Neural Networks for Signal Processing

by

Dipankar Bhattacharya
B. Tech. (Hons.), Indian Institute of Technology, Kharagpur, 1985
M. Eng., Memorial University of Newfoundland, St. John's, 1992

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

We accept this dissertation as conforming
to the required standard

Dr. Andreas Antoniou, Supervisor (Department of Electrical and Computer Engineering)

Dr. Wu-Sheng Lu, Departmental Member (Department of Electrical and Computer Engineering)

Dr. Nikitas Dimopoulos, Departmental Member (Department of Electrical and Computer Engineering)

Dr. Yury Stepanenko, Outside Member (Department of Mechanical Engineering)

Dr. Majid Ahmadi, External Examiner (University of Windsor)

Supervisor: Dr. Andreas Antoniou

ABSTRACT

The application of neural networks in the area of signal processing is examined. Two major areas are identified and suitable neural networks are developed. In the first area, neural networks are used as a tool for the design of digital filters. In the second area, neural networks are used for processing bathymetric data.

The field of artificial neural networks is first introduced with an emphasis on Hopfield networks. The optimizing capabilities of such networks are noted. Based on these networks, a feedback neural network is developed for the design of 1-D finite-duration impulse response (FIR) filters on the basis of given amplitude responses. A suitable cost function is formulated first and an associated network is developed. This work is then extended to the design of two more networks for the design of FIR filters based on given amplitude and phase responses and prescribed specifications. The idea is extended to the design of 2-D FIR filters. Two networks are presented for designing 2-D FIR filters on the basis of a given amplitude response and prescribed specifications. The design of 1-D infinite-duration impulse response (IIR) filters is studied next and two networks are developed. The first one is to design filters with prescribed specifications in the magnitude-squared domain. The other network designs IIR filters for a given frequency response. A network for designing equiripple 1-D FIR filters based on the weighted least-squares technique is presented next. A new updating algorithm is developed for this network.

Two different neural networks are proposed for classifying lidar waveforms into various categories. A single-layer network is developed for classifying lidar waveforms representing milt of varied densities. A fast version of the supervised learning algorithm is presented. A threshold term is also introduced in the recall phase to give the user flexibility to accept or reject any waveform. A two-stage, multi-layer network is presented next which uses waveform characteristics to assign a signature number to the waveform. This network extracts various ocean parameters from the waveforms as well.

The issue of implementing the feedback neural network is addressed next. Basic building blocks for implementing such networks are identified and a network is constructed from circuits existing in the literature. The network is simulated in Cadence using 0.8 μ BICMOS

technology. The results show that these networks have a high potential to be implemented in analog VLSI for real-time signal processing.

Examiners:

Dr. Andreas Antoniou, Supervisor (Department of Electrical and Computer Engineering)

Dr. Wu-Sheng Lu, Departmental Member (Department of Electrical and Computer Engineering)

Dr. Nikitas Dimopoulos, Departmental Member (Department of Electrical and Computer Engineering)

Dr. Yury Stepanenko, Outside Member (Department of Mechanical Engineering)

Dr. Majid Ahmadi, External Examiner (University of Windsor)

Contents

List of Tables

List of Figures

List of Abbreviations

BICMOS	:	Bipolar complementary metal-oxide semiconductor
BJT	:	Bipolar junction transistor
CAD	:	Computer aided design
CMOS	:	Complementary metal-oxide semiconductor
FIR	:	Finite-duration impulse response
GPS	:	Global positioning system
IIR	:	Infinite-duration impulse response
IR	:	Infra red
LIDAR	:	Light detection and ranging
LS	:	Least squares
MOS	:	Metal-oxide semiconductor
1-D	:	One-dimensional
2-D	:	Two-dimensional
VLSI	:	Very large-scale integration
WLS	:	Weighted least squares

List of Symbols

A_a	: Stopband attenuation
A_p	: Passband attenuation
act	: Activation value of a neuron
C_g	: Input capacitance of a g neuron
C_f	: Input capacitance of an f neuron
E	: Energy function of a feedback neural network
E_s	: Sum-square error function
f	: An f neuron or its transfer characteristic
F	: Integral of f
$f(\text{act})$	: Decision-making function of a neuron
$f'(\text{act})$	: Derivative of f with respect to argument
g	: A g neuron or its transfer characteristic
G_i	: Total input conductance of the ith g neuron
$H(e^{j\omega})$	: Frequency response of 1-D digital filter
$H_r(e^{j\omega})$	: Real part of frequency response
$H_i(e^{j\omega})$	: Imaginary part of frequency response
$H(e^{j\omega_1}, e^{j\omega_2})$	: Frequency response of 2-D digital filter
$h(n)$	: Impulse response of 1-D digital filter
$h(n_1, n_2)$	: Impulse response of 2-D digital filter
K	: Transconductance parameter of a MOS transistor
$M(\omega)$	: Amplitude response of a filter
η	: Learning rate

R_f : Input resistance of an f neuron

r_g : Input resistance of a g neuron

T_{ij}, W_{ij} : Conductance or weight between neurons j and i

u_i : Input voltage of the ith g neuron

δ_a : Error in stopband

δ_p : Error in passband

δ_{ok} : Error term for the kth output neuron

λ_f : Gain of an f neuron

λ_g : Gain of a g neuron

ω_a : Stopband edge

ω_p : Passband edge

ω_s : Sampling frequency

θ_i : Threshold parameter of neuron i

$\theta(\omega)$: Phase response of a filter

Acknowledgment

I sincerely acknowledge my supervisor Dr. Andreas Antoniou for all his help, intellectual as well as financial, during my Ph.D. program. I appreciate the guidance he provided during the research and writing of this dissertation. I also acknowledge Micronet, Networks of Centers of Excellence Program, and the Natural Sciences and Engineering Research Council, Canada, for financial support.

I also acknowledge Terra Surveys Ltd., B.C., Canada, for supplying the LARSEN waveforms.

I would like to mention some of the finest people I met in Victoria and in many ways they have become an important part of my student life. Sergio, Marcello, Sri, Inder, Radha, and Sonu, to name a few. I sincerely acknowledge all their help. I also mention Al for his help on Cadence and other stuff and Kevin for undergraduate labs while serving as teaching assistants.

Finally to my wife Soma and daughter Lira for their support and patience. I also deeply acknowledge the support of our families back in India.

For all of you, and many others whose names are not included here but surely deserve my respect and admiration, I humbly offer my deepest and sincere thanks.

To my mother

Chapter 1

Neural Networks and Signal Processing

1.1 Introduction

Artificial neural networks are networks of simple processing elements called neurons. They are connected to each other by weights. The strength of the connections or the weights determine the degree of interaction among the neurons. The knowledge that a neural network is supposed to possess lies in its connectivities. In the so called connectionist model, neurons are represented by amplifiers and resistors model the synapses or the weights. Each neuron multiplies the incoming signals by the corresponding weights and sums them up. If this sum or the activation value is more than a threshold, the neuron changes its output. The network can be trained to adjust its weights in the learning phase. This is a very coarse approximation of what goes on in our brain. Still, the network is able to perform some tasks more easily than a traditional computer because of massive connectivities and parallel operations of all the elements.

The applications of neural networks in the area of signal processing can be broadly classified into two groups:

1) Use of neural networks as a tool for the design of digital filters.

2) Use of neural networks directly for signal processing.

Filtering is one of the most important tasks in signal processing. In this process, a given signal is manipulated by performing certain operations to obtain an output signal whose

frequency spectrum is a modified version of that of the input signal. Filtering is performed by filters, either analog or digital. The theory and design of analog filters have been investigated for decades and are now well established. With the advent of high-speed computers and the availability of dedicated microelectronic circuits and chips, digital filters have evolved and have established themselves as more than a viable tool for signal processing. The procedures for the design of digital filters that have been reported in recent years are almost always software oriented, i.e., the design is carried out by computer programs. However, a properly formulated neural network can be designed to perform the same design tasks with the added advantage that it can be implemented in very-large-scale integration (VLSI).

In certain tasks, e.g., speech and image processing [43], [61], the input data need to be processed before any useful information can be extracted from the signal. Neural networks are a good choice for these tasks. Because of their massive connectivities, they tend to outperform traditional computers for certain cognitive tasks like pattern recognition and classification. The same idea can be used advantageously for bathymetric data. Because of the inherent embedded noise, the signals need to be classified into various groups before a suitable algorithm can be applied. A properly constructed neural network can be trained for this purpose. In the next section, a brief introduction to the field of airborne laser bathymetry is presented.

1.2 Bathymetry

Bathymetry is an area of science that deals with the measurement of depths of water in oceans, seas, and lakes as well as information derived from such measurements. To increase depth measuring capability and to enhance the accuracy, airborne laser bathymeters have been introduced. In airborne laser bathymetry, sea depth is measured using the light detection and ranging (lidar) system. LARSEN 500 is a lidar system introduced in Canada for mapping sea-bed topography in coastal areas.

The laser-beam geometry for the LARSEN 500 system is shown in Figure 1.1. Blue-green and infrared (IR) pulses are projected simultaneously from an aircraft in a quasi-circular fashion [67]. The reflected light is collected by an optical telescope. The received signal comprises a strong surface reflection, a volumetric backscatter from the water column, and

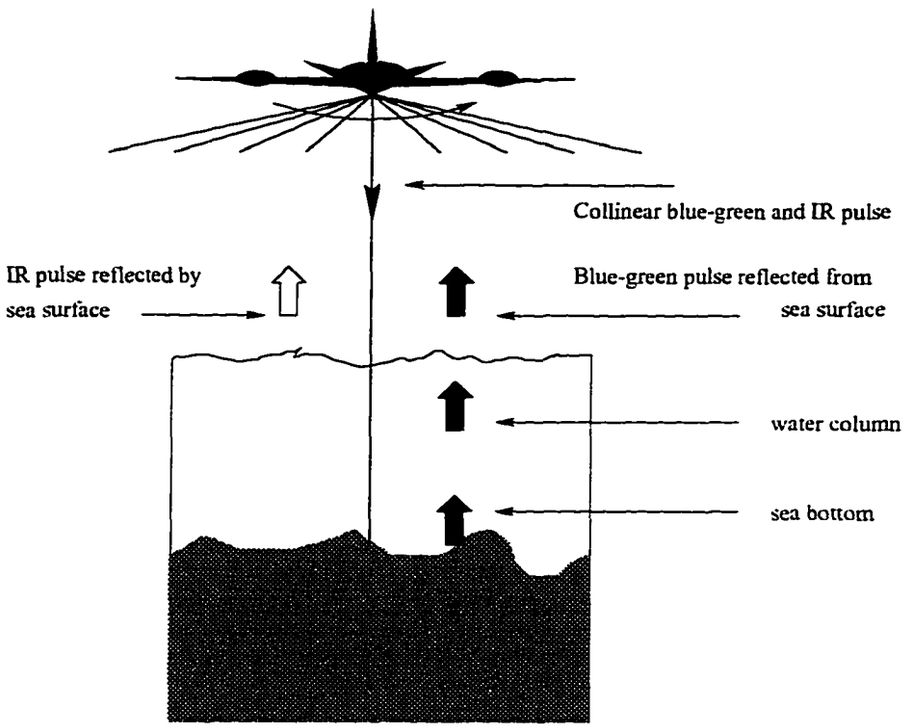


Figure 1.1: Laser-beam geometry for the LARSEN 500 system (adapted from [67]).

a weak reflection from the bottom. The nature of the reflected signal depends on a number of parameters, e.g., sea depth, water quality, sea state, and sea-bed structure, to name a few. A typical lidar waveform with its different parts identified is shown in Figure 1.2.

The presence of any floating or submerged objects such as milt or fish is also reflected in the received waveforms. Thus a properly formulated neural network can be used along with lidar technology for the detection of fish populations in coastal areas. Traditional surveys normally concentrate on offshore deep-water environments. However, for some important commercial species like herring, near-shore waters are the usual habitats of young fish. Because of the problems associated with the use of big ocean-going vessels in coastal areas, lidar technology has proven to be one of the most attractive choices for this purpose. However, a reliable method is required by which small variations in the waveform due to the presence of fish can be detected.

A number of signal processing algorithms [68]-[71] have been proposed to improve sea-depth estimation for different ocean conditions. The existing waveform-processing algorithms are mostly heuristic and do not consider varying ecological diversities. In order to

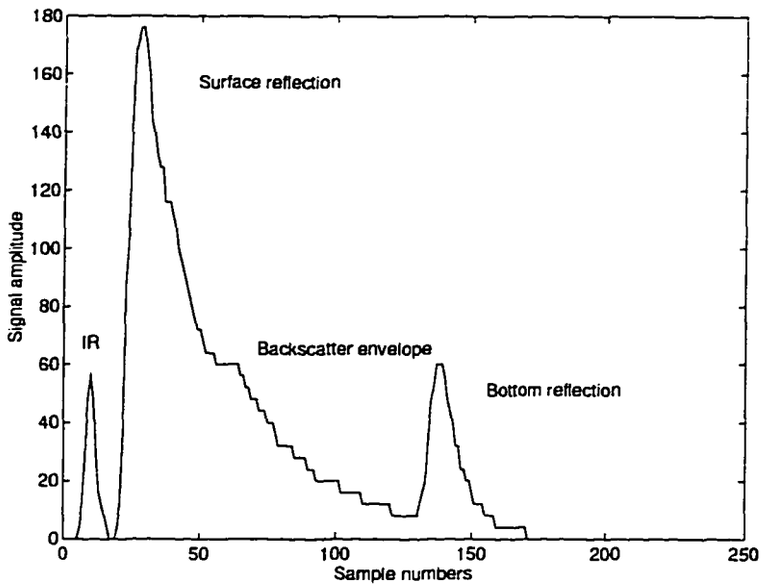


Figure 1.2: A typical lidar waveform.

apply these algorithms successfully to different ocean environments, a new classification scheme needs to be developed which should incorporate all the characteristics of the waveforms. Multi-layered neural networks have been used successfully in a number of real-world applications, e.g., speech recognition [16], military target identification [20], and are naturally suited for the classification of objects or patterns such as lidar waveforms. The network should be able to classify waveforms into different groups as well as extract various types of ocean information, and it should be such that new waveforms from different sites can be added without hindering its performance. In other words, the network should be able to handle data from diverse sea states and ecological conditions. The kind of neural network that should be used for a specific task depends on the task itself. In the next section, a brief introduction to artificial neural networks is presented.

1.3 Neural networks

Artificial neural networks were biologically inspired [59]. They are networks of simple processing units interconnected by weights of variable strengths. They are neural in the sense that the computation is done collectively rather than individually. In general, in a neural network, an amplifier with a nonlinear output transfer characteristic forms the cell

body. wires replace axons and dendrites. and resistors model the synaptic connections or weights among the interacting units. When a neuron is activated, it evaluates all inputs from other neurons and constructs the weighted sum. If the sum, which is often referred to as the activation, goes beyond a predetermined threshold, the neuron changes its output.

In mathematical terms, if O_j represents the jth neural output, then the total activation act_i of the ith neuron is given by

$$\text{act}_i = \sum_j W_{ij} O_j$$

where W_{ij} is the strength of connection from unit u_j to unit u_i. The output of u_i is given by $O_i = f(\text{act}_i, \theta_i)$ where f is a nonlinear decision-making function and θ_i is the threshold voltage.

The strength of connection between two neurons determines the degree of interaction between the two. It can be either excitatory or inhibitory, represented normally by positive or negative weights. If the connection is excitatory, then the activation of the neuron is increased and if it is inhibitory, the activation is reduced.

When a network is activated, all the neurons operate in parallel and respond by adjusting their states. In the synchronous update mode, they simultaneously update their states at each pulse of a central timing clock: while in asynchronous update, each of the neurons, at each instant of time, has a fixed probability of updating its state. Since the neurons update their states independently, in a very small timeframe only one neuron can be thought of updating its state. Whatever the updating procedure, eventually the neurons settle to a stable state representing some global configuration. This is achieved by utilizing the locally available information and the massive parallelism inherent to the system.

Different researchers have proposed networks employing different units in different configurations [1] but most can be encompassed within the stated framework. The major differences are noted below.

- Connectivity: Connectivity varies from single-layered networks (e.g., Hopfield nets) to multi-layered networks with hidden units (e.g., backprop nets). Backprop nets are strictly feedforward and connections are essentially unidirectional. Hopfield nets, on the other hand, have bidirectional connections.

- Neural units: Neural units can be linear if the output of the neuron is an identity function of the activation value [59]. On the other hand, in the linear threshold unit

like the perceptron [72], the output is 1 if the activation is more than the threshold value. However, the most common model is the one utilizing a semilinear activation function which is a monotonically non-decreasing differentiable function.

- States: Output states can be binary as in Hopfield's content addressable memory [34], backprop networks [59], or it can be a continuous value as in Hopfield's neural decision networks [37].

- Activation: The output function or the decision can be either deterministic or probabilistic. The models employing the former are Hopfield nets, backprop nets etc., whereas the Boltzman machine [1] employs a probabilistic response function.

- Representation: The overall representation can be local where the state of individual units may represent something meaningful. By contrast, in the distributed representation, the state of each unit has to be interpreted in conjunction with all other neurons.

It is worthwhile, in this context, to discuss Hopfield nets [35]. In a Hopfield net, every neuron is connected to every other neuron except to itself (i.e. $W_{ii} = 0$). The other restriction is that the weights are symmetrical, that is $W_{ij} = W_{ji}$. For a two-state neuron i, the total input is

$$x_i = \sum_j T_{ij} V_j + I_i$$

where I_i is an external input to neuron i and V_j is the output of neuron j. In the simplest, non-graded formulation, the output of neuron i is $V_i = V_i^1$ if $x_i > U_i$ and V_i^0 otherwise, where U_i is the threshold for neuron i. An energy function such as

$$E = -\frac{1}{2} \sum \sum_{i \neq j} T_{ij} V_i V_j - \sum_i I_i V_i + \sum_i U_i V_i \tag{1.1}$$

may be associated with the network [35]. Then the change in energy, ΔE, due to the change in the output of neuron i is given by

$$\begin{aligned} \Delta E &= -\left[\sum_i T_{ij} V_j + I_i - U_i\right] \Delta V_i \\ &= -[x_i - U_i] \Delta V_i \end{aligned} \tag{1.2}$$

The above quantity is always negative because if $x_i > U_i$, then ΔV_i is positive; otherwise, both of them are negative. Thus any change in V_i lowers the energy function. Since E is

bounded, the system eventually reaches a stable state when the outputs stabilize. A similar expression for the energy function can also be obtained for neurons with graded response [36].

1.3.1 Learning

The information content in a neural network resides in the connection strengths. Learning is the process of adjusting the connection strengths or weights in such a way as to produce a set of desired outputs. Learning can be broadly classified into supervised and unsupervised learning. In supervised learning, inputs are presented along with a set of teaching inputs. Weights are adjusted step by step under the supervision of the teaching inputs so that the network will produce a correct output pattern whenever the trained input pattern is presented. In unsupervised learning, there is no teaching input. However, the network learns by capturing the regularities of the input patterns and responding to any special feature that may be present in the input patterns. A brief review of the back propagation learning scheme (supervised learning) and competitive learning (unsupervised learning) follows next. A detailed discussion on these two learning schemes can be found in [59], [72], and [45].

Backpropagation neural networks are strictly hierarchical feedforward multi-layered networks as shown in Figure 1.3. The first layer is the input layer which receives external inputs and feeds the outputs to the next layer of hidden units. Any layer can receive inputs from the layer just before it and can transmit the outputs to the layer immediately after it. There may be more than one hidden layer and only one output layer. Neurons in hidden and output layers employing semilinear activation rules are useful for capturing higher-order regularities. Besides these units, there may also be bias units which are always on and are connected to hidden and output units.

Backpropagation learning involves two phases of computation. It basically minimizes the sum-squared error over all the output units and all the training patterns using a steepest descent method. Inputs are presented and the network computes the outputs (o_{pk}). These outputs are then compared with the desired or the teaching inputs (t_{pk}) to generate the error signal δ_{pk} where subscript p represents any pattern p and k is any unit. Weights are then adjusted for all the connections feeding the output layer according to

$$\Delta_p W_{kj} = \eta \delta_{ok} y_{pj} \tag{1.3}$$

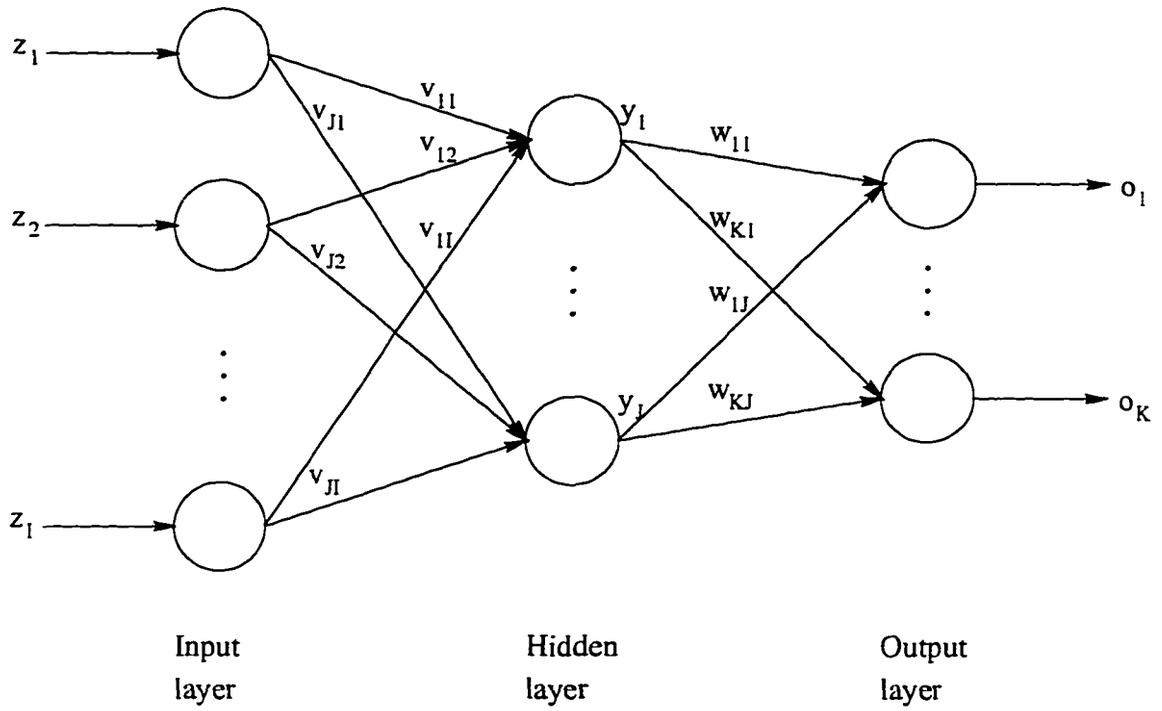


Figure 1.3: Multi-layer feedforward network.

where η is the learning rate and y_{pj} is the input to unit k from unit j for pattern p. It can be shown that for the output units

$$\delta_{ok} = (t_{pk} - o_{pk}) f_k'(\text{act}_k)$$

where f' is the derivative of the activation function with respect to its argument. The δ's are then computed for the penultimate layer as

$$\delta_{yj} = f_j'(\text{act}_j) \sum_k \delta_{ok} W_{kj} \tag{1.4}$$

where j and k are indices of interacting neurons. Thus the error is propagated back one layer. By utilizing the recursive formula of (1.4), the error can be computed for any units in a layer and the weights are adjusted according to (1.3). It is important to note that the patterns are required to be presented repeatedly in order to generate the proper internal representation.

A typical activation function employed by the hidden and the output units is given by

$$o_k = \frac{1}{1 + e^{-\text{act}_k}}$$

This is a sigmoid function which is differentiable as well. The derivative of f with respect to the activation value, f', is given by

$$\frac{\partial o_k}{\partial \text{act}_k} = o_k(1 - o_k)$$

This derivative is maximum for $o_k = 0.5$ and since the change in weight depends on this derivative, the weight change will be maximum for the units with outputs near the mid range.

In the competitive learning method [59], units are also organized in a hierarchical layered fashion. Any units can receive inputs from all the units in the layer immediately below and can feed the output to all the units in the next upper layer through excitatory connections only. All the units in a layer are grouped together into a number of clusters. Units in a cluster inhibit each other so that only one unit in a cluster is active (*winner-take-all* strategy). Each unit has a fixed amount of weight distributed over all the input lines, i.e., $\sum_j W_{ij} = 1$. Learning is achieved by shifting weights from the inactive input lines to the active ones. If a unit wins, then each of the input lines gives up a proportion (k) of its weight which is redistributed among the active lines, that is,

$$\Delta W_{ij} = \begin{cases} 0 & \text{if unit } i \text{ loses} \\ k\frac{L_j}{n} - kW_{ij} & \text{if unit wins} \end{cases}$$

where $L_j = 1$ if the input line from unit j is active and n is the total number of active units. However, if the the input patterns have a few active components, then some of the lines may never be on and the corresponding unit may never win. In order to remove this constraint, the weight can also be changed according to the above equation, even if the unit loses, but at a much lower proportion. That, at least, will enable the unit to be in the competition. Similar performance can also be achieved by changing the threshold in such a way that the unit becomes more sensitive when it loses and becomes less sensitive otherwise.

