square error of FNN approximation of Hermite signal versus number of Fourier functions.

Figure 7.18 Root mean square error of FNN approximation of Hermite signal versus sampling rate.
The first network contained a single element, the clean, Hermite function just prior to transmission. The second network, consisting of 20 Hermite functions, was used to model the noisy returned signal, shifted in phase in proportion to the target’s range. Range information is given by the shift in the peak of the correlated output signal of the two networks.

The Fourier correlator consisted of two Fourier networks. This operated in a similar way to the Hermite correlator except that the first network, fitted to the clean Hermite signal, consisted of more than one element. In order to apply the Fourier network, several parameters were first required; an appropriate sample rate must be chosen and the number of elements of the network must be large enough to enable a good fit with the Hermite signal. Figure 7.17 shows the root mean square error of the Fourier network fitted to the clean Hermite signal. A satisfactory root mean square error was achieved with 80 Fourier functions. Figure 7.18 shows the sample rate, which indicates that a rate of at least four samples per second is required; for convenience a sample rate of 10 per second was used.

Signal to noise ratios were evaluated on the correlated output of both networks. The results are shown in Figure 7.19. The SNR of the Hermite correlator is superior to the Fourier correlator due to the more efficient modeling of the signal, which allows a lower number of functions to perform the same correlation.

It is also of interest to compare the SNR properties of the Hermite radar signal with a conventional CHIRP signal (Figure 7.14). For the purpose of comparison, the CHIRP signal power, frequency sweep rate and duration were carefully chosen to match the Hermite signal and also the width of the first peak of the correlated Hermite signal as closely as possible (Figure 7.15 and Figure 7.16).
Figure 7.19 SNR of Hermite signal with HNN and FNN correlators.

Figure 7.20 SNR of Hermite signal and CHIRP signal both using FNN correlators.
Figure 7.21 SNR of Hermite signal with HNN correlator and CHIRP signal with FNN correlator.
The CHIRP signal was processed using the Fourier correlator. When the Hermite signal was also processed with the Fourier correlator the SNR were similar (Figure 7.20). However the SNR of the Hermite signal, processed with the Hermite correlator, was superior (Figure 7.21).

7.5 CONCLUSION

Several applications of the Hermite neural network correlation to radar were investigated using simulated random noise. The main results are now summarized.

For the matched filtering of radar signals (Section 7.1), the Hermite correlator achieved the same $SNR$ as a Fourier correlator of the same size. The Hermite correlator was shown to be free of phantom radar pulses, which plague the Fourier correlator at large phase differences between the radar echo and transmitted signal. Compared to a conventional neural network composed of sigmoidal functions, the Hermite correlator was of slightly lower $SNR$ but the training time of the Hermite correlator was considerably faster. Although specifically for radar, the comparison is also applicable to ultrasonic transducers and sonar.

The isomorphic Fourier transform of the Hermite neural network was shown to be useful in the processing of Doppler radar signals (Section 7.2). In this application the Hermite correlator is capable of determining range and velocity simultaneously. A neural network of Hermite-Rodriguez functions is also capable of this type of simultaneous processing. In a comparison between each type of neural network, the $SNR$ of both neural networks was the same but the Hermite correlator was considerably more efficient in terms of the number of network functions required.
The comparison with the Fourier correlator, which began with the simple radar system of Section 7.2, was continued in Section 7.4 with frequency modulated radar. When applied to Hermite CHIRP radar signals, the Hermite correlator produces a higher SNR and a more compact neural network than is possible with a Fourier correlator.
CHAPTER 8

ASSOCIATED TRAINED SIGMOID NEURAL NETWORK

8.1 INTRODUCTION

In the comparison of Chapter 5 it was shown that the sigmoid neural network produced a more compact network than was possible with the Hermite/Laguerre neural network. This suggests a hybrid neural network combining the Gaussian correlation properties of the Hermite/Laguerre neural network with the efficient interpolation properties of the sigmoid neural network. However before this can be achieved, further development of the sigmoid neural network is required to overcome some serious training problems. These problems [Mulgrew, 1996], which have become a hallmark of the sigmoid neural network, include the extreme length of training times, unreliable convergence to a solution, the indeterminate nature of the training times and the lack of a methodology for architecture selection.

In addition to the poor training, the filtering characteristics of the sigmoid neural network are unknown and unpredictable. In this chapter a new method for training the sigmoid neural network is proposed using an associated RBNN. This allows existing RBNN training to be applied, which overcomes the poor training of the sigmoid neural
CHAPTER 8 ASSOCIATED TRAINED SIGMOID NEURAL NETWORK

network. In addition, the associated RBNN allows a theoretical model to be developed for the filtering characteristics for the sigmoid neural network. Section 8.2 describes the new method, the principle of which is based on training to the rate of change of the non-linear function rather than to the function itself. Section 8.3 describes the RBNN training and the theoretical filtering characteristics. Section 8.4 describes the noise performance of the sigmoid network trained with the new algorithm. Section 8.5 tests the algorithm on a noisy sine wave with a comparison against a conventionally trained sigmoid neural network.

8.2 PRINCIPLE OF ASSOCIATED TRAINED SIGMOID NEURAL NETWORK

The conventional training algorithm for sigmoid neural networks is to randomly initialize the network weights and then seek a solution by gradient descent [Haykin, 1999]. A significant improvement is possible by applying a faster training algorithm than gradient descent; a popular one being the Levenberg-Marquardt algorithm [Ampazis and Perantonis, 2002], [Kollias and Anastassiou, 1989]. Other types of algorithms include variations of the conjugate gradient and quasi-Newton algorithms, which are reviewed by Battiti [1992]. These faster training algorithms generally offer between 10 to 50 times faster training speed over gradient descent [Hagan and Menhaj, 1994]. Training speed may be further increased by careful choice of the initial weights of the network. Some of the more recent schemes developed for initialization of the network weights, have been summarized by Yam and Chow [2001].