1.4 Literature review

The earliest neural network model introduced by Hopfield employed two-state neurons [35]. Later, he came up with a neuron model that uses a continuous nonlinear function for input-output mapping [36]. An implementation of this model using electrical components

is shown in Figure 1.4. The continuous-time model is characterized by a first-order ordinary nonlinear differential equation of the form

$$C_i \frac{du_i}{dt} = \sum_{j=1}^{n} T_{ij} v_j + I_i - \frac{1}{g_i} u_i \tag{1.5}$$

where u_i is the input voltage of neuron i, T_{ij} is the strength of the connection between neurons i and j, I_i is the external input, g_i is the total conductance, and C_i is the input capacitance to the input of neuron i. The quantity v_j is the output of neuron j and $v_j = g(u_j)$ is a nonlinear mapping. $\mathbf{T}$ is a symmetric matrix with zero diagonal entries, that is, $T_{ii} = 0$ and $T_{ij} = T_{ji}$. These networks are associated with an energy function in the n-dimensional output space given by

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} T_{ij} v_i v_j - \sum_{i=1}^{n} I_i v_i + \sum_{i=1}^{n} g_i \int_0^{v_i} g_i^{-1}(z) dz \tag{1.6}$$

The corresponding energy function in the state space u^n of an asymptotically stable system would be the system's Liapunov function [72]. It has been shown that the single-layer recurrent network evolves through its states in such a way that the energy function is minimized and the stable states of the network correspond to the local minima of the energy function.

Stability and convergence are the two main issues with Hopfield networks. It has been proved that the network is asymptotically stable and the network always reaches one of the stable states. In order to find the equilibrium points, one can differentiate the energy function with respect to the outputs and equate the resulting expression to zero. However, differentiating once more reveals that the Hessian matrix of the system is the negative of the weight matrix $\mathbf{W}$. Since the diagonal entries of the weight matrix are all zero, the system has neither minima or maxima because $\mathbf{W}$ is neither positive or negative definite [72]. This means that the energy function $E(\mathbf{v})$ possesses no unconstrained minima. It has been proved in [53], [54] that the constrained minima for the binary case are the vertices of the $[-1, 1]$ hypercube. In the continuous-time case where the gains of the neurons are very high, the minima would again be at the cube corners. However, for finite-gain neurons, the minima are within the cube, close to the corners. Since the system equation has more than one solution, the solution reached by the actual network depends on the system parameters and initial conditions.

Since the Hopfield network minimizes the energy function, it can be used in optimization problems by formulating a cost function that coincides with the energy function of the

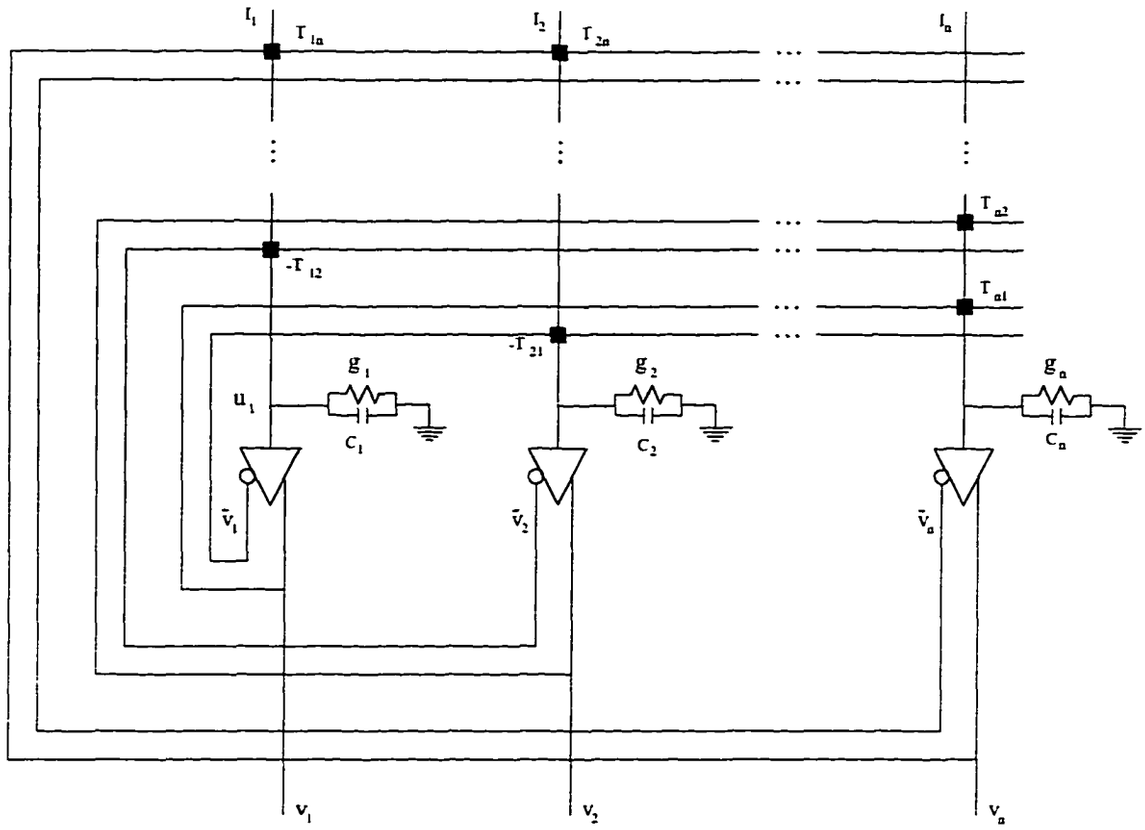


Figure 1.4: Continuous-time Hopfield network. The black square elements are the weights.

network. The search for an energy minimum performed by gradient-type networks corresponds to the search for a solution in an optimization problem. Hopfield and Tank used this network to solve the traveling-salesman problem [37], and to design A/D converters, signal-decision circuits, and linear programming circuits [34]. The results were inspiring and led to the publication of a number of papers dealing with other applications (for a list, please refer to [64]). The problems associated with these networks are that the solutions may or may not be global. In addition, the network may give rise to spurious states which are not stable [47]. The final states of the network are very much dependent on the initial states of the network. A number of papers have been published in recent years addressing these problems [6], [24], [25], [49], [53], and [60]. Some innovative networks along with their learning algorithms have been developed [25], [18]. In effect, the linear programming network has great potential in the field of signal processing.

In a linear programming problem, a cost function

$$c = \mathbf{a}^T \mathbf{v} \tag{1.7}$$

is minimized subject to a set of constraints

$$\mathbf{w}_j^T \mathbf{v} \geq b_j \ , \ j = 1, \cdots, m \tag{1.8}$$

where $\mathbf{a} = [a_1 \ a_2 \ \cdots \ a_n]^T$, $\mathbf{v} = [v_1 \ v_2 \ \cdots \ v_n]^T$, and $\mathbf{w}_j = [w_{j1} \ w_{j2} \ \cdots \ w_{jn}]^T$. The network consists of two sets of neurons. There are n linear neurons (g neurons) whose outputs (v_i) constitute the variables of the problem and m nonlinear f neurons whose outputs represent constraint satisfaction. The network employed by Hopfield is shown in Figure 1.5. The

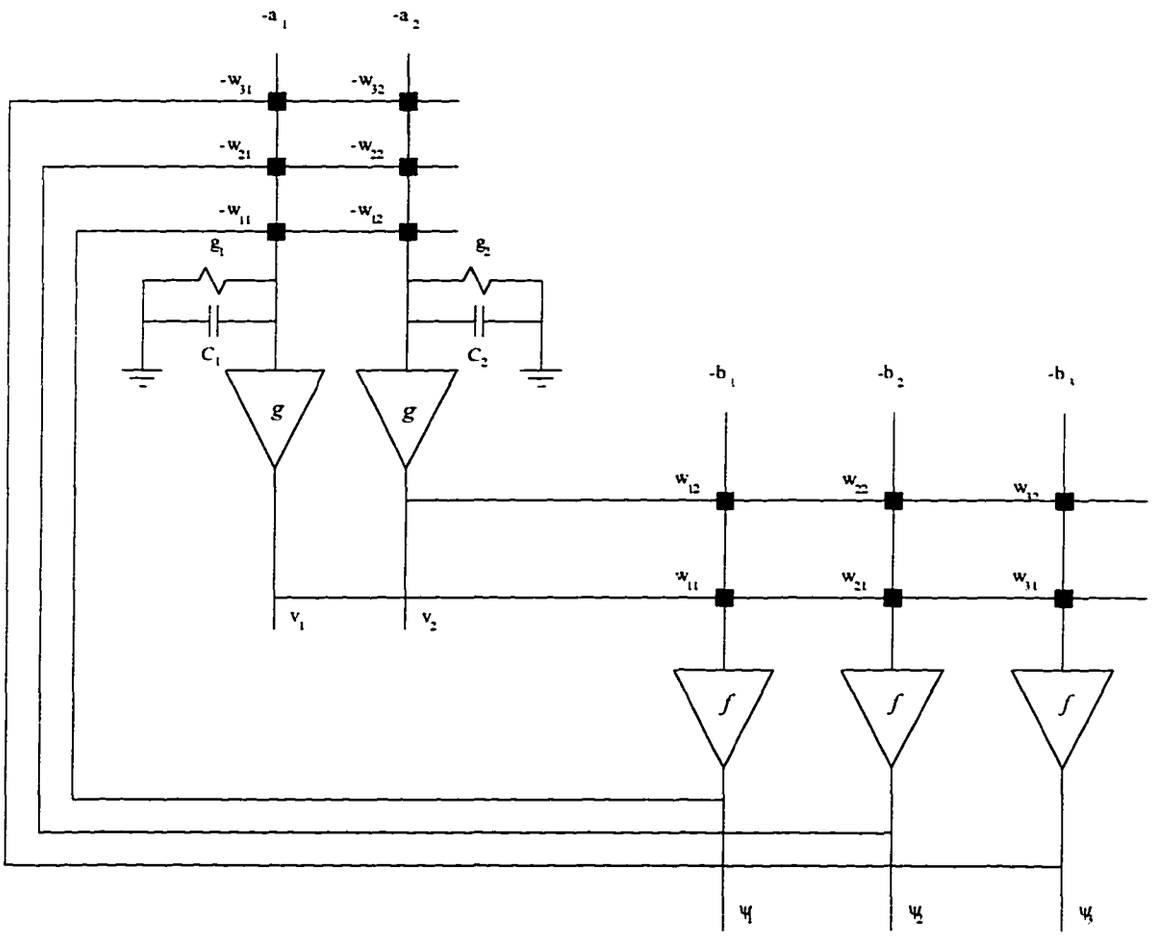


Figure 1.5: Hopfield's linear programming network.

detailed operation and the convergence characteristics for the network are discussed in [65] and [72].

The design of one-dimensional (1-D) finite-duration impulse response (FIR) filters based on Hopfield's linear programming network has been considered in [48]. A switched-capacitor integrator was used to realize each neuron. Adaptive FIR filters using a similar network were reported in [22], [23], and [21]. A least-square (LS) error function was minimized for the system identification of a low-order plant. A similar work for realizing adaptive FIR filter with a non-LS cost function has been reported in [26].

Kennedy and Chua [40] used integrator cells to model neurons and mapped the cost function and constraints into a canonical nonlinear circuit. Their model can be used for both linear and nonlinear programming. Rodriguez-Vazquez et al. [66] used a switched-capacitor network to realize the cost function and constraints. Another improved version of the network was reported in [17].

While Hopfield-type neural networks made their appearance in recent years, the history of feedforward neural networks can be traced back to the 40's when McCulloch and Pitts [72] first outlined the formal model of an elementary neuron. Since then different neural models and learning algorithms have been proposed [59], [72], [45], and [38]. The backpropagation learning algorithm formulated by the Parallel Distributed Processing group at MIT has been extensively used by researchers. Even dynamic backpropagation has been developed for recurrent neural networks for identification and control of nonlinear dynamical systems [50], [51]. Another network that has been studied extensibly is Kohonen's winner-take-all network [72], [45].

1.5 Motivation for and organization of the thesis

Many problems in science and engineering involve solving a set of linear equations [49]. In many applications like robotics, signal processing, and automatic control, an on-line solution of a set of linear equations is desired. For such real-time applications, a digital computer may not have the required speed or its use may be too expensive. One possible approach for solving such problems would be to use analog artificial neural networks such as Hopfield's linear programming network. The task at hand can be formulated as a cost function for an optimization problem. The cost function thus formed can be made identical with the associated energy function of the network. By virtue of its construction, the network minimizes the energy function and hence the cost function of the optimization

problem. When steady state is reached, the solution of the optimization problem is available at the output of the network.

The motivation for designing these kinds of networks is two-fold. First, by constructing and studying a simple network, one can possibly extend the idea for solving more complex problems. Second, the network can be implemented in analog VLSI. One can argue about the precision of an analog implementation. However, because of the massive connectivity, the error introduced in the network by the mismatch among devices and low noise immunity is tolerated. On top of that, one needs basically multipliers, adders, and amplifiers as the building blocks for such networks and all these components are more easily and efficiently implemented in analog VLSI than in digital VLSI.

Digital filters are essential tools for signal processing. In this thesis, a simple network like Hopfield's linear programming network for designing 1-D FIR filters is proposed. The idea is then extended to designing other complex filters. Since the network can be implemented in VLSI, filters thus obtained can be designed in real-time.

As for using neural networks to perform signal processing directly, two neural network schemes have been developed for classifying lidar waveforms into various categories. The motivation being their potential impact on fishing industries. From our regular contact with the relevant industries, it was apparent that there is a dire need for a reliable methodology to process lidar waveforms for various tasks. For stock estimation and preservation of fish population, spatial distribution of different fish species is desirable. On the other hand, if we could overlap the spatial distribution of various sea-bed structures, researchers may be able to deduce aspects of the fish habits which would have a tremendous impact on commercial fisheries.

The idea of designing 1-D FIR filters with Hopfield type neural networks is introduced in chapter 2. The three most common design techniques are addressed, namely, design on the basis of a given amplitude response, design on the basis of a given frequency response, and design to achieve prescribed specifications. The cost function for each case is first formulated and the associated network is then constructed. Detailed convergence and stability criteria are included along with examples. It is proved in each of the three networks that the solutions obtained by the network are global. Though the mathematical formulations are presented for linear-phase FIR filters with symmetrical impulse responses, the same networks can be used for any other symmetries.

Chapter 3 deals with the design of 2-D FIR filters. Two design methods are included, one for designing a filter based on a given amplitude response and the other to achieve prescribed specifications. Filters with quadrantal symmetry are considered here and a similar idea can be used for other types of filters. A detailed design procedure is included and the convergence properties are studied.

The design of 1-D infinite-duration impulse response (IIR) filters is examined in chapter 4. The first part deals with the design of IIR filters in magnitude-squared domain while the second part is concerned with the design of IIR filters based on given amplitude and phase responses.

The same idea is extended in chapter 5 to the design of equiripple FIR filters. By using a few extra components along with the network presented in chapter 2, it is shown that the network can solve weighted least-squares approximation problems for designing equiripple FIR filters, digital differentiators, and Hilbert transformers.

In chapter 6, real-life lidar waveforms have been classified by using both single-layer and multi-layer neural networks. Waveforms obtained off the coast of Vancouver Island representing milt of varied densities are successfully classified. The multi-layer network is also trained to extract information regarding the quality of water from lidar waveforms obtained from the Canadian Arctic region.

In chapter 7, the basic building blocks for implementing feedback networks are identified and designed. One network for designing 1-D linear-phase FIR filters for a given amplitude response has been simulated. It is shown that the network is able to generate the filter coefficients in a few microseconds. All the design and simulation work is done using the CAD package Cadence.

Chapter 8 summarizes the conclusions of the thesis and outlines recommendations for further studies.

Chapter 2

1-D FIR Filters

2.1 Introduction

The two most traditional methods for designing FIR filters are the window method and by optimization [5], [56]. The window method achieves closed-form solutions and hence it is much less computationally intensive. However, this method gives suboptimal solutions, that is, it is always possible to design a filter with the same order but with a better performance. In optimization methods, a cost function is constructed which is then minimized with respect to the filter coeffiecients. These techniques almost always require extensive computation and the final solutions depend heavily on the initial choice of the filter coefficients. By selecting a proper optimization algorithm, it is possible to achieve a good solution. Global convergence is not guaranteed but the overall performance of these procedures is normally quite acceptable.

Both these methods are essentially software oriented and implementing them in terms of hardware is very difficult, if not impossible. Furthermore, it is not possible to use these techinques in real-time signal processing. On the other hand, a computational method based on feedback neural networks can be devised that can be used to minimize a cost function for designing filters in real time [15], [8]. The network is constructed in such a way that the energy function of the network coincides with the cost function of the optimization problem. The evolution of the system is, in general, in the direction of the negative gradient of the energy function.

In this chapter, three networks are proposed for designing FIR filters. The next section deals with the design of FIR filters on the basis of the given amplitude response. In section

2.3, another network is proposed for designing any FIR filters with arbitrary amplitude and phase responses. Section 2.4 deals with the design of filters satisfying prescribed specifications. Detailed mathematical analyses with the convergence characteristics for each of the three methods are provided.

2.2 Design of FIR filters based on a given amplitude response

2.2.1 Theoretical background

The transfer function of a causal FIR filter is given by

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

where N is the length of the filter. Assuming that the sampling frequency is 2π, i.e., the sampling period is 1 s, the frequency response of the filter is given by

$$H(e^{j\omega}) = M(\omega) e^{j\theta(\omega)} = \sum_{n=0}^{N-1} h(n) e^{-j\omega}$$

where

$$M(\omega) = |H(e^{j\omega})|$$

is the amplitude response and

$$\theta(\omega) = \arg H(e^{j\omega})$$

is the phase response. For a linear phase FIR filter, the impulse response needs to be symmetrical or antisymmetrical, i.e.,

$$h(n) = h(N-1-n)$$

or

$$h(n) = -h(N-1-n) \quad \text{for } 0 \leq n \leq N-1$$

Depending on the length of the filter, the symmetry is about sample $(N-1)/2$ for odd case and midpoint between samples $(N-2)/2$ and $N/2$ for the even case.

For a symmetrical impulse response with N odd, the amplitude response is given by [5]

$$M(\omega_j) = |G(\omega_j)|$$

where

$$G(\omega_j) = \sum_{k=0}^{(N-1)/2} a_k \cos \omega_j k$$
$$= \mathbf{a}^T \mathbf{c}(\omega_j)$$

$$n = \frac{N-1}{2}$$
$$a_k = 2h\,(n-k)$$
$$a_0 = h(n)$$
$$\mathbf{c}(\omega_j) = [1 \;\; \cos\omega_j \;\; \cdots \;\; \cos\omega_j n]^T$$

Let the desired amplitude response be defined at m frequency points. The sum-squared error over all the frequency points is given by

$$E = \sum_{j=1}^{m} [d(\omega_j) - G(\omega_j)]^2 \tag{2.1}$$

where $d(\omega_j)$ is the ideal response at frequency ω_j.

The proposed network is shown in Figure 2.1. There are $(N\text{-}1)/2$ g neurons and m f neurons. The connection matrix between these two sets of neurons is fixed and consists of the cosines of the weighted frequency points. Both g and f neurons are linear constant-gain amplifiers and are dynamical in the sense that they have RC blocks at their inputs. The time constants of f neurons are very small compared to those of g neurons: hence they can be considered non-dynamical.

Neglecting the dynamics, the input voltage of an f neuron is expressed as

$$v_j = d(\omega_j) - \sum_{k=0}^{n} a_k \cos \omega_j k \tag{2.2}$$

The input-output relationship of an f neuron is given by

$$e_j = f(v_j) = \lambda_f R_f v_j \tag{2.3}$$

where e_j is the output voltage of neuron j, λ_f is the gain, and R_f is the input resistance of an f neuron. Substituting (2.2) in (2.3), we get

$$e_j = \lambda_f R_f \left[d(\omega_j) - \mathbf{a}^T \mathbf{c}(\omega_j) \right]$$

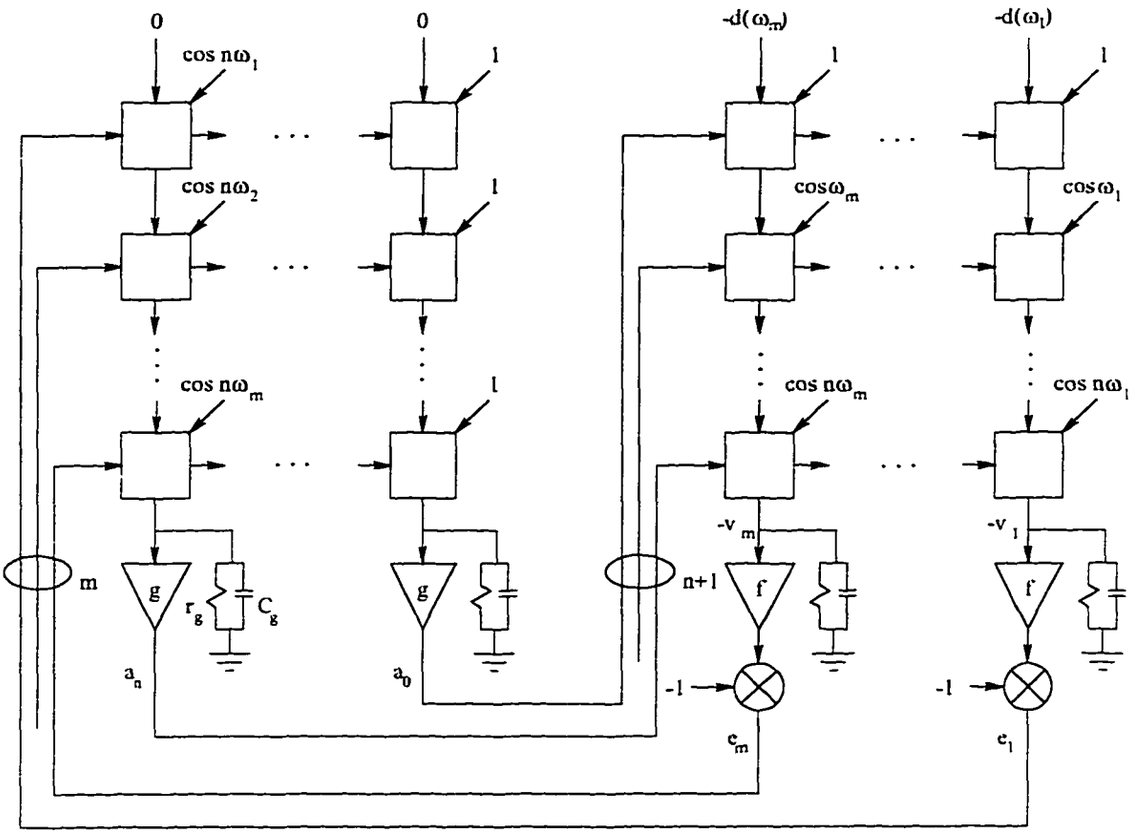


Figure 2.1: Neural network for the design of FIR filters.

for $j = 1, \cdots, m$. The quantity λ_f is the gain of an f neuron and R_f is the input resistance. The m-tuple error vector is given by

$$\begin{aligned}\mathbf{e} &= \lambda_f R_f \left\{ \mathbf{d} - \begin{bmatrix} 1 & \cos\omega_1 & \cdots & \cos\omega_1 n \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \cos\omega_{m-1} & \cdots & \cos\omega_{m-1} n \\ 1 & \cos\omega_m & \cdots & \cos\omega_m n \end{bmatrix} \mathbf{a} \right\} \\ &= \lambda_f R_f [\mathbf{d} - \mathbf{C}\mathbf{a}] \end{aligned} \tag{2.4}$$

where $\mathbf{d} = [d(\omega_1)\ d(\omega_2)\ \cdots\ d(\omega_n)]^T$ is the vector of the desired amplitude response and $\mathbf{C}$ is the matrix formed by the cosines.

The equation of motion of a g neuron is given by

$$C_g \frac{du_i}{dt} + \frac{u_i}{r_g} = \sum_{j=1}^{m} (e_j - u_i) \cos i\omega_j \tag{2.5}$$

for $i = 0, \ldots, n$ where u_i is the input voltage of neuron i and r_g and C_g are the input resistance and capacitance of a g neuron, respectively. Eq. (2.5) can be rewritten as

$$C_g \frac{du_i}{dt} + \left(\frac{1}{r_g} + \sum_{j=1}^{m} \cos i\omega_j \right) u_i = \sum_{j=1}^{m} e_j \cos i\omega_j$$

$$C_g \frac{du_i}{dt} + G_i u_i = \sum_{j=1}^{m} e_j \cos i\omega_j \tag{2.6}$$

where

$$G_i = \frac{1}{r_g} + \sum_{j=1}^{m} \cos i\omega_j$$

is the total input conductance of neuron i. The differential equation governing all g neurons now can be expressed as

$$C_g \frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \begin{bmatrix} 1 & \cdots & 1 & 1 \\ \cos\omega_1 & \cdots & \cos\omega_{m-1} & \cos\omega_m \\ \vdots & \cdots & \vdots & \vdots \\ \cos\omega_1 n & \cdots & \cos\omega_{m-1} n & \cos\omega_m n \end{bmatrix} \mathbf{e}$$

$$= \mathbf{C}^T \mathbf{e} \tag{2.7}$$

where $\mathbf{e} = [e_1\ e_2\ \ldots\ e_m]^T$ is the error vector as defined by (2.4). $\mathbf{u} = [u_0, u_1, \ldots, u_n]^T$ is the vector formed by the input voltages of the g neurons, and $\mathbf{G} = \mathrm{diag}(G_0, G_1, \cdots, G_n)$ is a diagonal matrix formed by the input conductances of the g neurons. The input output relationship of a g neuron is given by

$$a_i = g(u_i) = \lambda_g u_i \tag{2.8}$$

where λ_g is the gain of the g neuron.