Neither the faster algorithms, nor the improved initialization schemes guarantee convergence to a solution. Furthermore, the faster algorithms increase the complexity of the training. In order to improve the training of neural networks, Broomhead and Lowe [1988] introduced radial basis activation functions for neural networks; the term Radial
Basis Neural Networks (RNNN) has since been coined for these types of networks. Some of the earliest work on RBNNs was by Moody and Darken [1989] who demonstrated two orders of magnitude faster training times using the RBNN compared with the sigmoid neural network. The improved training is due to the localized response of the radial basis function as opposed to the global response of the sigmoid function. However the penalty for the faster training is that the RBNN generally requires a larger number of activation functions.

RBNNs and sigmoid neural networks are now the two most commonly encountered neural networks, generally complementing each other. In this section the main principle of the new training algorithm for the sigmoid neural network is described. It combines the benefits of both types of network: fast training of the RBNN with the low number of activation functions of the sigmoid neural network.

Let $x(t)$ be a differentiable function with derivative $v(t) = dx(t)/dt$. The approximation of $x(t)$ by a sigmoid network, with a single hidden layer of $N$ activation functions, is described mathematically by

$$x_s(t) = \sum_{n=0}^{N-1} a_n s(t - \tau_n, \lambda_n) + c \quad (8.1)$$

where $s(t, \lambda)$ is the sigmoid function defined as

$$s(t, \lambda) = \frac{1}{1 + e^{-\lambda t}} \quad (8.2)$$

and $a_n$, $\tau_n$, $\lambda_n$ are the network weights; $c$ is a constant. We define the associated sigmoid network as the new network, which is formed by differentiation of the sigmoid network of Equation (8.1), as:
\( v_s(t) = \frac{dx_s(t)}{dt} = \sum_{n=0}^{N-1} a_n \frac{ds(t - \tau_n, \lambda_n)}{dt} \) \hspace{1cm} (8.3)

where the derivative of the sigmoid function is

\[
\frac{ds(t, \lambda)}{dt} = \frac{\lambda e^{-\lambda t}}{(1 + e^{-\lambda t})^2}
\] \hspace{1cm} (8.4)

The derivative of the sigmoid function is a radial symmetric bell shaped function, as shown in Figure 8.3. We therefore define the associated sigmoid kernel function to be the derivative of the sigmoid function, that is:

\[
k_a(t) = \frac{ds(t - \tau_n, \lambda_n)}{dt} = \frac{\lambda_n e^{-\lambda_n t}}{(1 + e^{-\lambda_n t})^2}
\] \hspace{1cm} (8.5)

It then follows that the associated sigmoid network can be re-written as

\[
v_s(t) = \sum_{n=0}^{N-1} a_n k_a(t - \tau_n)
\] \hspace{1cm} (8.6)

which is an RBNN estimate of the derivative, \( v(t) \). It is relatively easy to train to \( v(t) \).

Once the associated sigmoid network is trained, the weights of the corresponding sigmoid network approximating \( x(t) \), can be obtained directly from those of the associated sigmoid network, without any further training required.

Also shown in Figure 8.3 is the Gaussian function for comparison. The associated sigmoid kernel function is generally very similar to the Gaussian function, though it is slightly more peaked. The similarity with the Gaussian function suggests the possibility of replacing the associated sigmoid kernel function with this type of function.
Figure 8.1 Standard trained sigmoid neural network

Figure 8.2 Associated trained sigmoid neural network
Figure 8.3 Associated sigmoid kernel function.

Figure 8.4 Fourier transform of associated sigmoid kernel function.
However the integration of the Gaussian function does not have an elementary form; its
evaluation requires numerical tables. In this case it becomes difficult to obtain the
network \( x_s(t) \) from \( v_s(t) \).

The main feature of the new algorithm is that it trains the sigmoid network via the
derivative of a function rather than to the function itself. The derivative is of a lower
order and therefore less complex than the original function. In view of the lower
complexity, this offers a possible explanation as to why generally fewer activation
functions are required by a sigmoid network than with a RBNN.

The new training method will be referred to as associated training in order to distinguish
it from a sigmoid neural network trained by the conventional gradient descent
algorithm. Figure 8.1 shows the training schematic of the conventional trained neural
network and Figure 8.2 shows the associated trained neural network.

**8.3 DIFFERENTIATION OF THE TRAINING DATA**

It is evident that training requires the derivative of the non-linear function, \( v(t) \). If this
derivative is not available, then it can be estimated from the training data using the
relation

\[
v(t_i) \approx \frac{x(t_{i+1}) - x(t_{i-1})}{2\Delta t}
\]

where \( \Delta t \) is the sampling interval defined as

\[
\Delta t = t_{i+1} - t_i
\]

This is a three-point estimate [Kreyzig, 1983, page 792]; more accurate estimates of the
derivative are available in the mathematical literature [Hildebrandt, 1956].
8.4 TRAINING OF THE ASSOCIATED SNN

The network weights $a_n$ and $\tau_n$ are conveniently referred to as the amplitude and center respectively of the kernel function. $\lambda_n$ is related to the width of the kernel function.

A standard approach, to training the weights of a RBNN is by the method of Moody and Darken [1989]. In this method, first the centers, $\tau_n$, of the kernel functions are arranged by a clustering algorithm, then the width of the kernel function is derived from the distance between the kernel function centers and finally the amplitude weights are obtained by the least-means square algorithm.