2.2.2 Convergence

The energy function of the network is given by

$$E = \sum_{j=1}^{m} F(v_j) + \sum_{i=0}^{n} G_i \int_0^{a_i} g^{-1}(a_i) da_i \tag{2.9}$$

where v_j is defined in (2.2). F is related to the characteristic function of an f neuron by

$$\frac{dF(x)}{dx} = f(x) \tag{2.10}$$

x being a dummy variable. Substituting (2.10) and (2.8) in (2.9) and carrying out the integration, we get

$$E = \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m}\left[d(\omega_j) - \sum_{k=0}^{n} a_k \cos k\omega_j\right]^2 + \sum_{i=0}^{n}\frac{1}{2}\frac{G_i}{\lambda_g}a_i^2 \tag{2.11}$$

In all practical cases, G_i/λ_g is very small, as will be shown below, and the energy function coincides with the sum-squared error function as given in (2.1) to within a constant factor.

In order to prove that the network converges to one of the minima, we have to show that the time derivative of the energy function is always less than or equal to zero. We can write

$$\frac{dE}{dt} = \sum_{i=0}^{n}\left(\frac{\partial E}{\partial a_i}\right)\left(\frac{da_i}{dt}\right) \tag{2.12}$$

and from (2.9), we get

$$\frac{\partial E}{\partial a_i} = \sum_{j=1}^{m}\frac{\partial F(v_j)}{\partial a_i} + G_i u_i \tag{2.13}$$

Now

$$\frac{\partial F(v_j)}{\partial a_i} = \frac{\partial F(v_j)}{\partial v_j}\frac{dv_j}{da_i} = -f(v_j)\cos i\omega_j \tag{2.14}$$

Substituting (2.14) in (2.13), we get

$$\frac{\partial E}{\partial a_i} = -\sum_{j=1}^{m} f(v_j)\cos i\omega_j + G_i u_i \tag{2.15}$$

Finally substituting (2.15) and (2.6) in (2.12) and noting that $f(v_j) = \epsilon_j$, we have

$$\frac{dE}{dt} = -C_g\sum_{i=0}^{n}\left(\frac{du_i}{dt}\right)\left(\frac{da_i}{dt}\right) \tag{2.16}$$

Since $a_i = g(u_i) = \lambda_g u_i$, the time derivative of the energy function can be written as

$$\frac{dE}{dt} = -\frac{C_g}{\lambda_g}\sum_{i=0}^{n}\left(\frac{da_i}{dt}\right)^2 \tag{2.17}$$

Since C_g and λ_g are positive, (2.17) is always positive when da_i/dt is not equal to zero, and is zero when da_i/dt is zero at the equilibrium point. This means that the network always settles to one of the mimima on the error surface.

In order to check whether the network converges to the global minimum, we have to compute the system equation. From (2.7) and (2.4), we get

$$C_g\frac{d\mathbf{u}}{dt} + \mathbf{Gu} = \lambda_f R_f \mathbf{C}^T[\mathbf{d} - \mathbf{Ca}] \tag{2.18}$$

Substituting (2.8) in (2.18), we get

$$\frac{d\mathbf{a}}{dt} + \frac{1}{C_g}\mathbf{G}\mathbf{a} = \frac{\lambda_f \lambda_g R_f}{C_g}\mathbf{C}^T[\mathbf{d} - \mathbf{C}\mathbf{a}] \tag{2.19}$$

Finally rearranging the terms

$$\frac{d\mathbf{a}}{dt} + \left[\frac{1}{C_g}\mathbf{G} + \frac{\lambda_f \lambda_g R_f}{C_g}\mathbf{C}^T\mathbf{C}\right]\mathbf{a} = \frac{\lambda_f \lambda_g R_f}{C_g}\mathbf{C}^T\mathbf{d} \tag{2.20}$$

It can be easily proved that the system matrix is always positive definite, which is the necessary and sufficient condition for the network to settle to the global minimum.

In order to compare the solution obtained by the network to the optimal solution, we can rewrite the cost function as

$$E = (\mathbf{d} - \mathbf{C}\mathbf{a})^T(\mathbf{d} - \mathbf{C}\mathbf{a}) \tag{2.21}$$

Computing the derivatives with respect to the elements of $\mathbf{a}$ and equating to zero, one gets

$$2\mathbf{C}^T\mathbf{C}\mathbf{a} - 2\mathbf{C}^T\mathbf{d} = 0 \tag{2.22}$$

The optimal solution can be written as

$$\mathbf{a}_{opt} = \left(\mathbf{C}^T\mathbf{C}\right)^{-1}\left(\mathbf{C}^T\mathbf{d}\right) \tag{2.23}$$

Now we can derive an expression for the solution obtained by the proposed network. The energy function (2.11) can be rewritten as

$$E = \frac{1}{2}\lambda_f R_f (\mathbf{d} - \mathbf{C}\mathbf{a})^T(\mathbf{d} - \mathbf{C}\mathbf{a}) + \frac{1}{2\lambda_g}\mathbf{a}^T\mathbf{G}\mathbf{a} \tag{2.24}$$

Differentiating (2.24) with respect to the elements of $\mathbf{a}$ and equating to zero, we get

$$\lambda_f R_f\left(\mathbf{C}^T\mathbf{C}\mathbf{a} - \mathbf{C}^T\mathbf{d}\right) + \frac{1}{\lambda_g} = 0$$

Rearranging the terms, we get

$$\mathbf{C}^T\mathbf{C}\mathbf{a} + \frac{1}{\lambda_f \lambda_g R_f}\mathbf{G}\mathbf{a} = \mathbf{C}^T\mathbf{d}$$

The solution is now given by

$$\mathbf{a}_s = \left(\mathbf{C}^T\mathbf{C} + \frac{1}{\lambda_f \lambda_g R_f}\mathbf{G}\right)^{-1}\mathbf{C}^T\mathbf{d} \tag{2.25}$$

Since the denominator of the second term is much much larger than the numerator, the second term can be neglected and the solution given by (2.25) coincides with the optimal solution in (2.23).

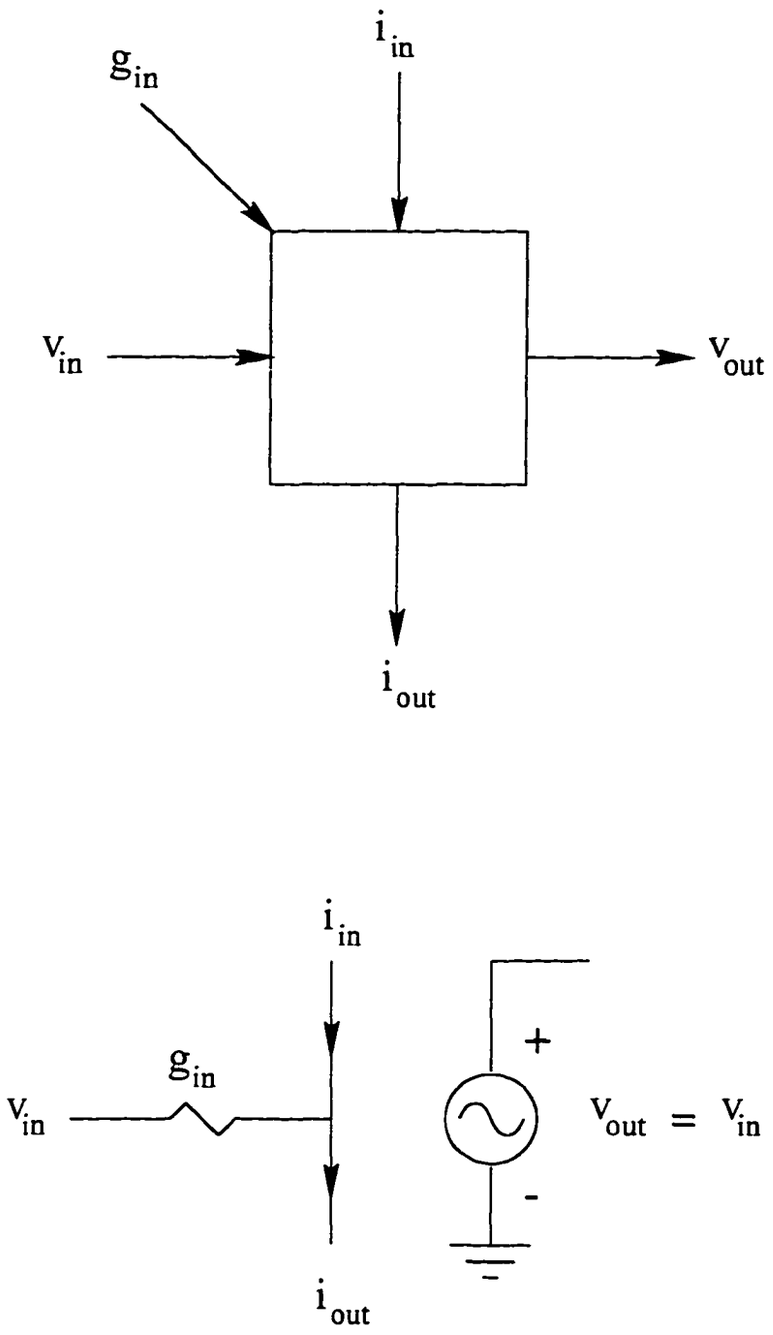


Figure 2.2: (a) Connection element of Figure 2.1 with all input and output signals. (b) schematic representation of the element.

2.2.3 The network

Each square element in Figure 2.1 represents a connection point. It accepts the input signal from the left and passes it to the next column through to the right. The input current is fed

from the top and the conductances are fed from the left or the right corners. Each element multiplies the input voltage by the conductance, adds the input current from the top and produces an output current which is fed to the next connection point immediately below it. A circuit representation of the connection element is shown in Figure 2.2. The neural network uses identical connection elements for both types of neurons and identical neurons as well. Since the time constants are different for g and f neurons, the input resistor-capacitor blocks are different too and have elements $R_g = 2$ kΩ, $C_g = 0.04$ μF, $R_f = 50$ Ω and $C_f = 1.0$ nF. Both g and f neurons are linear constant-gain amplifiers with gains λ_g and λ_f, respectively, of 2000. The desired response signals and the cosine conductances are scaled by a factor of 10^{-5} to prevent saturation.

Initially outputs of both f and g neurons are set to zero. Conductance and desired response values are applied and then the network is set free to evolve. In actual simulation, f neurons are updated first and the updated values of f neurons are used to compute new values for g neurons. This procedure is continued until the desired accuracy in filter coefficients is achieved. To check the convergence, the outputs of g neurons are compared with the output values obtained from the previous iteration. If the changes in outputs of all g neurons are less than a small quantity (say 10^{-6}), then the procedure is terminated and the outputs of the g neurons represent the filter coefficients.

2.2.4 Results

The network in Figure 2.1 was simulated to design a linear phase FIR filter with passband edge $\omega_p = 0.4\pi$, stopband edge $\omega_a = 0.5\pi$, with a sampling frequency $\omega_s = 2\pi$. The number of sample points were fixed to 40 and 50 in the passband and stopbands, respectively. The length N of the filter was set to 21. The number of f neurons was 90 and the number of g neurons 11. The network was simulated until the desired accuracy was achieved. The amplitude response of the filter is shown in Figure 2.3. The variation of the filter coefficients with respect to the number of iterations is illustrated in Figure 2.4. The absolute error between the desired response and the response obtained by the filter is depicted in Figure 2.5.

In order to justify the fact that G_i/λ_g is small, we note that $\lambda_g = 2000$, and G_i is given

by

$$G_i = \frac{1}{r_g} + \sum_{j=1}^{m} \cos i\omega_j$$

$r_g = 2000$ and, for this particular example, the maximum value of the sum term is 9×10^{-4}. It is to be noted that the cosine terms were already multiplied by 10^{-5}. So the maximum value of the quantity G_i/λ_g is 0.0014/2000 which is very small and can be ignored.

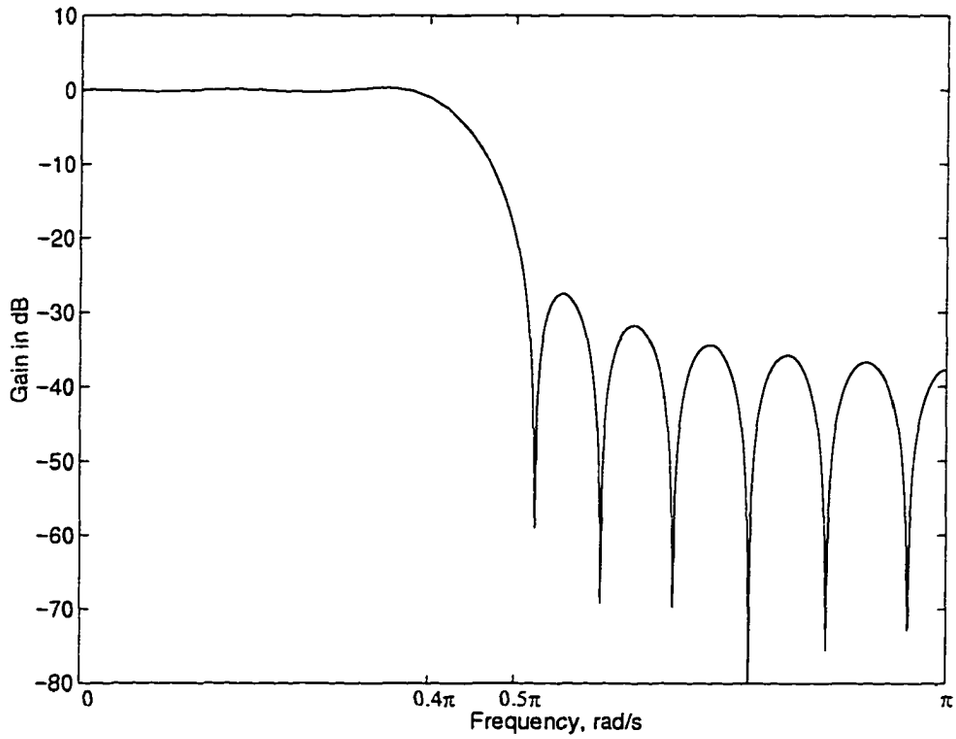


Figure 2.3: Amplitude response of lowpass filter with $\omega_p = 0.4\pi$, $\omega_a = 0.5\pi$. The length of the filter is 21.

In order to compare the results obtained by the proposed method with other standard methods, the same filter was designed using a quasi-Newton algorithm. The amplitude response of the filter is shown in Figure 2.6. The difference in responses is shown in Figure 2.7.

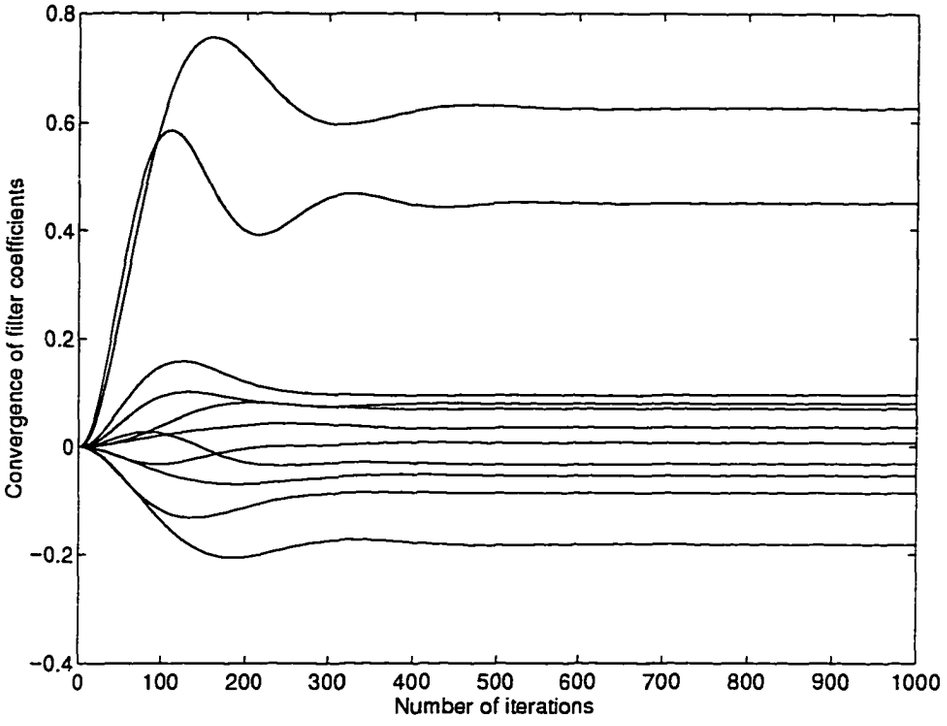


Figure 2.4: Convergence characteristic of the algorithm.

2.3 Design of FIR filters based on a given frequency response

2.3.1 Design details

The frequency response of a causal FIR filter of length N is given by

$$H(e^{jw}) = \sum_{n=0}^{N-1} h_n e^{-jn\omega} = H_r(\omega) - jH_i(\omega) \tag{2.26}$$

where $H_r(\omega)$ and $H_i(\omega)$ are the real and imaginary parts of the frequency response and are given by

$$\begin{aligned} H_r(\omega) &= \sum_{n=0}^{N-1} h_n \cos n\omega \\ H_i(\omega) &= \sum_{n=0}^{N-1} h_n \sin n\omega \end{aligned} \tag{2.27}$$

The desired response of the filter at a frequency point ω_j can be represented by

$$d(\omega_j) = d_r(\omega_j) + jd_i(\omega_j) \tag{2.28}$$

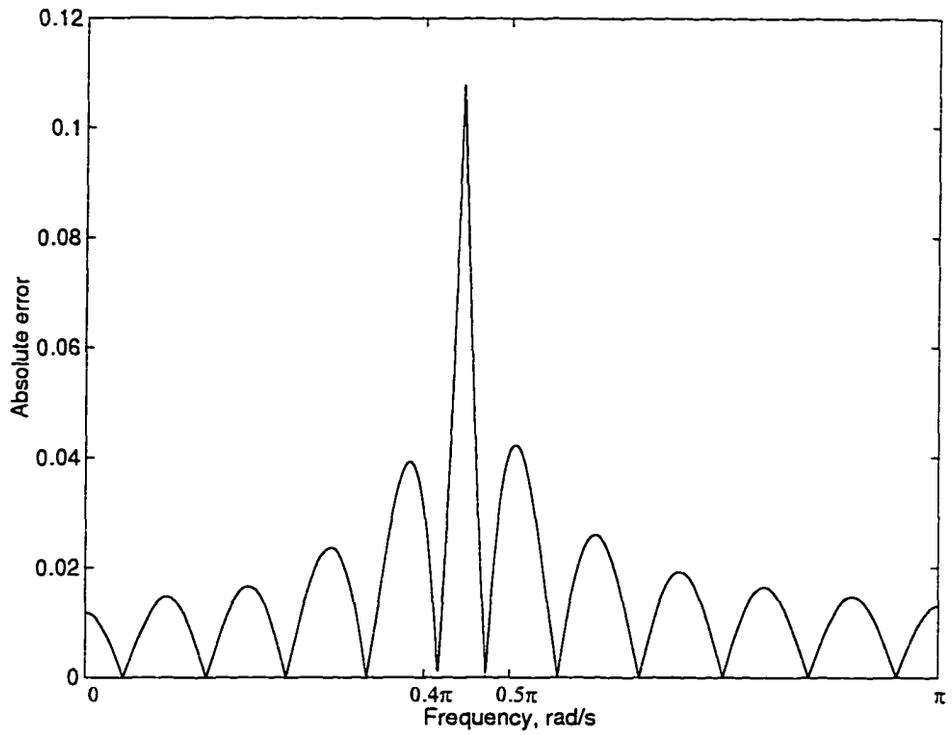


Figure 2.5: Absolute error between the desired response and the response of the filter.

The sum-squared error function can now be written as

$$E = \sum_{j=1}^{m} |d(\omega_j) - H(\omega_j)|^2 \tag{2.29}$$

Substituting (2.26), (2.27), and (2.28) in (2.29) and manipulating the resulting equation, the error function can be rewritten as

$$E = \sum_{j=1}^{m} [d_r(\omega_j) - H_r(\omega_j)]^2 + \sum_{j=1}^{m} [d_i(\omega_j) + H_i(\omega_j)]^2 \tag{2.30}$$

The proposed network for the solution of this problem is shown in Figure 2.8. Since the least-square error has two parts, the network has two sets of f neurons represented by f_r and f_i corresponding to the real and the imaginary parts, respectively. The outputs of f_i neurons are inverted and fed back to the g neurons. This is required to make the time derivative of the energy function negative. As before, the input voltages of f_r and f_i neurons are denoted as

$$v_{r_j} = d_r(\omega_j) - \sum_{n=0}^{N-1} h_n \cos n\omega_j$$

$$v_{i_j} = d_i(\omega_j) + \sum_{n=0}^{N-1} h_n \sin n\omega_j$$

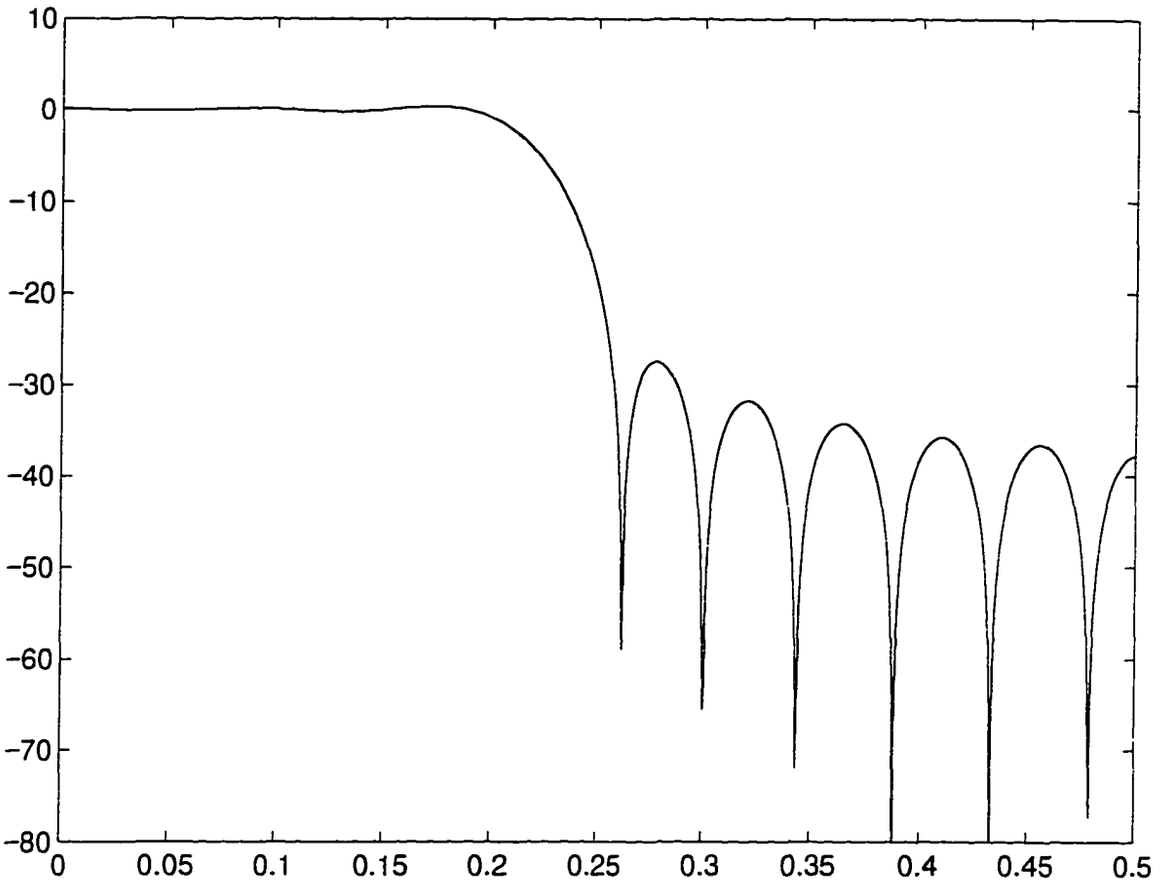


Figure 2.6: Amplitude response of the lowpass filter obtained by quasi-Newton optimization algorithm.

for $j = 1, 2, \ldots, m$. The output voltages are given by

$$e_{r_j} = f(v_{r_j}) = \lambda_f R_f v_{r_j}$$
$$e_{i_j} = f(v_{i_j}) = \lambda_f R_f v_{i_j}$$

The real part of the error vector can now be written as

$$\mathbf{e}_r = \lambda_f R_f [\mathbf{d}_r - \mathbf{C}\mathbf{h}] \tag{2.31}$$

where $\mathbf{d}_r = [d_r(\omega_1)\ d_r(\omega_2)\ \cdots\ d_r(\omega_m)]^T$ is the vector formed by the real part of the desired response and $\mathbf{h} = [h_0\ h_1\ \cdots\ h_{N-1}]^T$ is the impulse response vector. $\mathbf{C}$ is the matrix formed

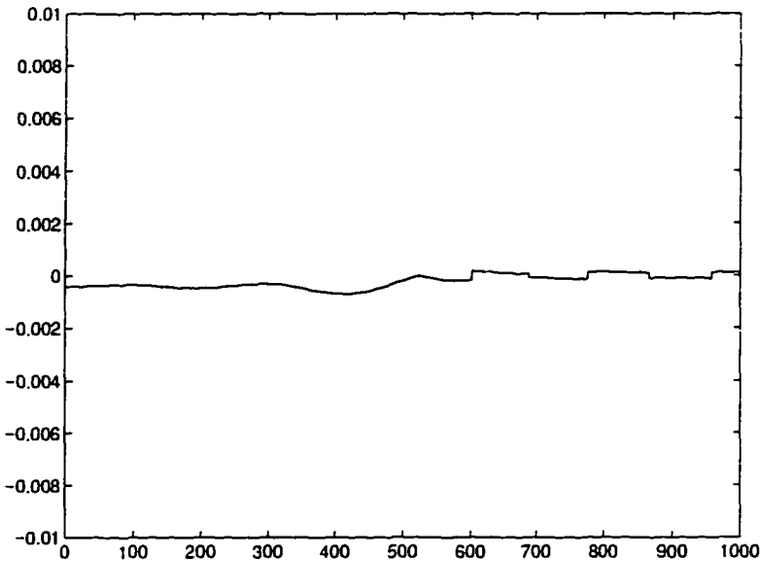


Figure 2.7: Difference of amplitude responses obtained by the proposed methos and quasi-Newton algorithm.