The particular RBNN training method chosen in this thesis is described in the following sections. It is applicable to training data, $\{t_i, x_i : i = 0, 1, 2, ..., I\}$, sampled at regular intervals across the range of the function $\{-T/2 \leq t \leq T/2\}$. The training method uses a direct analytical solution for the weights of the associated sigmoid RBNN followed by optimization of the weights using the Least Means Square (LMS) algorithm. These weights are then passed directly to the corresponding sigmoid network, which requires no further training other than the constant $c$.

The analytical solution for the weights of the RBNN is obtained by a Fourier method. The Fourier method allows the noise characteristics of the RBNN to be estimated. This
approach using the Fourier method has not been applied to the RBNN in any of the author’s references.

### 8.5 CONSTANT OF INTEGRATION

The constant of integration, $c$, in the sigmoid network was determined after the associated sigmoid network had been trained by taking the average difference between $x_s(t)$ and $x(t)$ over the complete training data set, that is

$$c = \frac{1}{I} \sum_{i=0}^{I-1} (x_s(t_i) - x(t_i))$$  \hspace{1cm} (8.9)

### 8.6 NETWORK CENTRES AND WIDTHS

The centers of the associated sigmoid kernel functions were distributed evenly across the interval with two of the kernel functions centered at the endpoints of the data and truncated by the edges of the data. The two truncated kernels at the edges of the data allow the network to converge in this region. The first kernel center is at

$$\tau_0 = -T/2$$  \hspace{1cm} (8.10)

while the remaining centers were determined recursively from

$$\tau_{n+1} = \tau_n + \Delta T$$  \hspace{1cm} (8.11)

where $\Delta T$ is the distance between the centers of the kernel functions which is

$$\Delta T = \frac{T}{N - I}$$  \hspace{1cm} (8.12)

The centers of the kernel functions were not adaptively trained; these remained fixed at the values given by Equations (8.10) and (8.11).
The width of the kernel function may be defined to be its standard deviation. The standard deviation, \( \sigma \), of the associated sigmoid kernel can be evaluated from integration tables to be [Prudnikov, et al., 1986, page 333, Equation 2.3.12.3]

\[
\sigma^2 = \int_{-\infty}^{\infty} k(t) dt = \int_{-\infty}^{\infty} \left( \frac{\lambda t^2 e^{-\lambda t}}{1 + e^{-2\lambda t}} \right) dt = \frac{\pi^2}{3\lambda^2}
\]

(8.13)

It depends only on \( \lambda \), which was chosen to be

\[
\lambda = \frac{2}{\Delta T}
\]

(8.14)

With this value of \( \lambda \), the standard deviation, \( \sigma \), is approximately equal to the distance between the centers of the kernel functions, which is a common practice with RBNN [Moody and Darken, 1989]:

\[
\sigma \equiv \Delta T
\]

(8.15)

Since the centers of the kernel functions are evenly distributed across the interval, the widths of all of the kernel functions are all the same and it follows that \( \lambda_n = \lambda \). These widths were also not adapted.

The reason for not adaptively changing the centers and widths of the kernel functions is that under these conditions the network will attempt to approach that of an ideal filter, providing an analytical solution for the weights and also allowing the noise performance to be estimated. This will be explained in more detail in Section 8.8.

**8.7 CONVOLUTION BASED NOTATION AND FOURIER TRANSFORM**

In order to simplify the derivation of the analytical solution, the associated sigmoid network will be written in an equivalent shortened form as the correlation
\[ v_x(t) = \sum_{n=0}^{N} a_n k(t - \tau_n) = \sum_{n=0}^{N} a_n k(t - n\Delta T) = a(t) \ast k(t) \]  

(8.16)

where \( a(t) \ast k(t) \) is the correlation integration

\[ a(t) \ast k(t) = \int_{-\infty}^{\infty} a(\tau) k(t - \tau) d\tau \]  

(8.17)

and \( a(t) \) is an arbitrary function satisfying the conditions that

\[ a(n\Delta T) = a_n \]  

(8.18)

and

\[ a(t) = \sum_{n=-\infty}^{\infty} a(n\Delta T) \delta(t - n\Delta T) \]  

(8.19)

\( \delta(t) \) is the Dirac delta function which is

\[ \delta(t) = \begin{cases} 1 & \text{for } t = 0 \\ 0 & \text{for } t \neq 0 \end{cases} \]  

(8.20)

\( a(t) \) will be referred to as the weight function. Let the Fourier transform of \( a(t) \) be \( A(\omega) \), then from Equation (8.19)

\[ A(\omega) = \sum_{n=-\infty}^{\infty} a(n\Delta T) e^{-jn\omega \Delta t} \]  

(8.21)

which is periodic in the frequency domain with period \( 2\pi/\Delta T \).

By transformation into the frequency domain, the correlation becomes a multiplication and the equivalent frequency domain form of Equation (8.16) is
CHAPTER 8 ASSOCIATED TRAINED SIGMOID NEURAL NETWORK

\[ V_s(\omega) = A(\omega)K(\omega) \]  \hspace{1cm} (8.22)

where \( V_s(\omega) \) and \( K(\omega) \) are the Fourier transforms of \( v_s(t) \) and \( k(t) \) respectively.

The Fourier transform of the associated sigmoid kernel may be shown to be [Prudnikov et al, page 450, Equation 2.5.34.7]

\[ K(\omega) = \int_{-\infty}^{\infty} \frac{\lambda e^{-bt}}{(1 + e^{-at})^2} \cos(2\pi\omega t) dt = \frac{\pi \omega/\lambda}{\sinh(\pi \omega/\lambda)} \]  \hspace{1cm} (8.23)

Figure 8.4 displays the Fourier transform of the associated sigmoid kernel function and the Gaussian function (Equation 5.15), in both cases the standard deviation, \( \sigma \), was set to be \( \sigma = 1.0 \) (see Equation (8.13) for the standard deviation of the associated sigmoid function).