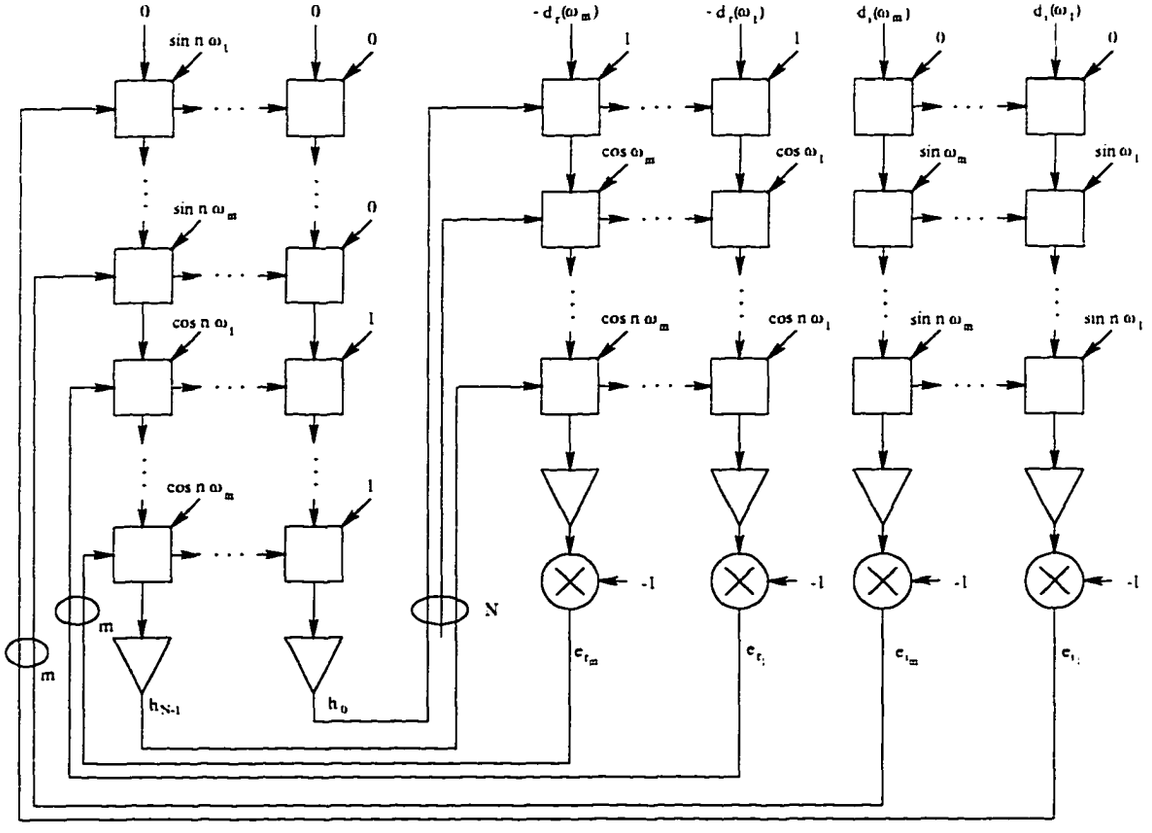


Figure 2.8: Network for designing FIR filters based on given amplitude and phase responses. The input resistor-capacitor blocks are not shown.

by the cosines of the weighted frequency points and is given by

$$\mathbf{C} = \begin{bmatrix} 1 & \cos\omega_1 & \cdots & \cos(N-1)\omega_1 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \cos\omega_{m-1} & \cdots & \cos(N-1)\omega_{m-1} \\ 1 & \cos\omega_m & \cdots & \cos(N-1)\omega_m \end{bmatrix}$$

Similarly, the imaginary part of the error vector can be represented by

$$\mathbf{e}_i = \lambda_f R_f [\mathbf{d}_i + \mathbf{Sh}] \tag{2.32}$$

where $\mathbf{d}_i = [d_i(\omega_1)\ d_i(\omega_2)\ \cdots\ d_i(\omega_m)]^T$ is the imaginary part of the desired response vector. $\mathbf{S}$ is a matrix formed by the sines of the weighted frequency points and is given by

$$\mathbf{S} = \begin{bmatrix} 0 & \sin\omega_1 & \cdots & \sin(N-1)\omega_1 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & \sin\omega_{m-1} & \cdots & \sin(N-1)\omega_{m-1} \\ 0 & \sin\omega_m & \cdots & \sin(N-1)\omega_m \end{bmatrix}$$

The equation of motion for the ith g neuron is given by

$$C_g \frac{du_i}{dt} + \frac{u_i}{r_g} = \sum_{j=1}^{m} \left(e_{r_j} - u_i\right) \cos i\omega_j - \sum_{j=1}^{m} \left(e_{i_j} + u_i\right) \sin i\omega_j$$

for $i = 0, 1, \ldots, N-1$. After rearranging the terms, one gets

$$C_g \frac{du_i}{dt} + G_i u_i = \sum_{j=1}^{m} e_{r_j} \cos i\omega_j - \sum_{j=1}^{m} e_{i_j} \sin i\omega_j$$

where G_i is the total input conductance of neuron i and is given by

$$G_i = \frac{1}{r_g} + \sum_{j=1}^{m} \cos i\omega_j + \sum_{j=1}^{m} \sin i\omega_j \tag{2.33}$$

The input-output characteristics of the g neuron is given by

$$h_i = g(u_i) = \lambda_g u_i \tag{2.34}$$

2.3.2 Convergence

The energy function of the network is given by

$$E = \sum_{j=1}^{m} F(v_{r_j}) + \sum_{j=1}^{m} F(v_{i_j}) + \sum_{i=0}^{n} G_i \int_0^{h_i} g^{-1}(h_i) dh_i \tag{2.35}$$

where F is as defined in (2.10). Substituting (2.10) and (2.34) in (2.35) and carrying out the integration, the energy function can be rewritten as

$$E = \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m} v_{r_j}^2 + \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m} v_{i_j}^2 + \sum_{i=0}^{N-1} \frac{1}{2}\frac{G_i}{\lambda_g} h_i^2 \tag{2.36}$$

As it was shown, G_i/λ_g is very small and the last term in (2.36) can be neglected. Under these conditions the energy function coincides with the error function given by (2.30).

As will be proved, the time derivative of the energy function is always non-positive. Since the error function is linearly related to the energy function, the error function also goes down with time. The time derivative of the energy function is given by

$$\frac{dE}{dt} = \sum_{i=0}^{N-1} \left(\frac{\partial E}{\partial h_i}\right)\left(\frac{dh_i}{dt}\right) \tag{2.37}$$

From (2.35) we get,

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{m} \frac{\partial F(v_{r_j})}{\partial h_i} + \sum_{j=1}^{m} \frac{\partial F(v_{i_j})}{\partial h_i} + G_i u_i \tag{2.38}$$

The first two terms on the right-hand side are given by

$$\begin{aligned} \frac{\partial F(v_{r_j})}{\partial h_i} &= -e_{r_j} \cos i\omega_j \\ \frac{\partial F(v_{i_j})}{\partial h_i} &= e_{i_j} \sin i\omega_j \end{aligned} \tag{2.39}$$

Substituting (2.39), and (2.38) in (2.37) and noting (2.3.1), we get

$$\frac{dE}{dt} = -C_g \sum_{i=0}^{N-1} \left(\frac{du_i}{dt}\right)\left(\frac{dh_i}{dt}\right) \tag{2.40}$$

Finally substituting (2.34)

$$\frac{dE}{dt} = -\frac{C_g}{\lambda_g} \sum_{i=0}^{N-1} \left(\frac{dh_i}{dt}\right)^2 \tag{2.41}$$

As can be seen, (2.41) is always less than or equal to zero.

The equation of motion of all g neurons can be written as

$$C_g \frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \mathbf{C}^T\mathbf{e}_r - \mathbf{S}^T\mathbf{e}_i$$

where $\mathbf{G}$ is the diagonal matrix of the conductances. Substituting $\mathbf{e}_r$ and $\mathbf{e}_i$ and rearranging terms,

$$C_g \frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \lambda_f R_f \left[\mathbf{C}^T\mathbf{d}_r - \mathbf{S}^T\mathbf{d}_i\right] - \lambda_f R_f \left[\mathbf{C}^T\mathbf{C} + \mathbf{S}^T\mathbf{S}\right]\mathbf{h}$$

Noting the fact that $h_i = \lambda_g u_i$, we get

$$\frac{d\mathbf{h}}{dt} + \frac{\lambda_f \lambda_g R_f}{C_g}\left[\frac{\mathbf{G}}{\lambda_f \lambda_g R_f} + \left(\mathbf{C}^T\mathbf{C} + \mathbf{S}^T\mathbf{S}\right)\right]\mathbf{h} = \frac{\lambda_f \lambda_g R_f}{C_g}\left(\mathbf{C}^T\mathbf{d}_r - \mathbf{S}^T\mathbf{d}_i\right)$$

As we can see, the system matrix is positive definite. So, the network settles to the global minimum.

The sum-squared error function can be written as

$$E = (\mathbf{d}_r - \mathbf{C}\mathbf{h})^T(\mathbf{d}_r - \mathbf{C}\mathbf{h}) + (\mathbf{d}_i - \mathbf{S}\mathbf{h})^T(\mathbf{d}_i - \mathbf{S}\mathbf{h})$$

The optimal solution for this problem is given by

$$\mathbf{h}_{opt} = \left(\mathbf{C}^T\mathbf{C} + \mathbf{S}^T\mathbf{S}\right)^{-1}\left(\mathbf{C}^T\mathbf{d}_r - \mathbf{S}^T\mathbf{d}_i\right) \tag{2.42}$$

The energy function of the network can be written as

$$E = \frac{1}{2}\lambda_f R_f (\mathbf{d}_r - \mathbf{C}\mathbf{h})^T(\mathbf{d}_r - \mathbf{C}\mathbf{h}) + \frac{1}{2}\lambda_f R_f (\mathbf{d}_i - \mathbf{S}\mathbf{h})^T(\mathbf{d}_i - \mathbf{S}\mathbf{h}) + \frac{1}{2\lambda_g}\mathbf{h}^T\mathbf{G}\mathbf{h}$$

After some manipulation, the solution obtained by the network can be written as

$$\mathbf{h}_s = \left(\mathbf{C}^T\mathbf{C} + \mathbf{S}^T\mathbf{S} + \frac{\mathbf{G}}{\lambda_f \lambda_g R_f}\right)^{-1}\left(\mathbf{C}^T\mathbf{d}_r - \mathbf{S}^T\mathbf{d}_i\right) \tag{2.43}$$

As we can see, (2.43) can be made arbitrarily as close as desired to optimal solution (2.42) by choosing high values for the denominator of the second term in (2.43).

2.3.3 Results

The network in Figure 2.8 was simulated to design an FIR filter with arbitrary amplitude and phase response as shown in Figure 2.9 (a) and (b), respectively. The length of the filter, which is the same as the number of g neurons, was set to 51. Figure 2.10 (a) and (b) shows the amplitude and the phase responses obtained by the proposed method. The absolute error in the amplitude between the desired response and the response of the filter obtained by the proposed network is shown in Figure 2.11(a) whereas Figure 2.11(b) shows the error in the phase response.

2.4 Design of FIR filters satisfying prescribed specifications

2.4.1 Theory

At times, it is required to design a FIR filter with given passband ripple (A_p) and stopband attenuation (A_a). The error between the desired response and the response of the obtained

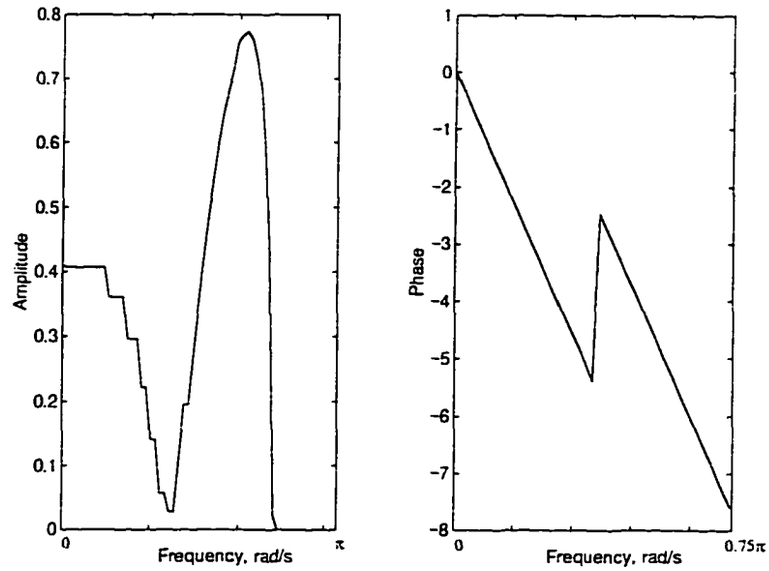


Figure 2.9: (a) Desired amplitude response and (b) phase response of the FIR filter.

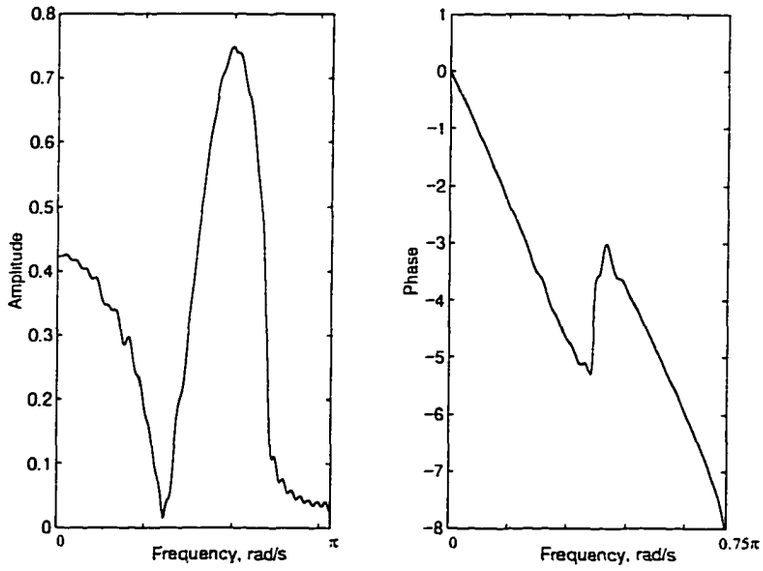


Figure 2.10: (a) Amplitude response and (b) phase response of the filter obtained.

filter is typically given by

$$-\delta(\omega_j) \leq d(\omega_j) - M(\omega_j) \leq \delta(\omega_j) \tag{2.44}$$

where ω_j is a frequency point and δ is defined as

$$\delta(\omega_j) = \begin{cases} \delta_p & \omega_j \in \text{passband} \\ \delta_a & \omega_j \in \text{stopband} \end{cases}$$

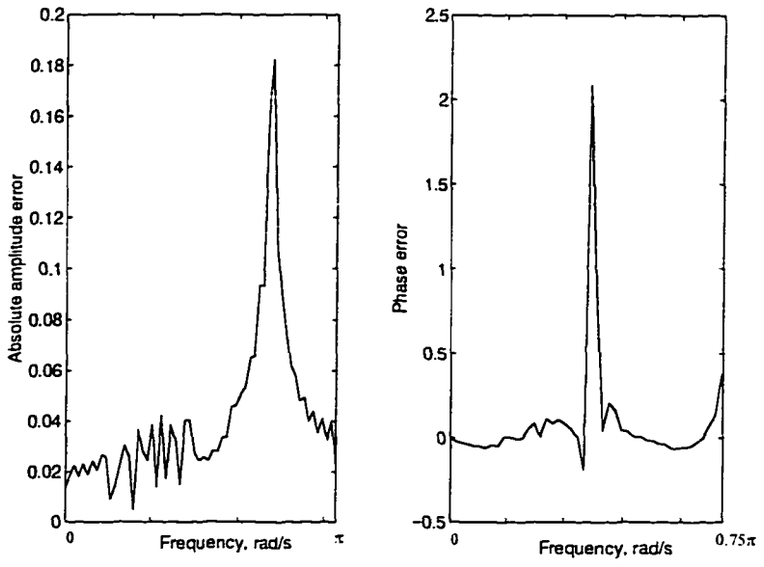


Figure 2.11: Error in (a) amplitude response and (b) phase response of the filter obtained.

where

$$\delta_p = \frac{10^{0.05A_p} - 1}{10^{0.05A_p} + 1}$$
$$\delta_a = 10^{-0.05A_a}$$

M is the amplitude response and for a linear-phase FIR filter of length N, it is given by

$$M(\omega_j) = |G(\omega_j)|$$
$$G(\omega_j) = \sum_{k=0}^{(N-1)/2} a_k \cos k\omega_j \tag{2.45}$$

From (2.44), we can form the error function

$$E = \sum_{j=1}^{m} [d(\omega_j) - G(\omega_j) - \delta(\omega_j)]^2 + \sum_{j=1}^{m} [-d(\omega_j) + G(\omega_j) - \delta(\omega_j)]^2 \tag{2.46}$$

The network that minimizes the above error function is shown in Figure 2.12. Here, we have two sets of f neurons corresponding to the two terms in the error function. As before, both f and g neurons are linear constant-gain amplifiers with resistor-capacitor blocks at their inputs. Ignoring the dynamics of the f neurons, the input voltages can be written as

$$v_{1_j} = d(\omega_j) - \sum_{k=0}^{N-1/2} a_k \cos k\omega_j - \delta(\omega_j)$$
$$v_{2_j} = -d(\omega_j) + \sum_{k=0}^{N-1/2} a_k \cos k\omega_j - \delta(\omega_j) \tag{2.47}$$

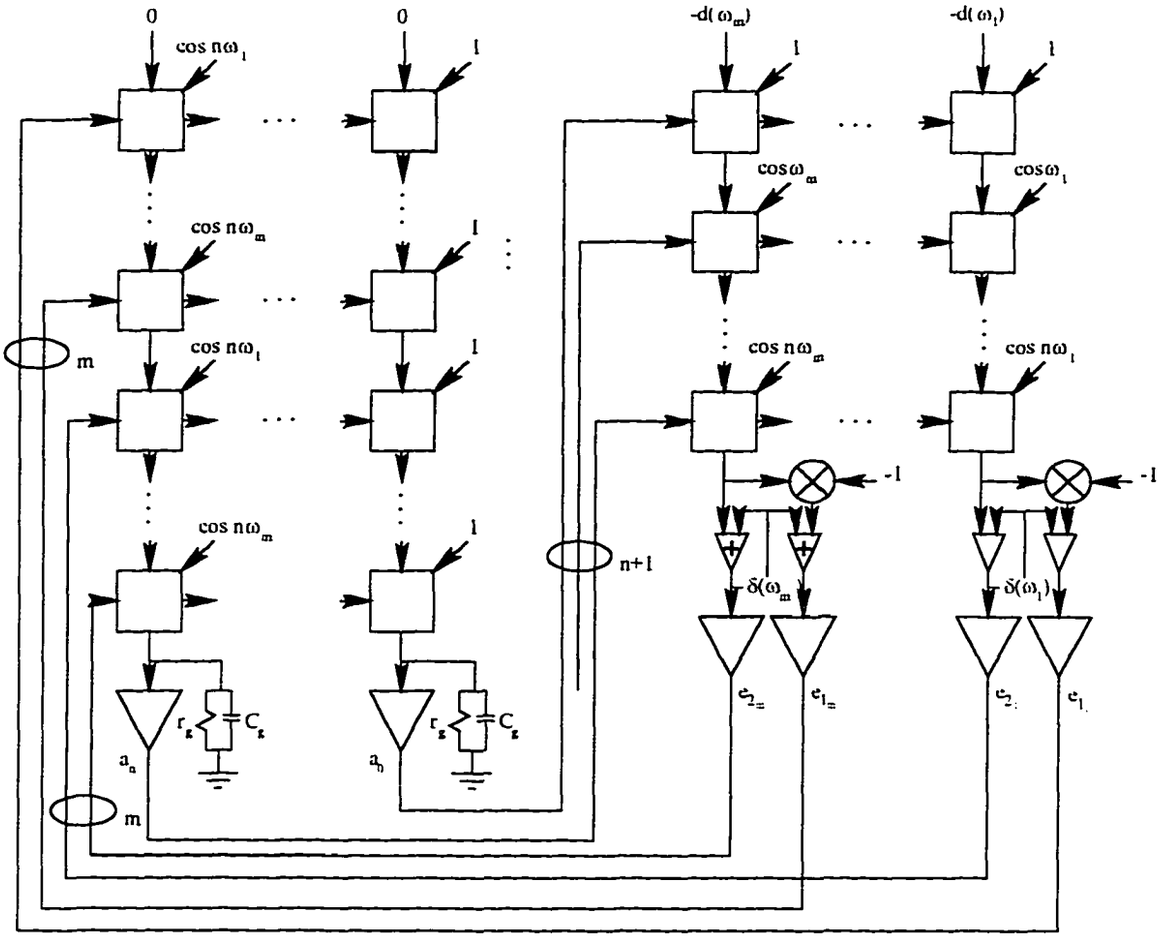


Figure 2.12: Network for designing FIR filters satisfying prescribed specifications. The input RC blocks for f neurons are not shown.

As we can see from (2.47), we can have only one set of connection matrices for the f neurons which computes the first two terms. We will invert the output of the matrices and then will subtract δ's both from the original outputs and their complements to form the input voltages for both f neurons. This scheme is represented in Figure 2.12. The output voltages of the f neurons are given by

$$e_{1_j} = f(v_{1_j}) = \lambda_f R_f v_{1_j}$$

$$e_{2_j} = f(v_{2_j}) = \lambda_f R_f v_{2_j}$$

for $j = 1, 2, \ldots, m$. For a g neuron i, the input-output equations are given by

$$C_g \frac{du_i}{dt} + \frac{u_i}{r_g} = \sum_{j=1}^{m} \left(e_{1_j} - u_i\right) \cos i\omega_j - \sum_{j=1}^{m} \left(e_{2_j} + u_i\right) \cos i\omega_j$$

$$a_i = g(u_i) = \lambda_g u_i \tag{2.48}$$

As before, the equation of motion can be rewritten as

$$C_g \frac{du_i}{dt} + +G_i u_i = \sum_{j=1}^{m} e_{1_j} \cos i\omega_j - \sum_{j=1}^{m} e_{2_j} \cos i\omega_j$$

where

$$G_i = \frac{1}{r_g} + 2\sum_{j=1}^{m} \cos i\omega_j$$

2.4.2 Convergence

The energy function of the network is given by

$$E = \sum_{j=1}^{m} F(v_{1_j}) + \sum_{j=1}^{m} F(v_{2_j}) + \sum_{i=0}^{N-1/2} G_i \int_0^{a_i} g^{-1}(a_i) da_i \tag{2.49}$$

where F is the same as defined in (2.10). Substituting (2.10) and (2.48) in (2.49) and carrying out the integration, the energy function can be rewritten as

$$E = \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m} v_{1_j}^2 + \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m} v_{2_j}^2 + \sum_{i=0}^{N-1/2} \frac{1}{2}\frac{G_i}{\lambda_g} a_i^2 \tag{2.50}$$

As in the previous case, G_i/λ_g is very small and the last term in (2.50) can be neglected, and thus the energy function coincides with the error function given by (2.46).