From Figure 8.4 it can be seen that, although similar to the Gaussian function, the amplitude of the associated sigmoid function drops off a little less sharply than the Gaussian function.

8.8 ANALYTICAL SOLUTION FOR AMPLITUDE WEIGHTS

Training of the associated sigmoid RBNN is with respect to the derivative. A standard criteria for measuring the fit of the neural network to, \( v(t) \), is the mean square error (mse), which is defined as

\[ mse = \int_{-\infty}^{\infty} (v(t) - v_s(t))^2 dt \]  \hspace{1cm} (8.24)

A feature of the training method is that the centers and widths of the kernel functions have not been adaptively trained and remain fixed. Under these conditions an analytical
solution exists for the associated sigmoid RBNN, which will be developed in this section.

The \( mse \) of the associated SNN interpolation of \( y(t) \) is

\[
\text{mse} = \int_{-\infty}^{\infty} (y(t) - v_s(t))^2 \, dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} (V(\omega) - V_s(\omega))^2 \, d\omega \quad (8.25)
\]

The second equality in Equation (8.23) follows from Parseval’s theorem [Papoulis, 1962, page 27]. Substituting Equation (8.18) for \( \omega_s V \) and integrating in segments of band-width \( 2\pi/\Delta T \) gives

\[
\text{mse} = \int_{-\infty}^{\infty} (V(\omega) - V_s(\omega))^2 \, d\omega
\]

\[
= \cdots + \int_{-\pi/\Delta T}^{\pi/\Delta T} (V(\omega) - A(\omega)K(\omega))^2 \, d\omega + \int_{-\pi/\Delta T}^{\pi/\Delta T} (V(\omega) - A(\omega)K(\omega))^2 \, d\omega +
\]

\[
+ \int_{\pi/\Delta T}^{2\pi/\Delta T} (V(\omega) - A(\omega)K(\omega))^2 \, d\omega + \cdots \quad (8.26)
\]

which is equivalent to the summation

\[
\text{mse} = \int_{-\pi/\Delta T}^{\pi/\Delta T} \sum_{n=-\infty}^{\infty} (V(\omega + 2n\pi/\Delta T) - A(\omega + 2n\pi/\Delta T)K(\omega + 2n\pi/\Delta T))^2 \, d\omega \quad (8.27)
\]

Due to the Fourier transform, \( A(\omega) \), being periodic (see Equation (8.21)), it follows that

\[ A(\omega + 2\pi/\Delta T) = A(\omega) \]

Then Equation (8.27) reduces to

\[
\text{mse} = \int_{-\pi/\Delta T}^{\pi/\Delta T} \sum_{n=-\infty}^{\infty} (V(\omega + 2n\pi/\Delta T) - A(\omega)K(\omega + 2n\pi/\Delta T))^2 \, d\omega \quad (8.28)
\]
By inspection this will be a minimum when

\[ A(\omega) \sum_{n=-\infty}^{\infty} K(\omega + 2n\pi/\Delta T) = U(\omega) \sum_{n=-\infty}^{\infty} V(\omega + 2n\pi/\Delta T) \quad (8.29) \]

where

\[ U(\omega) = \begin{cases} 1 & \text{for } |\omega| \leq \pi/\Delta T \\ 0 & \text{for } |\omega| > \pi/\Delta T \end{cases} \quad (8.30) \]

is the ideal filter. If the bandwidth of the radial function, \( K(\omega) \), and the signal, \( V(\omega) \), is less than \( \pi/\Delta T \), then Equation (8.29) becomes

\[ A(\omega)K(\omega) \equiv U(\omega)V(\omega) \quad (8.31) \]

The left-hand-side is the Fourier transform of the associated SNN. In other words, the optimum solution for the associated sigmoid RBNN is such that it approaches

\[ V_{o}(\omega) \rightarrow U(\omega)V(\omega) \quad (8.32) \]

which is equivalent to the correlation of the function \( v(t) \) with an ideal filter of bandwidth, \( B \), given by

\[ B = \frac{\pi}{\Delta T} \quad (8.33) \]

The network weight function to achieve the ideal filter solution is given by

\[ A(\omega) = \frac{V(\omega)U(\omega)}{K(\omega)} \quad (8.34) \]

The equivalent weight function in the time domain is the inverse Fourier transform of Equation (8.34), which is
The individual network weights can then be deduced from Equation (8.18) to be the Fourier coefficients

\[
a_n = \frac{1}{2\pi} \int_{-\infty}^{\infty} V(\omega)U(\omega) e^{j\omega_n} d\omega
\]  

(8.36)

This solution is very similar to the truncated, de-convolution solution for blurred images described in image processing [Petrou and P. Bosdogianni, 1999].

The analytical solution of Equation (8.36) can be evaluated numerically by using the discrete Fourier and inverse Fourier transforms. However the network weights will require tuning because the associated sigmoid kernel is not strictly band-limited. The LMS algorithm for this tuning is

\[
a_n(t_{i+1}) = a_n(t_i) - \alpha(v_n(t_i) - v(t_i))k(t_i - \tau_n)
\]  

(8.37)

where \(\alpha\) is the learning rate constant.

In some applications it may be inconvenient to evaluate the Fourier and inverse Fourier transforms. As an alternative to the analytical solution, an approximate solution that does not require the Fourier transform may be used, followed by tuning with the LMS training algorithm. The approximate solution can be derived by assuming that the non-linear function has a bandwidth much less than the bandwidth of the associated sigmoid kernel function. With this assumption, \(K(\omega) \approx 1.0\) which may then be substituted into Equation (8.36). Then by evaluating the integral, the approximate solution is

\[
a_n = v_n(\tau_n)\Delta T \equiv v(\tau_n)\Delta T
\]  

(8.38)
With this value for $a_n$, and without the LMS tuning, the network is equivalent in mathematical form to a simple filter applied to $v(t)$ of the type described in electrical engineering, with the associated sigmoid kernel function performing the filtering. In statistics it is equivalent in form to the Priestley and Chao [1972] kernel regression estimate for $v(t)$. However the associated sigmoid network has far fewer terms in the series than either the filter or the kernel regression; the later have the same number of terms as the training data.

Note that the mathematical solution derived in this section does not apply to a conventionally trained sigmoid network. This is because the conventionally trained network will not approach the ideal filter solution of Equation (8.32) due to the training of both the kernel centers and widths of the conventional trained network whereas for Equation (8.32) to apply they must be held fixed.

### 8.9 NOISE FILTERING PROPERTIES OF ASSOCIATED TRAINING

The noise at the output of the sigmoid network, trained via the associated network, can be estimated by a statistical analysis similar to that applied in kernel regression [Wand and Jones, 1995] or by the classical electrical engineering analysis of the convolution filter [Blinchikoff and Zverev, 1976]. We apply the classical electrical engineering approach.

The training of the associated sigmoid network requires the estimation of the derivative (Equation (8.7)). This estimation will be poor when the data is corrupted with noise because of the amplification caused by the small term, $\Delta t$, in the denominator of the derivative [Papoulis, 1965, page 398].
However this does not affect the corresponding sigmoid network, which is obtained from the associated sigmoid network by integration. The integration cancels the increase in noise caused by differentiation.

The fact that the frequency response of the associated sigmoid network converges towards the ideal filter will be applied to derive the noise performance of the corresponding sigmoid network. Differentiation in the time domain is equivalent to multiplication by $j\omega$ in the frequency domain. Consequently the frequency response of the sigmoid network can be obtained from the frequency response of the associated sigmoid network by replacing the derivatives, $V_s(\omega)$ and $X(\omega)$, in Equation (8.32) with $j\omega X_s(\omega)$ and $j\omega X(\omega)$, respectively so that Equation (8.32) becomes

$$j\omega X_s(\omega) \equiv j\omega X(\omega)U(\omega) \quad (8.38)$$

or

$$X_s(\omega) \equiv X(\omega)U(\omega) \quad (8.39)$$

where $X_s(\omega)$ and $X(\omega)$ are the Fourier transforms of $x_s(t)$ and $x(t)$ respectively. Assume that the noise is white additive noise denoted by $q(t)$ with Fourier transform $Q(\omega)$ then

$$X_s(\omega) \equiv (X(\omega) + Q(\omega))U(\omega) \quad (8.40)$$

The Fourier transform of white noise is a constant and the noise power, $P_{out}$, at the output of the sigmoid network is the integral over frequency

$$P_{out} = Q^2 \int_0^\infty U^2(\omega) d\omega = Q^2 \omega_{out} \quad (8.41)$$
\( \omega_{out} \) is the equivalent noise output bandwidth, which is equal to that of the ideal filter, that is

\[
\omega_{out} = B = \frac{\pi}{\Delta T}
\]  
(8.42)

Note that this result applies only to the sigmoid network trained via its associated network and does not apply to a conventionally trained sigmoid network.

### 8.10 Test Results for the Associated Training Algorithm

The associated training method was tested in comparison with the conventional training method. The conventional training method for the sigmoid neural network consisted of randomly initializing the network weights in the range \( \pm 1 \) and then applying gradient descent to locate the optimum weights.

The root mean square error (\( rmse \)) was based on the difference between \( x(t) \) and \( x_s(t) \). That is the root mean square error (\( rmse \)) is

\[
rmse = \sqrt{\frac{1}{T} \int_{-T/2}^{T/2} (x(t) - x_s(t))^2 dt}
\]  
(8.43)

The signal for these tests was a sine wave described mathematically by

\[
x(t) = \sin(2\pi t/T)
\]  
(8.44)

of period, \( T \), equal to 20 arbitrary units. This was sampled at the regular interval of 0.1 steps across the interval \( \{ -10 \leq t \leq 10 \} \) resulting in a sample size of 200. At this sample rate the error due to the numerical computation of the derivative is low (Figure 8.5).
Figure 8.5 Effect of sampling rate on rmse of a sigmoid neural network with associated training and 20 activation functions.

Figure 8.6 Frequency response of sigmoid neural network with associated training and 20 activation functions.
The difference in frequency response of the sigmoid neural network trained with the associated method from the ideal filter was evaluated experimentally. This was done by inputting a cosine wave of varying frequency into the network and recording the amplitude of the output signal. Figure 8.6 shows the frequency response for a network with 20 activation functions. The response is not as sharp as the ideal filter and is slightly peaked but the bandwidth is approximately the same. The reason for the difference from an ideal filter response is due the finite summation of the neural network and the infinite duration of the associated sigmoid kernel function. It is possible to obtain the ideal filter by replacing the associated sigmoid kernel function with the sinc function. Unfortunately the integration of the sinc function does not have an elementary form and it is therefore difficult to obtain $x_s(t)$ from $v_s(t)$.

According to the sampling theorem, a minimum of 2.5 samples per wavelength is required for the interpolation of a sine wave. Therefore the minimum number of activation functions possible with the network is 3. Figure 8.7, 8.8 and 8.9 shows the operation of the sigmoid network with associated training, 3 activation functions in the hidden layer, and using the initial solution for the weights taken from the approximate solution of Equation (31). Figure 8.7 is the initial network output with the amplitude weights determined only by the initial value of Equation (8.38). Figure 8.8 and 8.9 is the network output with 50 and 100 training cycles respectively of the LMS algorithm operating with a learning rate set at the constant value of $\alpha = 0.5$. The $rmse$ after 100 training cycles was 0.01 (1% of the amplitude of the sine wave). In the absence of noise, the $rmse$ represents the bias error of the network.
Figure 8.7 Associated training without LMS ($\alpha = 0.5$) and 3 activation functions.