The time derivative of the energy function is given by

$$\frac{dE}{dt} = \sum_{i=0}^{N-1/2} \left(\frac{\partial E}{\partial a_i}\right)\left(\frac{da_i}{dt}\right) \tag{2.51}$$

From (2.49), we get

$$\frac{\partial E}{\partial a_i} = \sum_{j=1}^{m} \frac{\partial F(v_{1_j})}{\partial a_i} + \sum_{j=1}^{m} \frac{\partial F(v_{2_j})}{\partial a_i} + G_i u_i \tag{2.52}$$

The first two terms on the right-hand side are given by

$$\begin{aligned} \frac{\partial F(v_{1_j})}{\partial a_i} &= -e_{1_j} \cos i\omega_j \\ \frac{\partial F(v_{2_j})}{\partial a_i} &= e_{2_j} \cos i\omega_j \end{aligned} \tag{2.53}$$

Substituting (2.52) and (2.53) in (2.51) and noting (2.48), we get

$$\frac{dE}{dt} = -C_g \sum_{i=0}^{N-1/2} \left(\frac{du_i}{dt}\right)\left(\frac{da_i}{dt}\right)$$

Finally substituting (2.48)

$$\frac{dE}{dt} = -\frac{C_g}{\lambda_g} \sum_{i=0}^{N-1/2} \left(\frac{da_i}{dt}\right)^2 \tag{2.54}$$

As can be seen, (2.54) is always less than or equal to zero.

The equation of motion of all g neurons can be written as

$$C_g \frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \mathbf{C}^T\mathbf{e}_1 - \mathbf{C}^T\mathbf{e}_2$$

where $\mathbf{G}$ is the diagonal matrix of the conductances. Substituting $\mathbf{e}_1$ and $\mathbf{e}_2$ and rearranging terms, we deduce

$$C_g \frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \lambda_f R_f \left[\mathbf{C}^T(\mathbf{d} - \mathbf{C}\mathbf{a} - \delta) - \mathbf{C}^T(-\mathbf{d} - \mathbf{C}\mathbf{a} - \delta)\right]$$

Manipulating the left-hand side, one gets

$$C_g \frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = 2\lambda_f R_f \left(\mathbf{C}^T\mathbf{d} - \mathbf{C}^T\mathbf{C}\mathbf{a}\right)$$

Noting the fact that $a_i = \lambda_g u_i$, we get

$$\frac{d\mathbf{a}}{dt} + \left[\frac{1}{C_g}\mathbf{G} + \frac{2\lambda_f\lambda_g R_f}{C_g}\mathbf{C}^T\mathbf{C}\right]\mathbf{a} = \frac{2\lambda_f\lambda_g R_f}{C_g}\mathbf{C}^T\mathbf{d}$$

As we can see, the system matrix is positive definite. So, the network settles to the global minimum.

As before, the optimal solution for the least-square error is given by

$$\mathbf{a}_{opt} = \left(\mathbf{C}^T\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{d}$$

and the solution obtained by the network is given by

$$\mathbf{a}_* = \left(\mathbf{C}^T\mathbf{C} + \frac{1}{2\lambda_f\lambda_g R_f}\mathbf{G}\right)^{-1}\mathbf{C}^T\mathbf{d}$$

Since the second term in the bracketed expression is very very small, the solution obtained by the network can be said to be optimal.

2.4.3 Results

The network in this section was simulated to design an FIR filter with passband edge $\omega_p = 0.3\pi$ and stopband edge $\omega_a = 0.5\pi$. The desired passband ripple (A_p) and minimum stopband attenuation (A_a) were 0.1 and 40 dB, respectively. The length of the filter was set to 27. The amplitude response of the filter is shown in Figure 2.13 with the passband ripple in the inset.

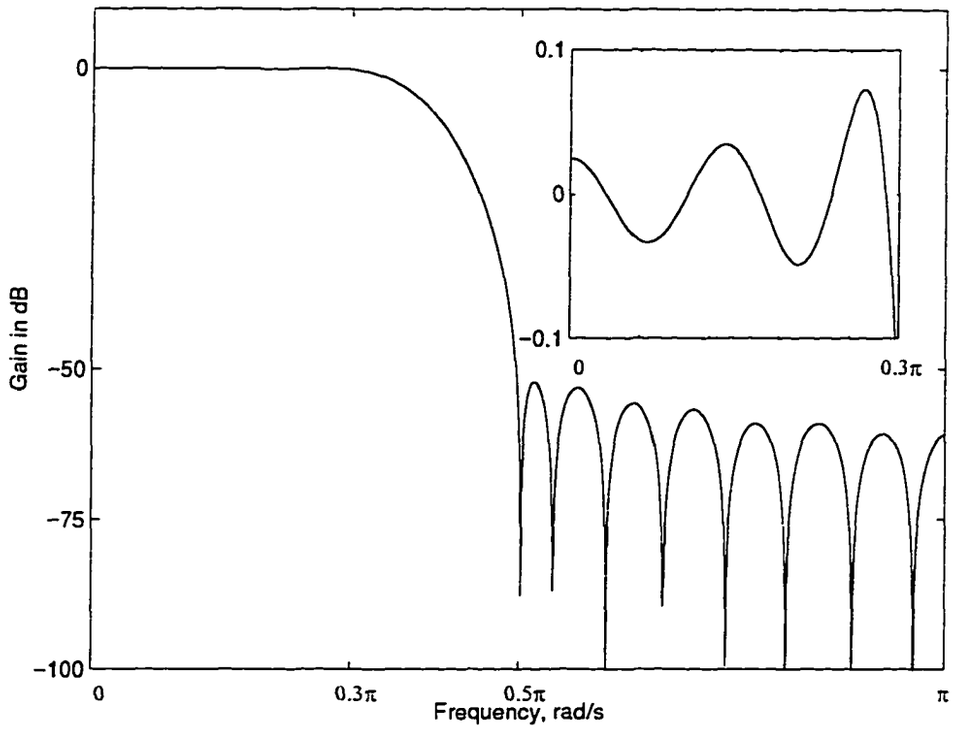


Figure 2.13: Amplitude response of the filter obtained with $\omega_p = 0.3\pi$, $\omega_a = 0.5\pi$, $A_p = 0.1$, and $A_a = 40$ dB. The passband ripple is shown in the inset.

2.5 Conclusions

Neural networks based on Hopfield's linear programming circuit for designing three different kinds of FIR filters have been proposed. A detailed mathematical analysis of the networks has been provided. Convergence analysis based on the energy function method has been included to show that each of the networks converges to the global minimum. Mathematical expressions have been provided only for filters with odd length, and can always be extended for filters with even length. The same network can also be used for designing highpass and multi-band filters as well as other FIR systems like differentiators and Hilbert transformers [13]. Results obtained by the proposed method have been compared with other existing methods and have been found to be comparable.

Chapter 3

2-D FIR Filters

3.1 Introduction

Two-dimensional signal processing is required for signals that are essentially two-dimensional. Images such as satellite photographs, radar and sonar maps, medical pictures etc. are typical 2-D signals which may need processing. Processing may be required either to improve the quality of the picture or to remove some undesired features.

Two-dimensional FIR filters are either designed by the window method or by optimization where a cost function is minimized with respect to the filter coefficients. Closed-form solutions are normally available for the window method and computionally it is much less intensive. However, the solutions obtained by this method are known to be suboptimal. The optimization approach, on the other hand, is computationally demanding but can provide optimal solutions. Yet another method is to apply appropriate transformations to 1-D transfer functions [46].

A computational method based on Hopfield's linear programming problem to solve the approximation problem for 1-D FIR filters had been discussed in chapter 2. In this chapter, it will be shown that a feedback neural network can be developed to solve the approximation problem for 2-D FIR filters [14]. In section 3.2, detailed design procedures for designing FIR filters on the basis of a given amplitude response is provided. In section 3.3, a procedure is developed for designing 2-D FIR filters for prescribed specifications. Both the procedures concentrate on designing filters with quadrantal symmetry.

3.2 Design of 2-D FIR filters based on a given amplitude response

3.2.1 Design procedure

The transfer function of a causal 2-D FIR filter is given by

$$H(z_1,\ z_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} h(n_1,\ n_2) z_1^{-n_1} z_2^{-n_2}$$

where the dimension of the filter is given by $(N_1 \times N_2)$. Its frequency response is given by

$$\begin{aligned} H(e^{j\omega_1}, e^{j\omega_2}) &= \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} h(n_1,\ n_2) e^{(jn_1\omega_1 + n_2\omega_2)} \\ &= M(\omega_1,\ \omega_2) e^{j\theta(\omega_1,\ \omega_2)} \end{aligned}$$

where

$$\begin{aligned} M(\omega_1,\ \omega_2) &= |H(e^{j\omega_1}, e^{j\omega_2})| \\ \theta(\omega_1, \omega_2) &= \arg H(e^{j\omega_1}, e^{j\omega_2}) \end{aligned}$$

are the amplitude and phase responses of the filter. For a linear phase FIR filter with symmetrical impulse response, it can be shown [46] that the impulse response values are related by

$$h(n_1, n_2) = h[(N_1 - 1 - n_1), (N_2 - 1 - n_2)]$$

For a filter with quadrantal symmetry, the frequency response is given by

$$H(e^{j\omega_1}, e^{j\omega_2}) = M(\omega_1,\ \omega_2) e^{j\left[\frac{(N_1-1)}{2}\omega_1 + \frac{(N_2-1)}{2}\omega_2\right]}$$

where

$$M(\omega_1,\ \omega_2) = |G(\omega_1, \omega_2)|$$

with

$$G(\omega_1,\ \omega_2) = \sum_{k_1=0}^{p_1}\sum_{k_2=0}^{p_2} a(k_1,\ k_2)\cos(k_1\omega_1)\cos(k_2\omega_2)$$

and $p_1 = (N_1 - 1)/2$ and $p_2 = (N_2 - 1)/2$.

Let the desired amplitude response be defined at $m_1 \times m_2$ grid points. The sum-squared error over all the frequency points is given by

$$E = \sum_{j=1}^{m_1} \sum_{k=1}^{m_2} [D(\omega_{1j}, \omega_{2k}) - G(\omega_{1j}, \omega_{2k})]^2 \tag{3.1}$$

where $D(\omega_{1j}, \omega_{2k})$ is the desired amplitude response at frequency $(\omega_{1j}, \omega_{2k})$. A network that can minimize (3.1) is shown in Figure 3.1. There are $(p_1 + 1) \times (p_2 + 1)$ g neurons and $m_1 \times m_2$ f neurons.

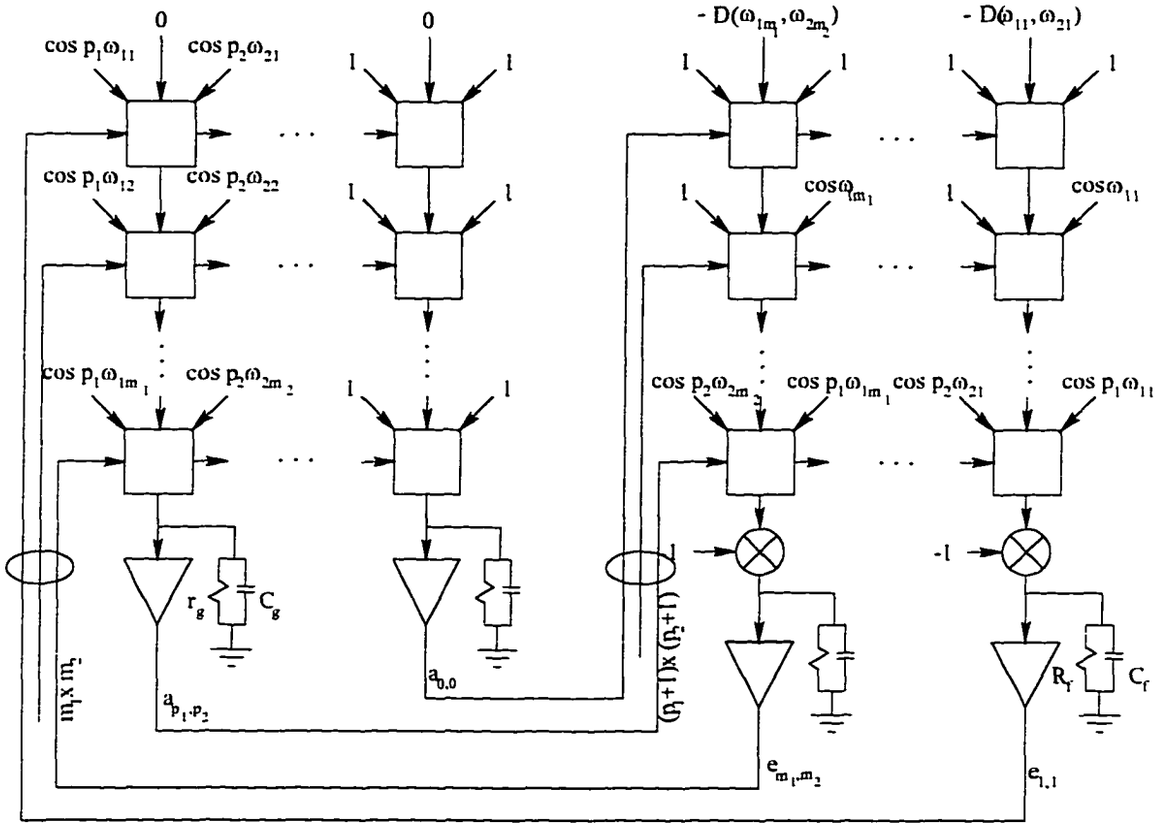


Figure 3.1: Network for the design of 2-D FIR filters.

Neglecting the dynamics of the f neurons, the input voltage of an f neuron is given by

$$\begin{aligned} v_{j,k} &= D(\omega_{1j}, \omega_{2k}) - \sum_{k_1=0}^{p_1} \sum_{k_2=0}^{p_2} a(k_1, k_2) \cos(k_1\omega_{1j}) \cos(k_2\omega_{2k}) \\ &= D(\omega_{1j}, \omega_{2k}) - \mathbf{c_1}(\omega_{1j})^T \mathbf{A} \mathbf{c_2}(\omega_{2k}) \end{aligned} \tag{3.2}$$

for $j = 1, 2, \cdots, m_1$, $k = 1, 2, \cdots, m_2$ where

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,p_2} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,p_2} \\ \vdots & \vdots & \cdots & \vdots \\ a_{p_1,0} & a_{p_1,1} & \cdots & a_{p_1,p_2} \end{bmatrix}$$

is the matrix formed by the filter coefficients, and $\mathbf{c}(\omega) = [1 \ \cos\omega \ \cos 2\omega \ \cdots]^T$. The input-output relationship of the f neuron is

$$e_{j,k} = f(v_{j,k}) = \lambda_f R_f v_{j,k} \tag{3.3}$$

where λ_f is the gain and R_f is the input resistance of an f neuron. The output voltage of all f neurons can be written as

$$\mathbf{E} = \lambda_f R_f \left[\mathbf{D} - \mathbf{C_1 A C_2}^T\right] \tag{3.4}$$

where $\mathbf{C_1}$ and $\mathbf{C_2}$ are $m_1 \times (p_1 + 1)$ and $m_2 \times (p_2 + 1)$ cosine matrices, $\mathbf{D}$ is an $m_1 \times m_2$ desired response matrix, and $\mathbf{A}$ is the $(p_1 + 1) \times (p_2 + 1)$ filter-coefficient matrix.

The equation of motion of a g neuron is given by

$$C_g \frac{du_{p,q}}{dt} + \frac{u_{p,q}}{r_g} = \sum_{j=1}^{m_1} \sum_{k=1}^{m_2} (e_{j,k} - u_{p,q}) \cos(p\omega_{1j}) \cos(q\omega_{2k}) \tag{3.5}$$

where $u_{p,q}$ is the input voltage of the (p,q)th g neuron and $p = 1, 2, \ldots, p_1$, $q = 1, 2, \ldots, p_2$. The input capacitance of a g neuron is denoted by C_g and the input resistance by r_g. After rearranging terms in (3.5), we get

$$C_g \frac{du_{p,q}}{dt} + G_{p,q} u_{p,q} = \sum_{j=1}^{m_1} \sum_{k=1}^{m_2} e_{j,k} \cos(p\omega_{1j}) \cos(q\omega_{2k}) \tag{3.6}$$

where $G_{p,q}$ is the total input conductance of the (p, q)th neuron and is given by

$$G_{p,q} = \frac{1}{r_g} + \sum_{j=1}^{m_1} \sum_{k=1}^{m_2} \cos(p\omega_{1j}) \cos(q\omega_{2k})$$

The activities of all the g neurons can be written compactly as

$$C_g \frac{d\mathbf{U}}{dt} + \mathbf{G}.\mathbf{U} = \mathbf{C_1}^T \mathbf{E C_2} \tag{3.7}$$

where $\mathbf{G}$ is a $p_1 \times p_2$ matrix whose elements are the input conductances of the g neurons. The quantity $\mathbf{G}.\mathbf{U}$ represents term by term multiplication. The input-output relationship of the g neuron is linear and is given by

$$a_{p,q} = g(u_{p,q}) = \lambda_g u_{p,q} \tag{3.8}$$

3.2.2 Convergence

The energy function of the network is given by

$$E = \sum_{j=1}^{m_1}\sum_{k=1}^{m_2} F(v_{j,k}) + \sum_{p=0}^{p_1}\sum_{q=0}^{p_2} G_{p,q}\int_0^{a_{p,q}} g^{-1}(a_{p,q})da_{p,q} \tag{3.9}$$

where $v_{j,k}$ is defined in (3.2). Function F is related to the characteristic function of the f neuron by

$$\frac{dF(x)}{dx} = f(x) \tag{3.10}$$

Substituting (3.2), (3.3), and (3.10) in (3.9), the energy function can be rewritten as

$$\begin{aligned} E = \ & \frac{1}{2}\ \lambda_f R_f \sum_{j=1}^{m_1}\sum_{k=1}^{m_2}\left[D(\omega_{1j},\omega_{2k}) - \sum_{k_1=0}^{p_1}\sum_{k_2=0}^{p_2} a(k_1,k_2)\cos k_1\omega_{1j}\cos k_2\omega_{2k}\right]^2 \\ & + \sum_{p=0}^{p_1}\sum_{q=0}^{p_2}\frac{1}{2\lambda_g}G_{p,q}a_{p,q}^2 \end{aligned} \tag{3.11}$$

The second term can be neglected because $G_{p,q}/\lambda_g$ is very small and the energy function coincides with the error function. It can now be shown that the energy function of this network is minimized until the network settles to one of the solutions. From (3.11) it is apparent that eventually the cost function of the design problem also gets minimized.

In order to check the time derivative of the energy function, we note that

$$\frac{dE}{dt} = \sum_{p=0}^{p_1}\sum_{q=0}^{p_2}\left(\frac{\partial E}{\partial a_{p,q}}\right)\left(\frac{da_{p,q}}{dt}\right) \tag{3.12}$$

Now

$$\frac{\partial E}{\partial a_{p,q}} = \sum_{j=1}^{m_1}\sum_{k=1}^{m_2}\frac{\partial F(v_{j,k})}{\partial a_{p,q}} + G_{p,q}u_{p,q} \tag{3.13}$$

The first term in (3.13) is given by

$$\frac{\partial F(v_{j,k})}{\partial a_{p,q}} = \frac{\partial F(v_{j,k})}{\partial v_{j,k}}\frac{dv_{j,k}}{da_{p,q}} \tag{3.14}$$

After using the relations in (3.10) and (3.2), (3.14) can be written as

$$\frac{\partial F(v_{j,k})}{\partial a_{p,q}} = -f(v_{j,k})\cos(k_1\omega_{1j})\cos(k_2\omega_{2k}) \tag{3.15}$$

Substituting (3.15) in (3.13), we get

$$\begin{aligned} \frac{\partial E}{\partial a_{p,q}} &= \sum_{j=1}^{m_1}\sum_{k=1}^{m_2} -e_{j,k}\cos(k_1\omega_{1j})\cos(k_2\omega_{2k}) + G_{p,q}u_{p,q} \\ &= -C_g\frac{du_{p,q}}{dt} \end{aligned}$$

Finally from (3.12), we deduce

$$\frac{dE}{dt} = \sum_{p=0}^{p_1}\sum_{q=0}^{p_2} -C_g \frac{du_{p,q}}{dt}\frac{da_{p,q}}{dt}$$
$$= -\frac{C_g}{\lambda_g}\sum_{p=0}^{p_1}\sum_{q=0}^{p_2}\left(\frac{da_{p,q}}{dt}\right)^2$$

where we have used (3.8). The above expression is either negative or zero at the stationary point. This means that the network minimizes the energy function and settles to one of the minima of the energy function.

3.2.3 The network

The network consists of two sets of neurons: $(m_1 \times m_2)$ f neurons and $(p_1 + 1) \times (p_2 + 1)$ g neurons. It uses identical connection elements for both types of neurons which are represented in Figure 3.1 as square boxes. The conductances for each connection point are fed from the left and right corners. The input voltage and input current are fed from the left-hand side and the top, respectively. Each element multiplies the input voltage by the conductance, adds with the input current to produce the output current which is used by the block immediately below it. Since each column on the right-hand side computes the negative of the error value, the output current is negated before being fed to the f neurons.

Initially the outputs of all the neurons are set to zero. f neurons are updated first and the updated values are used by the g neurons to update their states. This iterative process is continued until the desired accuracy is achieved. To achieve the stopping criterion, the outputs of the g neurons are compared with the values obtained from the previous iteration. If the difference between two successive values for each filter coefficient is less than a small quantity (say 10^{-6}), the iteration is terminated and the output of the g neurons are the desired filter coefficients.

3.2.4 Results

The network in Figure 3.1 was simulated to design a squared-symmetric lowpass FIR filter with the following specifications: $\omega_{p_{1,2}} = 1$, $\omega_{a_{1,2}} = 2$ and $\omega_{s_{1,2}} = 2\pi$ rad/s. The dimension of the filter was chosen to be 13×13. The network was simulated until the convergence criterion was met. The amplitude response of the filter obtained is shown in Figure 3.2.

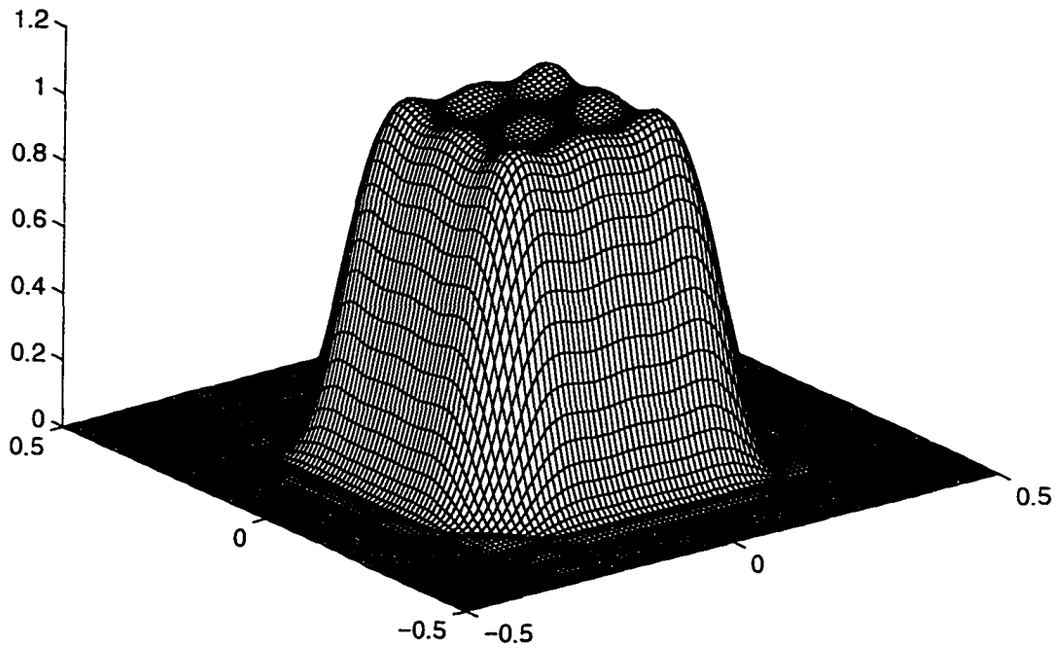


Figure 3.2: Amplitude response of lowpass filter with $\omega_{p_{1,2}} = 1$, $\omega_{a_{1,2}} = 2$ rad/s obtained by the proposed method. The dimension of the filter is 13×13.