Figure 8.8 Associated training with 50 training cycles of LMS ($\alpha = 0.5$) and 3 activation functions.

Figure 8.9 Associated training with 100 training cycles of LMS ($\alpha = 0.5$) and 3 activation functions.
Figure 8.10 Error of amplitude weights with respect to the LMS optimum solution (6 activation functions).

<table>
<thead>
<tr>
<th>n</th>
<th>$\tau_n$</th>
<th>LMS</th>
<th>Approx.</th>
<th>Analytical</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-10.00</td>
<td>-2.00</td>
<td>-1.26</td>
<td>-1.90</td>
</tr>
<tr>
<td>1</td>
<td>-6.00</td>
<td>-0.57</td>
<td>-0.39</td>
<td>-0.59</td>
</tr>
<tr>
<td>2</td>
<td>-2.00</td>
<td>1.54</td>
<td>1.02</td>
<td>1.54</td>
</tr>
<tr>
<td>3</td>
<td>2.00</td>
<td>1.54</td>
<td>1.02</td>
<td>1.54</td>
</tr>
<tr>
<td>4</td>
<td>6.00</td>
<td>-0.57</td>
<td>-0.39</td>
<td>-0.59</td>
</tr>
<tr>
<td>5</td>
<td>10.00</td>
<td>-2.00</td>
<td>-1.26</td>
<td>-1.90</td>
</tr>
</tbody>
</table>

Table 8.1 Weights for network with 6 activation functions (See figure 8.10 above for error of the analytical and approximate solution)
Figure 8.11 Error of amplitude weights with respect to the LMS solution (9 activation functions).

Table 8.2 Weights for network with 9 activation functions. (See figure 8.11 above for error of the analytical and approximate solution).
Figure 8.12 Error of amplitude weights with respect to the LMS solution (18 activation functions).

<table>
<thead>
<tr>
<th>n</th>
<th>$\tau_n$</th>
<th>LMS</th>
<th>Approx.</th>
<th>Analytic</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-10.00</td>
<td>-0.52</td>
<td>-0.37</td>
<td>-0.39</td>
</tr>
<tr>
<td>1</td>
<td>-8.82</td>
<td>-0.32</td>
<td>-0.34</td>
<td>-0.36</td>
</tr>
<tr>
<td>2</td>
<td>-7.65</td>
<td>-0.30</td>
<td>-0.27</td>
<td>-0.28</td>
</tr>
<tr>
<td>3</td>
<td>-6.47</td>
<td>-0.17</td>
<td>-0.16</td>
<td>-0.17</td>
</tr>
<tr>
<td>4</td>
<td>-5.29</td>
<td>-0.04</td>
<td>-0.02</td>
<td>-0.02</td>
</tr>
<tr>
<td>5</td>
<td>-4.12</td>
<td>0.11</td>
<td>0.10</td>
<td>0.11</td>
</tr>
<tr>
<td>6</td>
<td>-2.94</td>
<td>0.23</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>7</td>
<td>-1.76</td>
<td>0.33</td>
<td>0.32</td>
<td>0.33</td>
</tr>
<tr>
<td>8</td>
<td>-0.59</td>
<td>0.38</td>
<td>0.36</td>
<td>0.38</td>
</tr>
<tr>
<td>9</td>
<td>0.59</td>
<td>0.38</td>
<td>0.36</td>
<td>0.38</td>
</tr>
<tr>
<td>10</td>
<td>1.76</td>
<td>0.33</td>
<td>0.32</td>
<td>0.33</td>
</tr>
<tr>
<td>11</td>
<td>2.94</td>
<td>0.23</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>12</td>
<td>4.12</td>
<td>0.11</td>
<td>0.10</td>
<td>0.11</td>
</tr>
<tr>
<td>13</td>
<td>5.29</td>
<td>-0.04</td>
<td>-0.02</td>
<td>-0.02</td>
</tr>
<tr>
<td>14</td>
<td>6.47</td>
<td>-0.16</td>
<td>-0.16</td>
<td>-0.17</td>
</tr>
<tr>
<td>15</td>
<td>7.65</td>
<td>-0.30</td>
<td>-0.27</td>
<td>-0.28</td>
</tr>
<tr>
<td>16</td>
<td>8.82</td>
<td>-0.32</td>
<td>-0.34</td>
<td>-0.36</td>
</tr>
<tr>
<td>17</td>
<td>10.00</td>
<td>-0.53</td>
<td>-0.37</td>
<td>-0.39</td>
</tr>
</tbody>
</table>

Table 8.3 Weights for network with 18 activation functions (see figure 8.12 above for error)
Figure 8.13 displays the training curves for the sigmoid network with associated training, with 8, 9, 12, 15 and 18 activation functions respectively, and with the initial solution for the weights taken from Equation (8.38). There is a progressive reduction in the error and in the number of training cycles required as the number of activation functions of the network is increased. The \( \text{rmse} \) after training varied from between 0.2% to 0.7% of the amplitude of the sine wave depending on the size of the network.