3.3 Design of 2-D FIR filters satisfying prescribed specifications

In order to design a filter with a given passband error δ_p and stopband error δ_a, the error must satisfy the following relations

$$\begin{aligned} |E(\omega_{1j}, \omega_{2k})| &\leq \delta_p \quad \text{in passband} \\ &\geq \delta_a \quad \text{in stopband} \end{aligned}$$

where

$$E(\omega_{1j}, \omega_{2k}) = D(\omega_{1j}, \omega_{2k}) - G(\omega_{1j}, \omega_{2k})$$

For a quadrantal symmetric filter, the amplitude response is given by

$$G(\omega_{1j}, \omega_{2k}) = \sum_{k_1=0}^{p_1} \sum_{k_2=0}^{p_2} a(k_1, k_2) \cos(k_1 \omega_{1j}) \cos(k_2 \omega_{2k})$$

For a circular symmetric filter, the symmetry is octagonal. However, the same expression for quadrantal symmetric filter would be used in the subsequent deductions. However, the sample points would be taken on equidistant circular arcs in the first quadrant. Hence, the sample points can be put in a matrix of size $m \times 2$, where m is the number of sample points. Now, the sum-squared error can be written as

$$E = \sum_{j=1}^{m} [D(\omega_{j1}, \omega_{j2}) - G(\omega_{j1}, \omega_{j2}) - \delta]^2 + \sum_{j=1}^{m} [-D(\omega_{j1}, \omega_{j2}) + G(\omega_{j1}, \omega_{j2}) - \delta]^2 \quad (3.16)$$

where δ is equal to δ_p in the passband and δ_a in the stopband. Since the least-square error has two terms, we will have two sets of f neurons, one for each term.

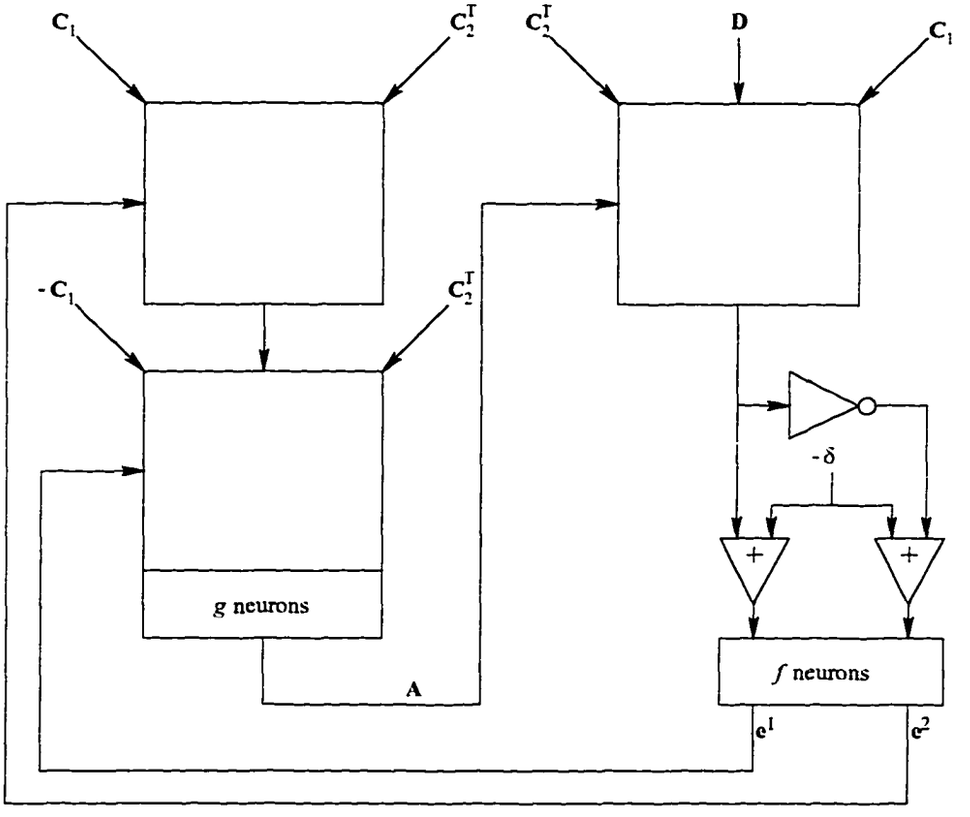


Figure 3.3: Network for the design of 2-D FIR filters satisfying prescribed specifications.

The proposed scheme is shown in Figure 3.3. The empty squares represent the connectivities shown in Figure 3.1. The input voltages of the jth f neurons are given by

$$v_j^1 = D(\omega_{j1}, \omega_{j2}) - \sum_{k_1=0}^{p_1} \sum_{k_2=0}^{p_2} a(k_1, k_2) \cos(k_1\omega_{j1}) \cos(k_2\omega_{j2}) - \delta(\omega_{j1}, \omega_{j2})$$

$$v_j^2 = -D(\omega_{j1}, \omega_{j2}) + \sum_{k_1=0}^{p_1} \sum_{k_2=0}^{p_2} a(k_1, k_2) \cos(k_1\omega_{j1}) \cos(k_2\omega_{j2}) - \delta(\omega_{j1}, \omega_{j2}) \quad (3.17)$$

The input-output relationship is given by

$$e_j^{1,2} = f(v_j^{1,2}) = \lambda_f R_f v_j^{1,2} \tag{3.18}$$

The outputs of all the f neurons can be written as

$$\mathbf{e}^1 = \lambda_f R_f \left[\mathbf{D} - \text{diag}\left(\mathbf{C}_1\mathbf{A}\mathbf{C}_2^T\right) - \boldsymbol{\delta}\right]$$
$$\mathbf{e}^2 = \lambda_f R_f \left[-\mathbf{D} + \text{diag}\left(\mathbf{C}_1\mathbf{A}\mathbf{C}_2^T\right) - \boldsymbol{\delta}\right]$$

where $\boldsymbol{\delta}$ is the array of passband and stopband errors.

The equation of motion of the (p,q)th g neuron is given by

$$C_g \frac{du_{p,q}}{dt} + \frac{u_{p,q}}{r_g} = \sum_{j=1}^{m} \left(e_j^1 - u_{p,q}\right) \cos(p\omega_{j1}) \cos(q\omega_{j2}) - \sum_{j=1}^{m} \left(e_j^2 + u_{p,q}\right) \cos(p\omega_{j1}) \cos(q\omega_{j2})$$

The change of sign in the second expression is required to make the time derivative of the energy function negative. After rearranging terms, we get

$$C_g \frac{du_{p,q}}{dt} + G_{p,q} u_{p,q} = \sum_{j=1}^{m} e_j^1 \cos(p\omega_{j1}) \cos(q\omega_{j2}) - \sum_{j=1}^{m} e_j^2 \cos(p\omega_{j1}) \cos(q\omega_{j2})$$

where $G_{p,q}$ is the (p,q)th input conductance and is given by

$$G_{p,q} = \frac{1}{r_g} + 2\sum_{j=1}^{m} \cos(p\omega_{j1}) \cos(q\omega_{j2})$$

The input-output relationship of the g neuron is given by

$$a_{p,q} = g(u_{p,q}) = \lambda_g u_{p,q} \tag{3.19}$$

3.3.1 Convergence

The energy function of the network can be written as

$$E = \sum_{j=1}^{m} F(v_j^1) + \sum_{j=1}^{m} F(v_j^2) + \sum_{p=0}^{p_1} \sum_{q=0}^{p_2} G_{p,q} \int_0^{a_{p,q}} g^{-1}(a_{p,q} da_{p,q} \tag{3.20}$$

Expanding the terms and carrying out the integration, we get

$$E = \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m} \left[(e_j^1)^2 + (e_j^2)^2\right] + \sum_{p=0}^{p_1} \sum_{q=0}^{p_2} \frac{1}{2\lambda_g} G_{p,q} a_{p,q}^2 \tag{3.21}$$

Substituting (3.2) and (3.3) in (3.21), we see that for a high λ_g/G ratio, the second term can be neglected and the energy function coincides with the least-square error to within a constant term.

The time derivative of the energy function is give by

$$\frac{dE}{dt} = \sum_{p=0}^{p_1} \sum_{q=0}^{p_2} \left(\frac{\partial E}{\partial a_{p,q}} \right) \left(\frac{da_{p,q}}{dt} \right) \tag{3.22}$$

From (3.20),

$$\begin{aligned} \frac{\partial E}{\partial a_{p,q}} &= \sum_{j=1}^{m} \frac{\partial F(v_j^1)}{\partial a_{p,q}} + \sum_{j=1}^{m} \frac{\partial F(v_j^2)}{\partial a_{p,q}} + G_{p,q} u_{p,q} \\ &= -e_j^1 \cos(p\omega_{ij}) \cos(q\omega_{2j}) + e_j^2 \cos(p\omega_{ij}) \cos(q\omega_{2j}) + G_{p,q} u_{p,q} \\ &= -C_g \frac{du_{p,q}}{dt} \end{aligned} \tag{3.23}$$

Finally substituting (3.23) and (3.19) in (3.22), we get

$$\frac{dE}{dt} = -\frac{C_g}{\lambda_g} \sum_{p=0}^{p_1} \sum_{q=0}^{p_2} \left(\frac{da_{p,q}}{dt} \right)^2$$

As before, the above expression is always less than or equal to zero. This means the network always settles to one of the solutions.

3.3.2 Results

The network was simulated to design a circular symmetric lowpass filter with passband edge $\omega_p = 0.3\pi$, stopband edge $\omega_a = 0.5\pi$, and sampling frequency $\omega_s = 2\pi$. The filter dimension was chosen to be 25 × 25. The maximum allowable passband and stopband errors were $\delta_p = 8.610^{-3}$ and $\delta_a = 10^{-2}$. The circuit was simulated with reference points along equidistant circular arcs in the first quadrant. Since the filter designed has octagonal symmetry, the coefficients obtained have perfect symmetry. The amplitude response of the filter is shown in Figure 3.4. The maximum errors in the passband and stopband were 7.3×10^{-3} and 3.4×10^{-3}.

3.3.3 Conclusions

A feedback neural network was proposed for designing 2-D FIR filters. A detailed mathematical analysis has been included along with the convergence properties. The proposed method can easily be extended to designing highpass as well as multiband filters. Though filters with quadrantal symmetry have been considered here, it is relatively simple to extend the work for designing 2-D filters with any kind of symmetries. It can also be used for designing filters with arbitrary amplitude and phase response. The network uses identical connection elements and neurons and can easily be implemented in analog VLSI.

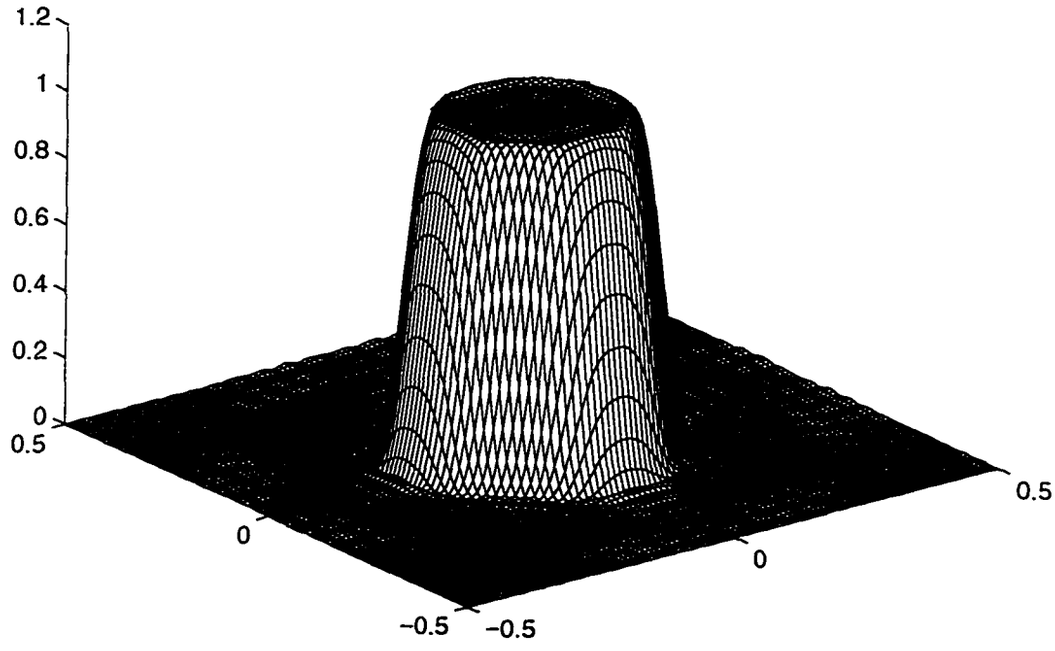


Figure 3.4: Amplitude response of the lowpass filter with $\omega_p = 0.3\pi$, $\omega_a = 0.5\pi$. The passband and stopband errors are $\delta_p = 8.6\times10^{-3}$, $\delta_a = 10^{-2}$, respectively, and the dimension of the filter is 25×25.

Chapter 4

1-D IIR Filters

4.1 Introduction

A large number of techniques are available for designing digital IIR filters, the most common being by applying transformations to an analog filter [5]. An analog filter is first designed and then the equivalent digital filter is obtained by a suitable transformation like the bilinear transformation. These techniques are suitable for designing standard filters like lowpass, highpass, bandpass, and bandstop filters with piecewise constant amplitude responses. In order to design IIR filters with arbitrary amplitude responses, one can use various optimization techniques. Though convergence is not guaranteed and the final solution depends heavily on the initial conditions, these are by far the most useful design tools available today. Yet another method for designing IIR filter is to design in magnitude-squared domain where the frequency response of the filter is essentially a ratio of trigonometric polynomials [58]. The design of equiripple IIR filters in magnitude-squared domain have been considered in [58], [3] and [19].

In this chapter, a feedback neural network is proposed for designing IIR filters in the magnitude-squared domain. The method can design filters with arbitrary amplitude responses as shown in section 4.2. The design of IIR filters satisfying given amplitude and phase responses can be carried out by another neural network based on Hopfield's linear programming network as will be shown in section 4.3 [9]. The chapter is concerned with the mathematical analysis of the two methods and provides examples to illustrate their applications.

4.2 Design of IIR filters in magnitude-squared domain

4.2.1 Theory

The transfer function of an IIR filter is given by

$$H(z) = \frac{N(z)}{D(z)} = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{i=1}^{N} a_i z^{-i}} \tag{4.1}$$

where the numerator polynomial $N(z)$ is of degree M and the denominator polynomial $D(z)$ is of degree N. The function in the magnitude-squared domain is given by

$$\begin{aligned} H(z)H(z^{-1}) &= \frac{N(z)N(z^{-1})}{D(z)D(z^{-1})} \\ &= \frac{\sum_{i=-M}^{M} c_i z^{-i}}{1 + \sum_{i=-N}^{N} d_i z^{-i}} \end{aligned} \tag{4.2}$$

where

$$c_i = c_{-i} \text{ for } i = 1, 2, \ldots, M$$

$$d_i = d_{-i} \text{ for } i = 1, 2, \ldots, N$$

The magnitude-squared function is obtained by evaluating (4.2) on the unit circle, i.e.,

$$\begin{aligned} H^*(\omega) &= H(z)H(z^{-1})|_{z=exp(-j\omega)} \\ &= \frac{c_0 + \sum_{i=1}^{M} 2c_i \cos(i\omega)}{1 + \sum_{i=1}^{N} 2d_i \cos(i\omega)} \\ &= \frac{\hat{N}(\omega)}{1 + \hat{D}(\omega)} \end{aligned} \tag{4.3}$$

The magnitude-squared function, as seen in (4.3), is a ratio of trigonometric polynomials. It is also seen that both the numerator and denominator polynomials are linear in the unknown filter coefficients $\{c_i\}$ and $\{d_i\}$.

Let $D^*(\omega)$ be the desired magnitude-squared characteristic. The approximation problem now can be stated as

$$-\epsilon(\omega) \leq D^*(\omega) - \frac{\hat{N}(\omega)}{1 + \hat{D}(\omega)} \leq \epsilon(\omega) \tag{4.4}$$

where $\epsilon(\omega)$ is the allowable error. Expanding (4.4), we get a set of two linear equations

$$\begin{aligned} &[D^*(\omega) - \epsilon(\omega)] - \hat{N}(\omega) + \hat{D}(\omega)\,[D^*(\omega) - \epsilon(\omega)] \leq 0 \\ &\hat{N}(\omega) - \hat{D}(\omega)\,[D^*(\omega) + \epsilon(\omega)] - [D^*(\omega) + \epsilon(\omega)] \leq 0 \end{aligned} \tag{4.5}$$

The left-hand side of (4.5) can be minimized and the corresponding errors can be written in vector form as

$$\tilde{\mathbf{E}}_1 = \tilde{\mathbf{H}}_1 - \left(\tilde{\mathbf{U}}\mathbf{c} - \tilde{\mathbf{U}}_1\mathbf{d}\right)$$
$$\tilde{\mathbf{E}}_2 = \left(\tilde{\mathbf{U}}\mathbf{c} - \tilde{\mathbf{U}}_2\mathbf{d}\right) - \tilde{\mathbf{H}}_2$$

where

$$\tilde{\mathbf{E}}_i = [E_i(\omega_1),\ E_i(\omega_2),\ \cdots]^T \quad \text{for i =1, 2}$$
$$\tilde{\mathbf{H}}_i = [H_i(\omega_1),\ H_i(\omega_2),\ \cdots]^T \quad \text{for i =1, 2}$$
$$H_1(\omega_j) = D^*(\omega_j) - \epsilon(\omega_j)$$
$$H_2(\omega_j) = D^*(\omega_j) + \epsilon(\omega_j)$$
$$\tilde{\mathbf{U}} = \begin{bmatrix} 1 & 2\cos\omega_1 & \cdots & 2\cos M\omega_1 \\ 1 & 2\cos\omega_2 & \cdots & 2\cos M\omega_2 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$
$$\tilde{\mathbf{U}}_i = \begin{bmatrix} -2H_i(\omega_1)\cos\omega_1 & \cdots & -2H_i(\omega_1)\cos N\omega_1 \\ -2H_i(\omega_2)\cos\omega_2 & \cdots & -2H_i(\omega_2)\cos N\omega_2 \\ \vdots & \vdots & \vdots \end{bmatrix} \quad \text{for i =1, 2}$$
$$\mathbf{c} = [c_0\ c_1\ \cdots\ c_M\ ]^T$$
$$\mathbf{d} = [d_1\ d_2\ \cdots\ d_N\ ]^T$$

The least-square error function can now be written as

$$E = \sum_{j=1}^{m}\left[H_1(\omega_j) - \left(c_0 + 2\sum_{i=1}^{M} c_i\cos i\omega_j - 2H_1(\omega_j)\sum_{i=1}^{N} d_i\cos i\omega_j\right)\right]^2 + \sum_{j=1}^{m}\left[\left(c_0 + 2\sum_{i=1}^{M} c_i\cos i\omega_j - 2H_2(\omega_j)\sum_{i=1}^{N} d_i\cos i\omega_j\right) - H_2(\omega_j)\right]^2 \tag{4.6}$$

where m is the number of sample points.

A network that can minimize the above error function is shown in Figure 4.1. There are two sets of f neurons to minimize (4.6) and two sets of g neurons corresponding to two sets of coefficients. The outputs of g neurons are fed back to the connection elements for the f neurons and, similarly, the outputs of f neurons are connected to the connection elements for the g neurons. Both f and g neurons are constant-gain linear neurons with capacitor-resistor pairs at their inputs. The time constant of the f neurons are chosen to be much less than that of the g neurons and hence the f neurons are considered non-dynamical.

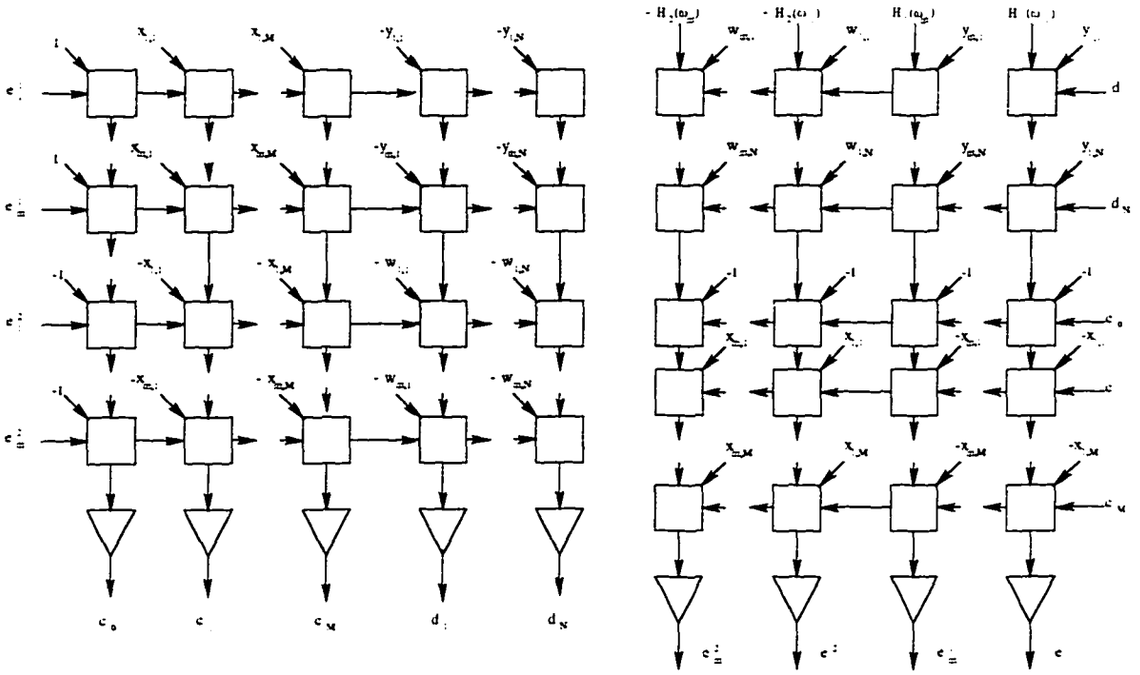


Figure 4.1: Network for the design of 1-D IIR filters. The input resistor-capacitor blocks are not shown: $w_{j,i} = -2H_2(\omega_j)\cos i\omega_j$, $x_{j,i} = 2\cos i\omega_j$ and $y_{i,j} = 2H_1(\omega_j)\cos i\omega_j$.

The input voltages of f neurons are given by

$$v_j^1 = [D^*(\omega_j) - \epsilon(\omega_j)](1 + \sum_{i=1}^{N} d_i \cos i\omega_j) - (c_0 + 2\sum_{i=1}^{M} c_i \cos i\omega_j)$$

$$v_j^2 = -[D^*(\omega_j) + \epsilon(\omega_j)](1 + 2\sum_{i=1}^{N} d_i \cos i\omega_j) + (c_0 + 2\sum_{i=1}^{M} c_i \cos i\omega_j)$$

for $j = 1, 2, \cdots, m$. The output voltages are given by

$$\begin{aligned} e_j^1 &= f(v_j^1) = \lambda_f R_f v_j^1 \\ e_j^2 &= f(v_j^2) = \lambda_f R_f v_j^2 \end{aligned} \tag{4.7}$$

In vector form, the output voltages can be written as

$$\begin{aligned} \mathbf{e}^1 &= \lambda_f R_f \left[\tilde{\mathbf{H}}_1 - (\tilde{\mathbf{U}}\mathbf{c} - \tilde{\mathbf{U}}_1\mathbf{d})\right] = \lambda_f R_f \left[\tilde{\mathbf{H}}_1 - [\tilde{\mathbf{U}} \;\; -\tilde{\mathbf{U}}_1]\mathbf{x}\right] \\ \mathbf{e}^2 &= \lambda_f R_f \left[(\tilde{\mathbf{U}}\mathbf{c} - \tilde{\mathbf{U}}_2\mathbf{d}) - \tilde{\mathbf{H}}_2\right] = \lambda_f R_f \left[[\tilde{\mathbf{U}} \;\; -\tilde{\mathbf{U}}_2]\mathbf{x} - \tilde{\mathbf{H}}_2\right] \end{aligned} \tag{4.8}$$

where $\mathbf{x} = [c_0\ c_1\ \cdots\ c_M\ d_1\ d_2\ \cdots\ d_N]^T$.