The analytical solution for the network weights of the sine function is to a first approximation

\[
a_n = \frac{2\pi T \cos(2\pi n \Delta T / T)}{K(2\pi / T) \Delta T}
\]  

(8.45)

Table 8.1 shows the LMS solution, the approximate solution and the analytical solution for the network amplitude weights with 8 activation functions. Tables 8.2 and 8.3 are for networks with 9 and 18 activation functions respectively. The error between the approximate and analytical solution and the LMS solution is shown in Figures 8.10 to 8.12. For a network with 18 activation functions, both the approximate and analytical solutions are close to the LMS solution, except at the endpoints of the data (Figure 8.12) where there is a significant error. A similar error at the endpoints of the data also occurs in kernel regression [see Chapter 6] where it is caused by the boundary of the data truncating the kernel function. As the network size decreases from 18 to 8 activation functions the analytical solution provides an increasing improvement compared to the approximate solution. Also, the error at the endpoints of the data becomes less noticeable (Figures 8.10 and 8.11).

Figure 14 displays the training curves for a sigmoid network with associated training,
with 8, 9, 12, 15 and 18 activation functions respectively, and with the initial solution for the weights taken from the approximate analytical solution of Equation (8.45). It may be observed from the training curves, that the analytical solution does not significantly improve the training speed.

A set of ten different initial sets of random weights was applied to the conventionally trained sigmoid network. For a network consisting of 3 activation functions, the conventionally trained sigmoid network failed to converge properly in eight out of the ten initial sets of random weights. However with a network of 8 or more activation functions at least eight out of the ten initial weight sets converged to a solution. Figure 8.15 displays the training curves for the conventional sigmoid network with 8, 9, 12, 15 and 18 activation functions. These training curves are the ensemble average of the individual training curves produced from those sets of random weights that converged.

The erratic behavior present in the training curves has also been reported elsewhere [Hagan and Menhaj, 1994]. The size of the network does not influence the number of training cycles required. The conventionally trained network required 50 000 training cycles compared to the 500 training cycles required by the associated training method. The \textit{rmse} after training of the conventionally trained network was between 0.12\% and 0.17\% of the amplitude. The lower error achieved by the conventionally trained network compared with the network trained with the associated method is for several reasons. One reason is that the width and centers of the conventionally trained network are adapted providing a better fit to the data, particularly noticeable for networks with a low number of activation functions. The penalty for the better fit is a poorer SNR, the bias/variance dilemma reported by Geman \textit{et al} [1992].
Figure 8.13 Associated training curves ($\alpha = 0.1$) with approximate solution of Equation (8.31).

Figure 8.14 Associated training curves ($\alpha = 0.1$) with approximate analytical solution of equation (8.45).
Figure 8.15 Conventional neural network training curves ($\alpha = 1$).

Figure 8.16 Comparison of theoretical estimation of SNR with actual SNR for an associated trained sigmoid neural network with 20 activation functions.
Another reason for the lower error of the conventionally trained network is that the associated trained network suffers from an additional error caused by the estimation of the derivative.

In order to investigate the noise performance of the associated training method, random noise was added to the sine wave. The noise was simulated from a pseudo-random sequence of numbers. The SNR was calculated on the sine wave at the input and output of the network. The input signal to noise ratio, $SNR_{in}$, is defined as the ratio of the signal power without noise to the noise power at the input. The output signal to noise ratio, $SNR_{out}$, is the ratio of the output signal without noise to the noise power at the output.

The noise power at the output of a sigmoid network with associated training was shown in Section 8.9 (Equation (8.41)) to be proportional to the ideal filter bandwidth. At the input, the noise equivalent bandwidth can be taken to be

$$\omega_{in} = \frac{\pi}{At}$$  \hspace{1cm} (8.43)

It then follows that the ratio of output to input SNR is

$$\frac{SNR_{out}}{SNR_{in}} = \frac{\omega_{in}}{\omega_{out}}$$  \hspace{1cm} (8.44)

Figure 8.16 shows the SNR of the sigmoid network with 20 activation functions and the theoretical SNR of Equation (8.44). Good agreement is obtained with Equation (8.44) validating the theoretical analysis of Section 8.9 and demonstrating that the differentiation does not increase the noise of the sigmoid network trained via its associated network.
Figure 8.17 SNR of associated trained sigmoid neural network versus conventionally trained network (6 activation functions).

Figure 8.18 SNR of associated trained sigmoid neural network (6 activation functions) versus Gaussian kernel regression.
Figure 8.17 shows the SNR of the sigmoid network with associated training and 6 activation functions in comparison with a conventionally trained network with the same number of activation functions. The associated trained network achieves a SNR, which is on average, 40% greater than the conventionally trained network. The reason for the improved SNR of the associated trained network, is that it has fewer unknown weights that need to be trained; that is the centers and width of the sigmoid functions are not stochastic variables.

Figure 8.18 shows the SNR in comparison with Gaussian kernel regression similar to that described by Specht [1991]. The bandwidth of the Gaussian kernel regression was chosen to be $\sigma = 0.4$. With this bandwidth, the rmse matches the sigmoid neural network with associated training on the signal without noise. Both networks achieve similar SNR on the noisy signal. The sigmoid network has the advantage in size consisting of only 6 activation functions whereas the Gaussian regression has the same number of terms as the training data, that is 200 terms.

8.11 CONCLUSION

An associated training algorithm has been developed for the sigmoid neural network for the approximation of a non-linear function from a uniformly sampled set of training data. The algorithm trains to the rate of change of the unknown non-linear function. The algorithm has been proven in tests on a sine wave in the presence of noise. Although more rigorous test signals are available, the sine wave provides a simple, well-known signal with which to demonstrate the training characteristics of the new method against the conventionally trained sigmoid network.

An obvious disadvantage of the associated training is that it requires the derivative of
the non-linear function to be calculated (if it is not already available). This means that
the method is only suitable for those applications in which sufficient data is available to
allow a sufficiently accurate calculation of the derivative. Although the estimation of
the derivative will be poor on noisy data, it was demonstrated that this does not degrade
the noise performance of the sigmoid network trained with the associated training
method.

Test results on a sine wave indicated that the network was capable of training with a
factor of at least 100 times fewer iterations of the gradient descent algorithm compared
to a conventionally trained sigmoid network. A further attractive feature of the training
is that as the size of the network increases, the number of training cycles required
decreases.

The training is reliable in the sense that the weights of each of the network activation
functions converge to unique and different solutions. When using a conventional
sigmoid neural network, the number of activation functions is often greater than
required due to the fact that the not all of the weights converge or because the weights
of some of the activation functions converge to the same values.

Although the associated training method produced a larger bias error, it has a distinct
advantage in regards to noise performance. The associated trained sigmoid neural
network obtained a higher SNR than the conventionally trained network. This SNR was
equivalent to that of a Gaussian kernel regression filter. In addition to the higher SNR,
the filtering characteristics of the associated trained network agreed well with the
theoretically predicted SNR.

For convenience in application, the test function for the associated training algorithm
was a sine wave. Further test on more complex non-linear functions should be conducted in the future.
CHAPTER 9

CONCLUSION AND RECOMMENDATIONS

The aim of this thesis was to investigate the correlation of Hermite functions in the form of a Hermite neural network. The terminology “Hermite neural network” describes the Hermite series of orthonormal functions, which have been generalized to include numerical algorithms that are often associated with artificial neural networks.

The relative performance of the Hermite neural network correlator was compared to a Fourier neural network correlator. The main result is that the correlation of a Hermite neural network is a summation of $N \times N$ associated Laguerre functions whereas a Fourier neural network correlation is a summation of $N$ Fourier functions. In this regard the Fourier neural network will be more efficient for the general correlation of functions. An exception occurs for the correlation of the Hermite neural network with a Gaussian function, or with the Hermite radar signal. For these signals the correlation is also a summation of $N$ terms. In these applications the Hermite correlator proved to be superior to the Fourier correlator for the following reasons.

- It does not suffer from a circular correlation error, which is a characteristic of the Fourier correlator.

- It allows the Gaussian inverse correlation to be computed without the numerical instability that occurs with a Fourier correlator.
• It achieves a more compact signal interpolation for the CHIRP radar signal correlator than is possible with a Fourier correlator.

The correlation with the Gaussian function was thoroughly investigated because of its potential as a filter and inverse filter. To distinguish the Hermite neural network from other correlation methods for the Gaussian filter, the term Hermite/Laguerre filter was adopted.

A theoretical model for the statistical errors that occur with the Hermite/Laguerre filter was derived and good agreement was obtained in simulations. The $\text{SNR}$ achieved with the Hermite/Laguerre filter closely agreed with a theoretical $\text{SNR}$ based on the bandwidth of the network.

At the edges of the data, a boundary error was found to occur. A method for correcting the boundary error of the Hermite/Laguerre filter was proposed using asymmetric kernel regression. In numerical simulation good results were achieved in comparison to the conventional Gaussian kernel function.

In a comparison with a sigmoid neural network, the $\text{SNR}$ of the Hermite/Laguerre filter was 40% higher. Due to the orthogonal structure of the Hermite/Laguerre filter the training speed was also superior being two orders of magnitude greater than the sigmoid neural network. In other criteria of performance the Hermite neural network was inferior to the sigmoid neural network. Accuracy was inferior to the sigmoid neural network and in addition the sigmoid neural network did not suffer from a boundary error. The sigmoid neural network could also achieve a much more compact network than the Hermite/Laguerre filter.

To take advantage of the compact form of the sigmoid neural network a hybrid neural
network was proposed combining the Gaussian correlation properties of the Hermite/Laguerre neural network with the compact properties of the sigmoid neural network. In order to achieve a practical hybrid neural network, a new fast method of training the sigmoid neural network was developed. The principle of the new training method is that it trains the sigmoid neural network to the rate of change of the unknown non-linear function rather than to the function itself. This allows the sigmoid neural network to be trained with an associated radial basis neural network with the speed that is inherent with this type of network.

In tests on a sine wave the associated training method was 100 times faster than the conventional training. In regards to accuracy, the associated training method produced a larger bias error than the conventional training but obtained a higher SNR. In addition to the higher SNR, the filtering characteristics of the associated trained sigmoid neural network were able to be predicted which is difficult to do with the conventional training. Future tests should be performed on more complex waveforms to support the results obtained with the sine wave.

The chief recommendation of the author is to further develop the Hermite neural network correlator using parallel electronic circuitry to take advantage of the speed offered by the neural network technology. Such a network would prove to be very useful as a filter, inverse filter or as a detector for radar signals.

Apart from the Fourier orthonormal functions, the author is only aware of a correlation formula that has been published for one other series of orthonormal functions, the Laguerre functions [Brinker, 1993]. Therefore it is recommended that the correlation of other types of orthonormal functions be investigated such as the Legendre, Tchebychev, and Bessel functions.
REFERENCES


7) L. Brillouin, 1926, C. R. Juli.


REFERENCES


22) I. G. Currie, 1974, Fundamental mechanics of fluids, McGraw-Hill, USA.


33) P. Hall and T. E. Wehrly, 1991, “A geometrical method for removing edge effects


52) Y. W. Lee, 1960, Statistical theory of communication, John Wiley and Sons, USA.


vol. 28, pp. 925-935.


64) H. Mishina, T. Koyama and T. Asakura, 1975, Velocity measurements of blood flow in the capillary and vein using a laser Doppler microscope, Applied Optics,


70) Neural computing, (author(s) not given), 1993, NeuralWare, Pittsburgh, USA.


REFERENCES


104) G. G. Walter, 1994, Wavelets and other orthogonal systems with applications, CRC Press, USA.


118) J. Yang and A. Reeves, 1995, Bottom-up visual image processing probed with weighted Hermite polynomials, Neural Networks, 8, pp. 669-691.