The equation of motion for the g neuron representing the c coefficients is given by

$$C\frac{du_i^c}{dt} + \frac{u_i^c}{r_g} = \sum_{j=1}^{m}(e_j^1 - u_i^c)K_{ij} - \sum_{j=1}^{m}(e_j^2 + u_i^c)K_{ij}$$

$$C\frac{du_i^c}{dt} + G_i^c = \sum_{j=1}^{m}(e_j^1 - e_j^2)K_{ij} \tag{4.9}$$

for $i = 0, 1, \ldots, M$, where

$$K_{ij} = \begin{cases} 1 & \text{for } i=0 \\ 2\cos i\omega_j & \text{otherwise} \end{cases}$$

and

$$G_i^c = \frac{1}{r_g} + 2\sum_{j=1}^{m} K_{ij}$$

In vector form, the equation of motion can be written as

$$C\frac{d\mathbf{u}^c}{dt} + \mathbf{G}^c\mathbf{u}^c = \tilde{\mathbf{U}}^T(\mathbf{e}^1 - \mathbf{e}^2)$$

where $\mathbf{G}^c$ is a diagonal matrix with G_i^c's as the entries. Similarly, the equation for the other g neurons representing the d coefficients can be written as

$$C\frac{du_i^d}{dt} + \frac{u_i^d}{r_g} = -\sum_{j=1}^{m}(e_j^1 + u_i^d)2\,[D^*(\omega_j) - \epsilon(\omega_j)]\cos i\omega_j + \sum_{j=1}^{m}(e_j^2 - u_i^d)2\,[D^*(\omega_j) + \epsilon(\omega_j)]\cos i\omega_j$$

for $i = 1, 2, \ldots, N$. In other words,

$$C\frac{du_i^d}{dt} + G_i^d u_i^d = -\sum_{j=1}^{m} 2e_j^1\,[D^*(\omega_j) - \epsilon(\omega_j)]\cos i\omega_j + \sum_{j=1}^{m} 2e_j^2\,[D^*(\omega_j) + \epsilon(\omega_j)]\cos i\omega_j \tag{4.10}$$

where

$$G_i^d = \frac{1}{r_g} + \sum_{j=1}^{m} 2\,[D^*(\omega_j) - \epsilon(\omega_j)]\cos i\omega_j + \sum_{j=1}^{m} 2\,[D^*(\omega_j) + \epsilon(\omega_j)]\cos i\omega_j$$

In vector form, the equation of motion of the d neurons can be written as

$$C\frac{d\mathbf{u}^d}{dt} + \mathbf{G}^d\mathbf{u}^d = -\tilde{\mathbf{U}}_1^T\mathbf{e}^1 + \tilde{\mathbf{U}}_2^T\mathbf{e}^2$$

The input-output relationship for the g neurons can be written as

$$c_i = g(u_i^c) = \lambda_g u_i^c$$

$$d_i = g(u_i^d) = \lambda_g u_i^d \tag{4.11}$$

Once the network settles, the coefficients are used in the polynomials in z in (4.2) and factored. Poles and zeros that are within the unit circles are retained. Zeros on the unit circle appear in pairs and only one pair of each is retained and finally the transfer function is constructed [58].

4.2.2 Convergence

The energy function of the network can be written as

$$E = \sum_{j=1}^{m} F(v_j^1) + \sum_{j=1}^{m} F(v_j^2) + \sum_{i=0}^{M} G_i^c \int_0^{c_i} g^{-1}(c_i)dc_i + \sum_{i=1}^{N} G_i^d \int_0^{d_i} g^{-1}(d_i)dd_i \tag{4.12}$$

where F is related to the characteristic function f of the f neurons by

$$\frac{dF(z)}{dz} = f(z) \tag{4.13}$$

z being a dummy variable. Noting (4.13) and carrying out the integration, (4.12) can be rewritten as

$$E = \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m} \left[(v_j^1)^2 + (v_j^2)^2\right] + \frac{1}{2}\sum_{i=0}^{m} \frac{G_i^c}{\lambda_g} c_i^2 + \frac{1}{2}\sum_{i=1}^{N} \frac{G_i^d}{\lambda_g} d_i^2$$

For a high λ_g/G_i ratio, the last two terms can be neglected and the energy function coincides with the error function to within a constant factor.

In order to check the time derivative of the energy function, we note that

$$\frac{dE}{dt} = \sum_{i=0}^{M} \frac{\partial E}{\partial c_i}\frac{dc_i}{dt} + \sum_{i=1}^{N} \frac{\partial E}{\partial d_i}\frac{dd_i}{dt} \tag{4.14}$$

Now

$$\begin{aligned}
\frac{\partial E}{\partial c_i} &= \sum_{j=1}^{m} \frac{\partial F(v_j^1)}{\partial c_i} + \sum_{j=1}^{m} \frac{\partial F(v_j^2)}{\partial c_i} + G_i^c c_i \\
\frac{\partial E}{\partial d_i} &= \sum_{j=1}^{m} \frac{\partial F(v_j^1)}{\partial d_i} + \sum_{j=1}^{m} \frac{\partial F(v_j^2)}{\partial d_i} + G_i^d d_i
\end{aligned} \tag{4.15}$$

The partial derivatives on the right-hand hand side of (4.15) can be evaluated as

$$\begin{aligned}
\frac{\partial F(v_j^1)}{\partial c_i} &= \frac{\partial F(v_j^1)}{\partial v_j^1}\frac{dv_j^1}{dc_i} = -f(v_j^1)K_{ij} \\
\frac{\partial F(v_j^2)}{\partial c_i} &= \frac{\partial F(v_j^2)}{\partial v_j^2}\frac{dv_j^2}{dc_i} = f(v_j^2)K_{ij} \\
\frac{\partial F(v_j^1)}{\partial d_i} &= \frac{\partial F(v_j^1)}{\partial v_j^1}\frac{dv_j^1}{dd_i} = 2f(v_j^1)\left[D^*(\omega_j) - \epsilon(\omega_j)\right]\cos i\omega_j \\
\frac{\partial F(v_j^2)}{\partial d_i} &= \frac{\partial F(v_j^2)}{\partial v_j^2}\frac{dv_j^2}{dd_i} = -2f(v_j^2)\left[D^*(\omega_j) - \epsilon(\omega_j)\right]\cos i\omega_j
\end{aligned} \tag{4.16}$$

After substituting (4.15) - (4.16) in (4.14), the time derivative of the energy function can be written as

$$\begin{aligned}\frac{dE}{dt} &= -\sum_{i=0}^{M} C\frac{du_i^c}{dt}\frac{dc_i}{dt} - \sum_{i=1}^{N} C\frac{du_i^d}{dt}\frac{dd_i}{dt} \\ &= -\frac{C}{\lambda_g}\sum_{i=0}^{M}\left(\frac{dc_i}{dt}\right)^2 - \frac{C}{\lambda_g}\sum_{i=1}^{N}\left(\frac{dd_i}{dt}\right)^2 \end{aligned} \tag{4.17}$$

Here we have used (4.9)-(4.11) for brevity. From (4.17) we see that the time derivative of the energy function is always less than or equal to zero and the network always settles to one of the minima.

In order to check the uniqueness of the solution in the magnitude-squared domain, the system equation can be written as

$$C\frac{\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = [\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_1]^T\mathbf{e}^1 - [\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_2]^T\mathbf{e}^2 \tag{4.18}$$

where $\mathbf{u} = [\mathbf{u}^{c^T} \quad \mathbf{u}^{d^T}]^T$ and $\mathbf{G}$ is a diagonal matrix with the conductances as its erntries. Substituting (4.8) in (4.18), we get

$$\begin{aligned} C\frac{\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \lambda_f R_f \Big[&[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_1]^T\tilde{\mathbf{H}}_1 - [\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_1]^T[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_1]\mathbf{x} \\ &- [\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_2]^T[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_2]\mathbf{x} + [\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_2]\tilde{\mathbf{H}}_2\Big] \end{aligned} \tag{4.19}$$

After some manipulation, (4.19) can be written as

$$\begin{aligned} \frac{d\mathbf{x}}{dt} + \left[\frac{1}{C}\mathbf{G} + \frac{\lambda_f\lambda_g R_f}{C}[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_1]^T[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_1] + \frac{\lambda_f\lambda_g R_f}{C}[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_2]^T[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_2]\right]\mathbf{x} \\ = \frac{\lambda_f\lambda_g R_f}{C}[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_1]^T\tilde{\mathbf{H}}_1 + \frac{\lambda_f\lambda_g R_f}{C}[\tilde{\mathbf{U}} \quad -\tilde{\mathbf{U}}_2]^T\tilde{\mathbf{H}}_2 \end{aligned} \tag{4.20}$$

It can be proved that the system matrix in (4.20) is positive definite. So the network settles to the global solution in the magnitude-squared domain.

4.2.3 Design procedure

In order to design an IIR filter in the magnitude-squared domain, the specified error terms are first squared. The network is then simulated to get the c and d coefficients. The transfer function of the filter in the magnitude-squared domain is constructed. The poles and zeros of the filter are computed next. Poles and zeros outside the unit circle are rejected. Zeros that lie within the unit circle are retained. Only half of the zeros on the unit circle are also retained. The transfer function of the filter is then obtained with these poles and zeros.

4.2.4 Results

The network was simulated to design an IIR filter of length 5. The specifications for the filter were as follows: passband edge $\omega_p = 0.4\pi$, stopband edge $\omega_a = 0.6\pi$, passband error 0.22, and and stopband error 0.0224. The filter was designed following the steps outlined in the above procedure. The amplitude response of the filter is shown in Figure 4.2.

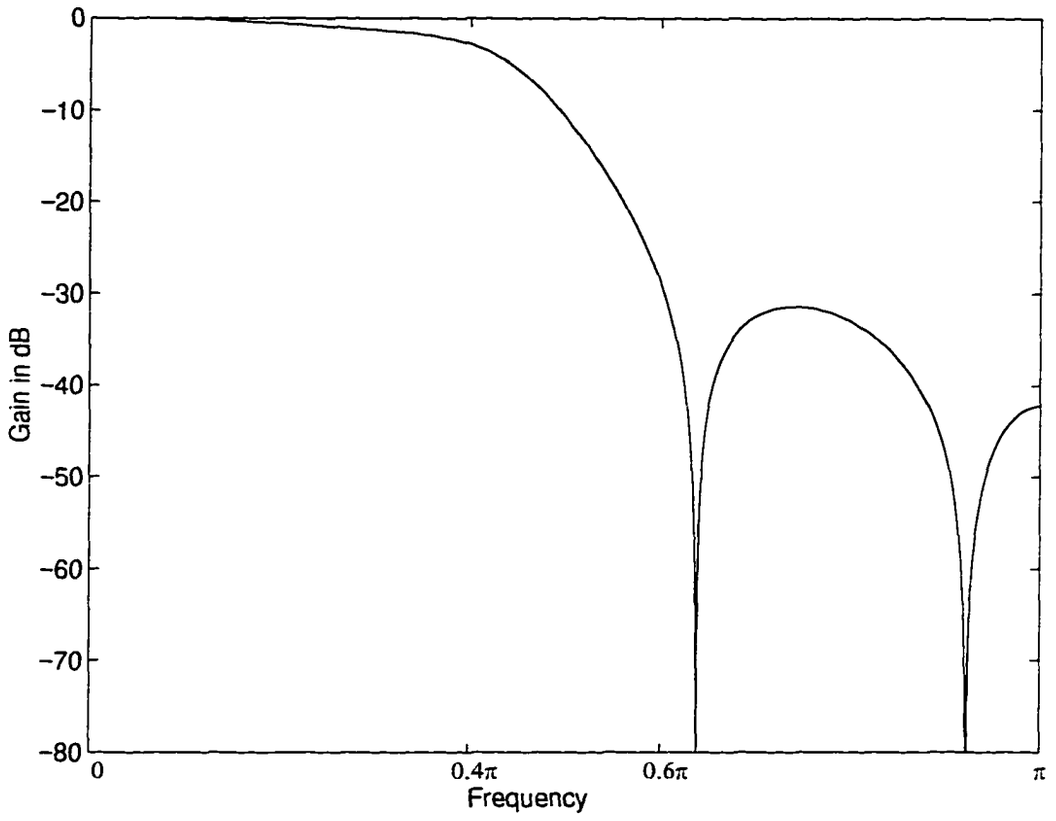


Figure 4.2: Amplitude response of the filter with $\omega_p = 0.4\pi$, $\omega_a = 0.6\pi$.

4.3 Design of IIR filters with specified amplitude and phase responses

The frequency response of an IIR filter whose transfer function is given in (4.1) can be written as

$$H(e^{-j\omega}) = \frac{\sum_{i=0}^{M} b_i e^{-j\omega}}{1 + \sum_{i=1}^{N} a_i e^{-j\omega}} = \frac{N(e^{-j\omega})}{1 + D(e^{-j\omega})}$$

where M and N are the numerator and denominator degrees, respectively. Let the desired response be $\hat{H}(e^{-j\omega})$. The error between the desired response and the response of the designed filter is given by

$$\nu(e^{-j\omega}) = \hat{H}(e^{-j\omega}) - \frac{N(e^{-j\omega})}{1 + D(e^{-j\omega})}$$

Simplifying the above expression, we can write

$$\begin{aligned} E(e^{-j\omega}) &= \left[1 + D(e^{-j\omega})\right] \nu(e^{-j\omega}) \\ &= \hat{H}(e^{-j\omega}) - N(e^{-j\omega}) + \hat{H}(e^{-j\omega}) D(e^{-j\omega}) \end{aligned}$$

where $E(e^{-j\omega})$ is the error to be minimized. Separating the error into real and imaginary parts, we get

$$\begin{aligned} E^r(\omega) &= \mathrm{Re}\left[E(e^{-j\omega})\right] = d^r(\omega) - \\ &\quad \sum_{i=0}^{M} b_i \cos i\omega + d^r(\omega) \sum_{i=1}^{N} a_i \cos i\omega + d^i \sum_{i=1}^{N} a_i \cos i\omega \\ E^i(\omega) &= \mathrm{Im}\left[E(e^{-j\omega})\right] = d^i(\omega) + \\ &\quad \sum_{i=0}^{M} b_i \sin i\omega + d^i(\omega) \sum_{i=1}^{N} a_i \cos i\omega - d^r \sum_{i=1}^{N} a_i \sin i\omega \end{aligned}$$

where $\hat{H}(\omega) = d^r(\omega) + jd^i(\omega)$. Let the desired response be specified at m frequency points. The error function can be written as

$$E = \sum_{j=1}^{m} \left[E^r(\omega_j)^2 + E^i(\omega_j)^2\right] \tag{4.21}$$

This least-squares error can be minimized by the network depicted in Figure 4.3. There are $(M + 1)$ g_b neurons representing the b coefficients, N g_a neurons representing the a coefficients, and m f^r and f^i neurons representing the errors corresponding to the real and imaginary parts. The connection matrix between these two sets of neurons is fixed and consists of the cosines and sines of the weighted frequencies. Both g and f neurons are linear constant-gain amplifiers and are dynamical in the sense that they have RC blocks at their inputs. The time constants of the f neurons are very small compared to those of the g neurons; hence the f neurons can be considered non-dynamical.

Neglecting the dynamics of the f neurons, the input voltages are given by

$$v_j^r = d^r(\omega_j) - \sum_{i=0}^{M} b_i \cos i\omega_j + d^r(\omega_j) \sum_{i=1}^{N} a_i \cos i\omega_j$$

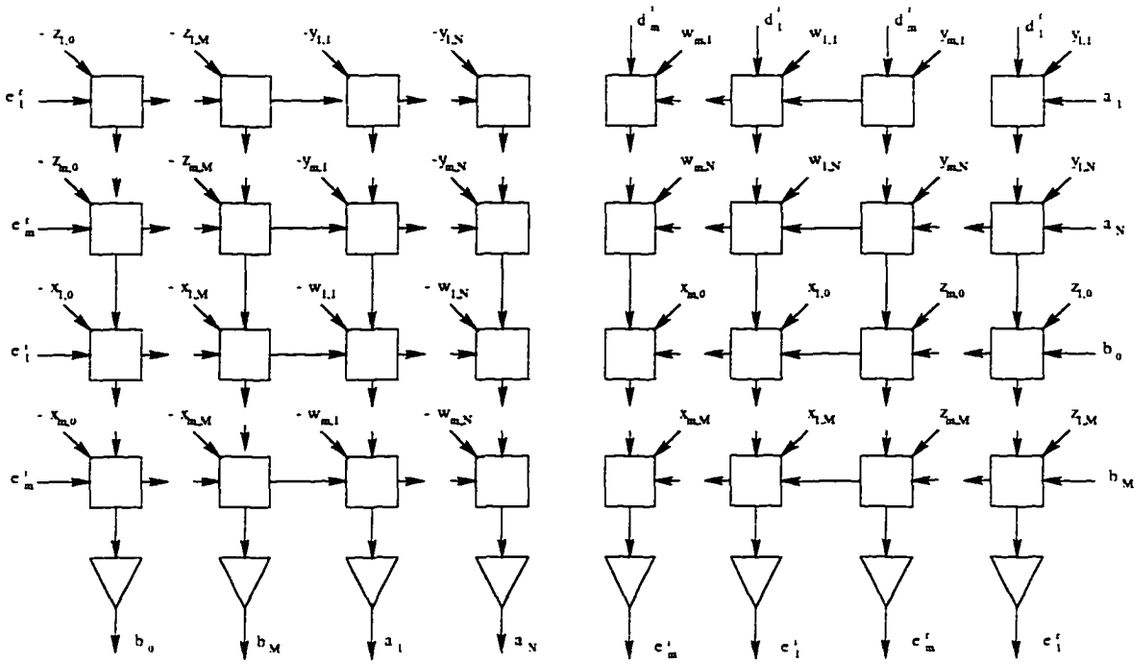


Figure 4.3: Network for the design of IIR filters with specified amplitude and phase responses. The input resistor-capacitor blocks are not shown: $w_{j,i} = d_j^i \cos i\omega_j - d_j^r \sin i\omega_j$, $x_{j,i} = \sin i\omega_j$, $y_{j,i} = d_j^r \cos i\omega_j + d_j^i \sin i\omega_j$, and $z_{j,i} = -\cos i\omega_j$.

$$
\begin{aligned}
& + \quad d^i(\omega_j) \sum_{i=1}^{N} a_i \sin i\omega_j \\
v_j^i = \quad & d^i(\omega_j) \quad + \sum_{i=0}^{M} b_i \sin i\omega_j + d^i(\omega_j) \sum_{i=1}^{N} a_i \cos i\omega_j \\
& - \quad d^r(\omega_j) \sum_{i=1}^{N} a_i \sin i\omega_j
\end{aligned}
$$

The input-output relationship is linear and hence the outputs of the f neurons are given by

$$
\begin{aligned}
e_j^r &= f(v_j^r) = \lambda_f R_f v_j^r \\
e_j^i &= f(v_j^i) = \lambda_f R_f v_j^i
\end{aligned}
\tag{4.22}
$$

In vector notation,

$$
\begin{aligned}
\mathbf{v}^r &= \mathbf{d}^r - \mathbf{C}^b\mathbf{b} + \mathbf{D}^1\mathbf{a} = \mathbf{d}^r - \begin{bmatrix} \mathbf{C}^b & -\mathbf{D}^1 \end{bmatrix} \mathbf{x} \\
\mathbf{v}^i &= \mathbf{d}^i + \mathbf{S}^b\mathbf{b} - \mathbf{D}^2\mathbf{a} = \mathbf{d}^i + \begin{bmatrix} \mathbf{S}^b & -\mathbf{D}^2 \end{bmatrix} \mathbf{x}
\end{aligned}
\tag{4.23}
$$

where $\mathbf{C}^b$ is an $m \times (M+1)$ cosine matrix. $\mathbf{S}^b$ is an $m \times (M+1)$ sine matrix. $\mathbf{x} = [b_0\ b_1\ \cdots\ b_M\ a_1\ a_2 \cdots\ a_N]^T$, and $\mathbf{D}^1$ and $\mathbf{D}^2$ are defined as

$$\mathbf{D}^1 = \begin{bmatrix} d^r(\omega_1)\cos\omega_1 + d^i(\omega_1)\sin\omega_1 & \cdots & d^r(\omega_1)\cos N\omega_1 + d^i(\omega_1)\sin N\omega_1 \\ d^r(\omega_2)\cos\omega_2 + d^i(\omega_2)\sin\omega_2 & \cdots & d^r(\omega_2)\cos N\omega_2 + d^i(\omega_2)\sin N\omega_2 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$\mathbf{D}^2 = \begin{bmatrix} d^r(\omega_1)\sin\omega_1 - d^i(\omega_1)\cos\omega_1 & \cdots & d^r(\omega_1)\sin N\omega_1 - d^i(\omega_1)\cos N\omega_1 \\ d^r(\omega_2)\sin\omega_2 - d^i(\omega_2)\cos\omega_2 & \cdots & d^r(\omega_2)\sin N\omega_2 - d^i(\omega_2)\cos N\omega_2 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{4.24}$$

The equation of motion for a g neuron corresponding to the b coefficients can be written as

$$C\frac{du_i^b}{dt} + \frac{u_i^b}{rg} = \sum_{j=1}^{m}(\epsilon_j^r - u_i^b)\cos i\omega_j - \sum_{j=1}^{m}(\epsilon_j^i + u_i^b)\sin i\omega_j$$

$$C\frac{du_i^b}{dt} + G_i^b u_i^b = \sum_{j=1}^{m}\left[e_j^r \cos i\omega_j - e_j^i \sin i\omega_j\right] \tag{4.25}$$

for $i = 0, 1, \ldots, M$. The input resistance is represented by r_g and G_i^b is the total input conductance of neuron i and is given by

$$G_i^b = \frac{1}{r_g} + \sum_{j=1}^{m}\cos i\omega_j + \sum_{i=1}^{m}\sin i\omega_j$$

The activities of all g^b neurons can be put together through the differential equation

$$C\frac{d\mathbf{u}^b}{dt} + \mathbf{G}^b\mathbf{u}^b = \mathbf{C}^{b^T}\mathbf{e}^r - \mathbf{S}^{b^T}\mathbf{e}^i \tag{4.26}$$

Similarly, for the g^a neurons.

$$C\frac{du_i^a}{dt} + \frac{u_i^a}{r_g} =$$
$$-\sum_{j=1}^{m}(\epsilon_j^r + u_i^a)\left[d^r(\omega_j)\cos i\omega_j + d^i(\omega_j)\sin i\omega_j\right]$$
$$+\sum_{j=1}^{m}(\epsilon_j^i - u_i^a)\left[d^r(\omega_j)\sin i\omega_j - d^i(\omega_j)\cos i\omega_j\right]$$

for $i = 1, 2, \ldots, N$. After putting all conductance parts together, the above expression can be written as

$$C\frac{du_i^a}{dt} + G_i^a u_i^a = -\sum_{j=1}^{m} e_j^r\left[d^r(\omega_j)\cos i\omega_j + d^i(\omega_j)\sin i\omega_j\right]$$
$$+\sum_{j=1}^{m} e_j^i\left[d^r(\omega_j)\sin i\omega_j - d^i(\omega_j)\cos i\omega_j\right] \tag{4.27}$$

where

$$G_i^a = \frac{1}{r_g} + \sum_{j=1}^{m} \left[d^r(\omega_j) \cos i\omega_j + d^i(\omega_j) \sin i\omega_j \right] + \sum_{j=1}^{m} \left[d^r(\omega_j) \sin i\omega_j - d^i(\omega_j) \cos i\omega_j \right]$$

In vector notation, the activities of the g^a neurons can be represented in terms of

$$C\frac{d\mathbf{u}^a}{dt} + \mathbf{G}^a\mathbf{u}^a = -\mathbf{D}^{1^T}\mathbf{e}^r + \mathbf{D}^{2^T}\mathbf{e}^i \tag{4.28}$$

where $\mathbf{G}^a$ is a diagonal matrix with the G_i^a's as its entries. The input voltages of the g^b and g^a neurons are denoted by u^b and u^a and the input-output characteristics for the two types of neurons are given by

$$\begin{aligned} b_i &= g(u_i^b) = \lambda_g u_i^b \\ a_i &= g(u_i^a) = \lambda_g u_i^a \end{aligned} \tag{4.29}$$

4.4 Convergence

The energy function of the network is given by

$$E = \sum_{j=1}^{m} F(v_j^r) + \sum_{j=1}^{m} F(v_j^i) + \sum_{i=0}^{M} G_i^b \int_0^{b_i} g^{-1}(b_i)db_i + \sum_{i=1}^{N} G_i^a \int_0^{a_i} g^{-1}(a_i)da_i \tag{4.30}$$

where F is related to the characteristic function f of the f neurons by

$$\frac{dF(z)}{dz} = f(z) \tag{4.31}$$

with z being a dummy variable. Noting (4.31) and carrying out the integration, (4.30) can be rewritten as

$$E = \frac{1}{2}\lambda_f R_f \sum_{j=1}^{m} \left[(v_j^r)^2 + (v_j^i)^2 \right] + \frac{1}{2}\sum_{i=0}^{m} \frac{G_i^b}{\lambda_g} b_i^2 + \frac{1}{2}\sum_{i=1}^{N} \frac{G_i^a}{\lambda_g} a_i^2 \tag{4.32}$$

For a high λ_g/G_i ratio, the last two terms can be neglected and, as a consequence, the energy function coincides with the error function (4.21) to within a constant factor.

In order to check the time derivative of the energy function, we note that

$$\frac{dE}{dt} = \sum_{i=0}^{M} \frac{\partial E}{\partial b_i}\frac{db_i}{dt} + \sum_{i=1}^{N} \frac{\partial E}{\partial a_i}\frac{da_i}{dt} \tag{4.33}$$

Now

$$\frac{\partial E}{\partial b_i} = \sum_{j=1}^{m} \frac{\partial F(v_j^r)}{\partial b_i} + \sum_{j=1}^{m} \frac{\partial F(v_j^i)}{\partial b_i} + G_i^b b_i$$
$$\frac{\partial E}{\partial a_i} = \sum_{j=1}^{m} \frac{\partial F(v_j^r)}{\partial a_i} + \sum_{j=1}^{m} \frac{\partial F(v_j^i)}{\partial a_i} + G_i^a a_i \tag{4.34}$$

The partial derivatives on the right-hand side of (4.34) can be expressed as

$$\frac{\partial F(v_j^r)}{\partial b_i} = \frac{\partial F(v_j^r)}{\partial v_j^r}\frac{dv_j^r}{db_i} = -f(v_j^r)\cos i\omega_j$$
$$\frac{\partial F(v_j^i)}{\partial b_i} = \frac{\partial F(v_j^i)}{\partial v_j^i}\frac{dv_j^i}{db_i} = f(v_j^i)\sin i\omega_j$$
$$\frac{\partial F(v_j^r)}{\partial a_i} = \frac{\partial F(v_j^r)}{\partial v_j^r}\frac{dv_j^r}{da_i}$$
$$= f(v_j^r)[d^r(\omega_j)\cos i\omega_j + d^i(\omega_j)\sin i\omega_j]$$
$$\frac{\partial F(v_j^i)}{\partial a_i} = \frac{\partial F(v_j^i)}{\partial v_j^i}\frac{dv_j^i}{da_i}$$
$$= -f(v_j^i)[d^r(\omega_j)\sin i\omega_j - d^i(\omega_j)\cos i\omega_j] \tag{4.35}$$

After substituting (4.34) - (4.35) in (4.33), the time derivative of the energy function can be written as

$$\frac{dE}{dt} = -\sum_{i=0}^{M} C\frac{du_i^b}{dt}\frac{db_i}{dt} - \sum_{i=1}^{N} C\frac{du_i^a}{dt}\frac{da_i}{dt}$$
$$= -\frac{C}{\lambda_g}\sum_{i=0}^{M}\left(\frac{db_i}{dt}\right)^2 - \frac{C}{\lambda_g}\sum_{i=1}^{N}\left(\frac{da_i}{dt}\right)^2 \tag{4.36}$$

Here we have used (4.25), (4.27) and (4.29). From (4.36) we note that the time derivative of the energy function is always less than or equal to zero and, therefore, the network always settles to a minimum.

In order to check the uniqueness of the solution, the system equations (4.26) and (4.28) can now be combined into

$$C\frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \begin{bmatrix}\mathbf{C}^b & -\mathbf{D}^1\end{bmatrix}^T \mathbf{e}^r - \begin{bmatrix}\mathbf{S}^b & -\mathbf{D}^2\end{bmatrix}^T \mathbf{e}^i \tag{4.37}$$

where $\mathbf{u} = [\mathbf{u}^{b^T}\ \mathbf{u}^{a^T}]^T$ and $\mathbf{G}$ is a diagonal matrix with the conductances as its erntries. Substituting (4.22)-(4.23) in (4.37), we get

$$C\frac{d\mathbf{u}}{dt} + \mathbf{G}\mathbf{u} = \lambda_f R_f \Big[(\mathbf{C}^b \quad -\mathbf{D}^1)^T\mathbf{d}^r - (\mathbf{C}^b \quad -\mathbf{D}^1)^T(\mathbf{C}^b \quad -\mathbf{D}^1)\mathbf{x}$$
$$- (\mathbf{S}^b \quad -\mathbf{D}^2)^T\mathbf{d}^i - (\mathbf{S}^b \quad -\mathbf{D}^2)^T(\mathbf{S}^b \quad -\mathbf{D}^2)\mathbf{x}\Big] \tag{4.38}$$

After some manipulation, (4.38) can be written as

$$\frac{d\mathbf{x}}{dt} + \left[\frac{1}{C}\mathbf{G} + \frac{\lambda_f \lambda_g R_f}{C}(\mathbf{C}^b \quad -\mathbf{D}^1)^T(\mathbf{C}^b \quad -\mathbf{D}^1) + \frac{\lambda_f \lambda_g R_f}{C}(\mathbf{S}^b \quad -\mathbf{D}^2)^T(\mathbf{S}^b \quad -\mathbf{D}^2)\right]\mathbf{x} = \frac{\lambda_f \lambda_g R_f}{C}(\mathbf{C}^b \quad -\mathbf{D}^1)^T\mathbf{d}^r + \frac{\lambda_f \lambda_g R_f}{C}(\mathbf{S}^b \quad -\mathbf{D}^2)^T\mathbf{d}^i$$

It can be proved that the system matrix is positive definite. So the network settles to the global solution in the magnitude-squared domain.

4.4.1 Results

The network was used to design an IIR filter with given amplitude and phase responses as depicted in Figure 4.4(a) and (b). The amplitude and phase responses were specified at 60 uniformly spaced sample points in the frequency range 0 to π. The network was simulated for 5 μs with $M = N = 16$. The amplitude response achieved is shown in Figure 4.4(c) and the phase response is shown in Figure 4.4(d). To achieve the same performance, an FIR filter of length 61 has to be used.

4.5 Conclusions

Two networks for the design of IIR filters have been proposed. The first method deals with designing IIR filters in the magnitude-squared domain and can be used to achieve a given amplitude response. The second method can design an IIR filter with arbitrary amplitude and phase responses. The results look satisfactory, though no control has been imposed on the position of the poles. For the first case, exactly half the poles are outside the unit circle. Since the numbers of poles and zeros in the magnitude-squared domain are twice those in the z domain, those poles which lie outside the unit circle are rejected. For the second case, it was found that if the iteration steps are small, the poles are within the unit circle. However, if the network is allowed to run for longer time, some of the poles go outside the unit circle, thereby making the filter unstable. In order to design a stable filter, one can use a network with constraint on the position of poles.

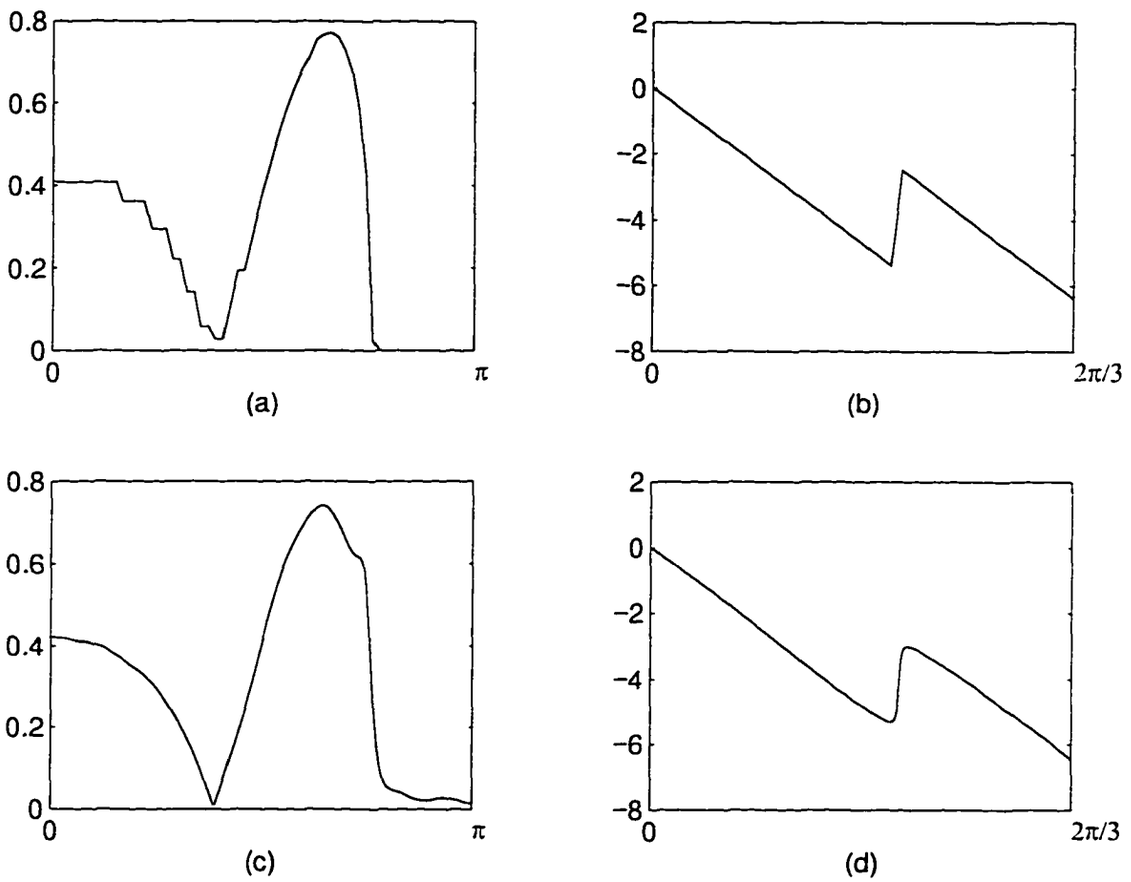


Figure 4.4: (a) Desired amplitude response, (b) desired phase response. (c) amplitude response of the filter obtained, (d) phase response of the filter.

Chapter 5

1-D WLS Filters

5.1 Introduction

The design of FIR filter is essentially a problem of finding a Fourier series and then modifying it through the use of a window function to achieve an approximation with a finite number of terms. If $D(\omega)$ is the required amplitude response and $M(\omega)$ is the response of the actual filter, then the approximation error is given by the difference between the two responses. In a least squares design, the L_2^2 norm of the above error function is minimized. The minimum error achieved tends to be unevenly distributed over the frequency range of interest [5]. In a minimax design, the magnitude of the largest error is minimized and the resulting solution leads to an error which is evenly distributed over the frequency bands. To design filters which are optimum in the minimax sense, one can use the Remez exchange algorithm. The design of FIR filters using the Remez exchange algorithm has been dealt exhaustively in [5]. This algorithm is computationally very intensive and the nature of the solution depends a lot on the way superfluous extremal frequencies are rejected. An alternative to the Remez exchange algorithm is the weighted least-squares (WLS) optimization method [44], [2], and [63]. The WLS method produces an equiripple design, if a suitable least-square weighting function is used and is easy to implement.

The design of FIR filters based on a feedback neural network has been examined in chapter 2. In this chapter, with the addition of a few more blocks, it is shown that the feedback neural network is capable of designing equiripple filters based on the WLS method [12]. Even though all the mathematical formulations are given for FIR filters, the method can be used for designing equiripple digital differentiators (DD's) and Hilbert transformers

(HT's) [11].

5.2 Problem formulation

The weighted least-squares error function can be expressed as

$$E_{\text{mag}} = \sum_{j=1}^{m} W(\omega_j) E^2(\omega_j) \tag{5.1}$$

where

$$E(\omega_j) = D(\omega_j) - G(\omega_j)$$

The quantity m is the number of points at which the error function is sampled, and $W(\omega_j)$ is a weighting function. It is not possible to find $W(\omega_j)$ analytically and an iterative procedure is employed to identify the appropriate weighting function. For a linear phase FIR filter with an odd length of N, the amplitude response is given by

$$M(\omega_j) = |G(\omega_j)|$$

where

$$G(\omega_j) = \sum_{k=0}^{(N-1)/2} a_k \cos k\omega_j$$

where coefficients a_k are linearly related to the filter impulse response $h(n)$ as defined in chapter 2. For differentiators and Hilbert transformers, the phase response must be linear and hence the impulse response is required to be antisymmetrical. The corresponding response is given by

$$G(\omega_j) = \sum_{k=1}^{n} a_k \sin(k-p)\omega_j$$

where $n = (N-1)/2$, $p = 0$ for N odd and $n = N/2$, $p = 1/2$ for N even. Since the amplitude response for antisymmetrical filters contains integral sine terms, full-band DD's and HT's are not realizable when N is odd. This problem can be solved by using an even length. However, an even N gives rise to a fractional delay which may not be acceptable for some applications. However, these problems are not very critical and can be taken care of by choosing an appropriate filter order.

A suitable network for the minimization of the least-squares error is depicted in Figure 5.1. There are $n+1$ g neurons and m f neurons. The connection matrix between

these two sets of neurons is fixed and consists of the cosines (or sines for DD's and HT's) of the weighted frequencies. Both g and f neurons are linear constant-gain amplifiers and are dynamical in the sense that they have RC' blocks at their inputs. The time constants of f neurons are very small compared to those of g neurons; hence they can be considered non-dynamical.

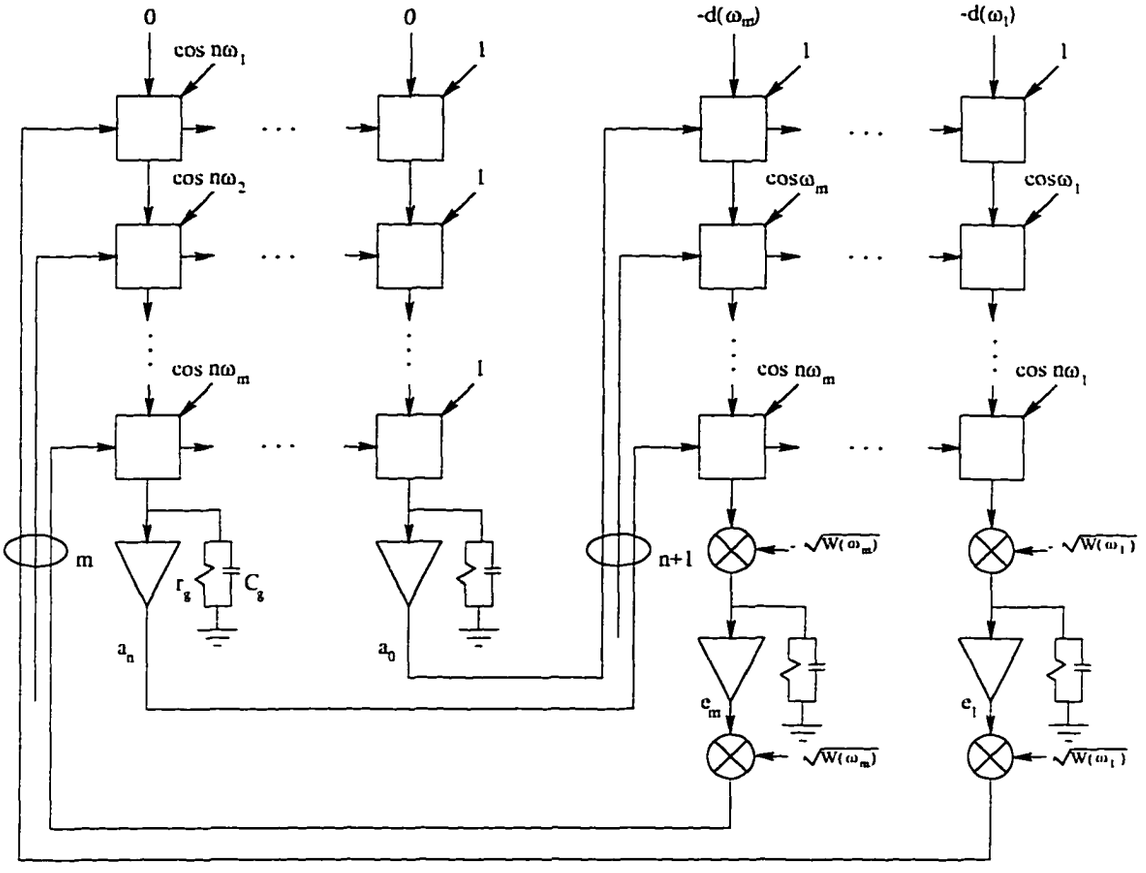


Figure 5.1: Network for the design of 1-D FIR filters using the WLS method.

Neglecting the dynamics of the f neurons,

$$e_j = f\left(-\sqrt{W(\omega_j)}\left[\mathbf{a}^T\mathbf{c}(\omega_j) - d(\omega_j)\right]\right)$$
$$= \lambda_f R_f \sqrt{W(\omega_j)}\left[d(\omega_j) - \mathbf{a}^T\mathbf{c}(\omega_j)\right]$$

for $j = 1, \ldots, m$, $\mathbf{a} = [a_0\ a_1\ \cdots\ a_n]^T$, and $\mathbf{c}(\omega_j) = [1\ \cos\omega_j\ \cdots\ \cos\omega_j n]^T$, and $n = (N-1)/2$. The quantities λ_f and R_f are the gain and the input resistance of an f neuron. The output e_j of f neuron j is multiplied by $\sqrt{W(\omega_j)}$ to assure convergence. The outputs

of all f neurons can be put together as

$$\mathbf{e} = \lambda_f R_f \mathbf{W}_{sq} [\mathbf{d} - \mathbf{Ca}] \tag{5.2}$$

where $\mathbf{d}$ is the vector formed by the values of the desired responses, $\mathbf{C}$ is the $m \times (n+1)$ cosine matrix and

$$\mathbf{W}_{sq} = \begin{bmatrix} \sqrt{W(\omega_1)} & 0 & \cdots & 0 \\ 0 & \sqrt{W(\omega_2)} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \sqrt{W(\omega_m)} \end{bmatrix}$$

The equation of motion of a g neuron is given by

$$C_g \frac{du_i}{dt} + \frac{u_i}{r_g} = \sum_{j=1}^{m} \left[\sqrt{W(\omega_j)} e_j - u_i \right] \cos i\omega_j$$

for $i = 0, \ldots, n$ where C_g and r_g are the input capacitance and resistance, respectively, and u_i is the input voltage of neuron i with a corresponding output of

$$a_i = g(u_i) = \lambda_g u_i \tag{5.3}$$

Alternatively, we can write

$$C_g \frac{du_i}{dt} + G_i u_i = \sum_{j=1}^{m} \sqrt{W(\omega_j)} e_j \cos i\omega_j$$

where

$$G_i = \frac{1}{r_g} + \sum_{j=1}^{m} \cos i\omega_j$$

In terms of vector-matrix notation

$$C_g \frac{du_i}{dt} + G_i u_i = \mathbf{e}^T \mathbf{W}_{sq} \mathbf{c}_i(\omega)$$

where $\mathbf{c}_i(\omega) = [\cos i\omega_1 \ \cos i\omega_2 \ \cdots \ \cos i\omega_m]^T$. The equation of motion of all the g neurons can be expressed as

$$\begin{aligned} C_g \frac{d\mathbf{u}}{dt} + \mathbf{Gu} &= \mathbf{C}^T \mathbf{W}_{sq} \mathbf{e} \\ &= \lambda_f R_f \mathbf{C}^T \mathbf{W}_{sq} \mathbf{W}_{sq} [\mathbf{d} - \mathbf{Ca}] \\ &= \lambda_f R_f \mathbf{C}^T \mathbf{W}_{sq}^2 [\mathbf{d} - \mathbf{Ca}] \end{aligned} \tag{5.4}$$

where

$$\mathbf{G} = \text{diag}\,(G_0 \ G_1 \ \cdots \ G_n)$$

5.3 Convergence

The energy function of this network is given by

$$E = \sum_{j=1}^{m} F\left(\sqrt{W(\omega_j)}\left[d(\omega_j) - \mathbf{a}^T\mathbf{c}(\omega_j)\right]\right) + \sum_{i=0}^{n} G_i \int_0^{a_i} g_i^{-1}(a_i)da_i \tag{5.5}$$

where function F is related to the characteristic function f of the f neuron by

$$f(z) = \frac{dF(z)}{dz}$$

and z is a dummy variable. Substituting this relation, and carrying out the intergration, one gets

$$E = \frac{1}{2}\lambda_f R_f W(\omega_j)\left[d(\omega_j) - \mathbf{a}^T\mathbf{c}(\omega_j)\right]^2 + \sum_i \frac{G_i}{2\lambda_g}a_i^2 \tag{5.6}$$

If G_i/λ_g is small, we can ignore the second term. Then the energy function (5.6) becomes coincident with the error function given by (5.1).

In order to check convergence, we differentiate (5.5) to get

$$\frac{dE}{dt} = \sum_{i=0}^{n} \frac{\partial E}{\partial a_i}\frac{da_i}{dt} \tag{5.7}$$

Again from (5.5), we have

$$\frac{\partial E}{\partial a_i} = \sum_j \frac{\partial F(y_j)}{\partial a_i} + G_i u_i \tag{5.8}$$

Now

$$\frac{\partial F(y_j)}{\partial a_i} = \frac{\partial F(y_j)}{\partial y_j}\frac{dy_j}{da_i} = -f(y_j)\cos i\omega_j\sqrt{W(\omega_j)} \tag{5.9}$$

Substituting (5.9) in (5.8), we get

$$\frac{\partial E}{\partial a_i} = -C_g\frac{du_i}{dt} \tag{5.10}$$

Now from (5.7), (5.10), and (5.3)

$$\frac{dE}{dt} = -\sum_i \frac{C_g}{\lambda_g}\left(\frac{da_i}{dt}\right)^2 \tag{5.11}$$

From (5.11) we see that $dE/dt \leq 0$. So the network always converges to one of the solutions.

Substituting (5.2) in (5.4), and noting (5.3), the equation of motion for the g neurons can be written as

$$\frac{C_g}{\lambda_g}\frac{d\mathbf{a}}{dt} + \frac{1}{\lambda_g}\mathbf{Ga} = \lambda_f R_f \mathbf{C}^T \mathbf{W}_{sq}^2 [\mathbf{d} - \mathbf{Ca}]$$

$$\frac{d\mathbf{a}}{dt} + \frac{1}{C_g}\mathbf{Ga} = \frac{\lambda_f \lambda_g R_f}{C_g}\mathbf{C}^T \mathbf{W}_{sq}^2 [\mathbf{d} - \mathbf{Ca}]$$

$$\frac{d\mathbf{a}}{dt} + \left[\frac{1}{C_g}\mathbf{G} + \frac{\lambda_f \lambda_g R_f}{C_g}\mathbf{C}^T \mathbf{W}_{sq}^2 \mathbf{C}\right]\mathbf{a} = \frac{\lambda_f \lambda_g R_f}{C_g}\mathbf{C}^T \mathbf{W}_{sq}^2 \mathbf{d}$$

It can be seen that the system matrix is positive definite. This means that the network settles to the global solution for that particular weighting function in each iteration.

5.4 The algorithm

For a least-squares weighting function, the optimum solution can be found analytically. However, there is no known analytical method for deriving the weighting function that would produce a minimax design [44]. Hence, an iterative approach is taken. If $W^k(\omega_j)$ is the weighting function for the frequency point ω_j in the kth iteration, then in $(k+1)$th iteration it is expressed as

$$W^{k+1}(\omega_j) = W^k(\omega_j)\beta^k(\omega_j)$$

where $\beta^k(\omega)$ is a function of the weighted error function [44]. First, all the local minima and maxima of the error function are computed. Between two consecutive minima there is one maximum. These maxima are stored in an array $\mathbf{q}$. Let $\text{peak}_{\text{max}} = \max(\mathbf{q})$ and $\text{peak}_{\text{min}} = \min(\mathbf{q})$. If

$$\frac{\text{peak}_{\text{max}} - \text{peak}_{\text{min}}}{\text{peak}_{\text{max}}} \leq \epsilon$$

where ϵ is a small positive number (say, 0.05), then the error becomes equiripple.

The network is initiated with zero initial conditions and the initial weights are set to 1. The network is run for 5 μ-s. Weights are then updated and the final outputs of the g and f neurons are used as the initial conditions for the next iteration. The error is computed at the end of each iteration and the convergence criterion is checked at the same time. If $W^k(\omega_j)$ and $W^{k+1}(\omega_j)$ are the weights at ω_j in the kth and $(k+1)$th iterations, respectively, then the updating formula can be written as

$$W^{k+1}(\omega_j) = \frac{1}{\text{peak}_{\text{max}}} \frac{W^k(\omega_j) \times |E^{\alpha}_{\text{mag}}(\omega_j)|}{\sum_{j=1}^{m} W^k(\omega_j) \times |E_{\text{mag}}(\omega_j)|} \tag{5.12}$$

It has been found that a value of α in the range $1.0 < \alpha < 1.4$ speeds up the convergence. The WLS algorithm used in conjuction with feedback neural networks is as follows:

1) Set $k = 0$; initialize $W^k = 1$, $\mathbf{a} = \mathbf{0}$, $\mathbf{e} = \mathbf{0}$, $\alpha = 1.0$.

2) Run the network for 5 μs.

3) Compute E_{mag}, peak_{max}, and peak_{min}.

4) If $\frac{(\text{peak}_{\text{max}} - \text{peak}_{\text{min}})}{\text{peak}_{\text{max}}} \leq 0.05$, stop.

5) Set $k = k + 1$ and update weights as in (5.12).

6) Go to step 2.

5.5 The network

Each square element in Figure 5.1 refers to a connection point. It accepts the input signal from the left and passes it to the next column through to the right. The input current is fed from the top and the conductances are fed from the left top corners. Each element multiplies the input voltage by the conductance, adds the input current from the top, and produces an output current which is fed to the next connection point immediately below it. The network uses identical connection elements and identical neurons for each type of neuron. Since the time constants are different for g and f neurons, the input resistor-capacitor blocks are different too and have elements $R_g = 2$ kΩ, $C_g = 0.04$ μF, $R_f = 50$ Ω, and $C_f = 1.0$ nF. Both g and f neurons are linear constant-gain amplifiers with gains λ_g and λ_f, respectively, of 2000. The desired response signals and the cosine conductances are scaled by a factor of 10^{-5} to avoid saturation. Figure 5.2 depicts the details for the weight updating scheme and stopping criterion.

5.6 Results

The network in Figure 5.1 and 5.2 was used to design an FIR filter with $\omega_p = 0.3\pi$, $\omega_a = 0.4\pi$ with a sampling frequency of 2π. The order of the filter was chosen to be 29. The numbers of sample points were taken to be 60 and 120 in the passband and stopband, respectively. The value of α was taken to be 1.2 and it took 8 iterations to reach the desired accuracy. The amplitude response of the filter is shown in Figure 5.3 and the response in the passband is shown in the inset. Figure 5.4(a) and (b) shows the absolute error in the

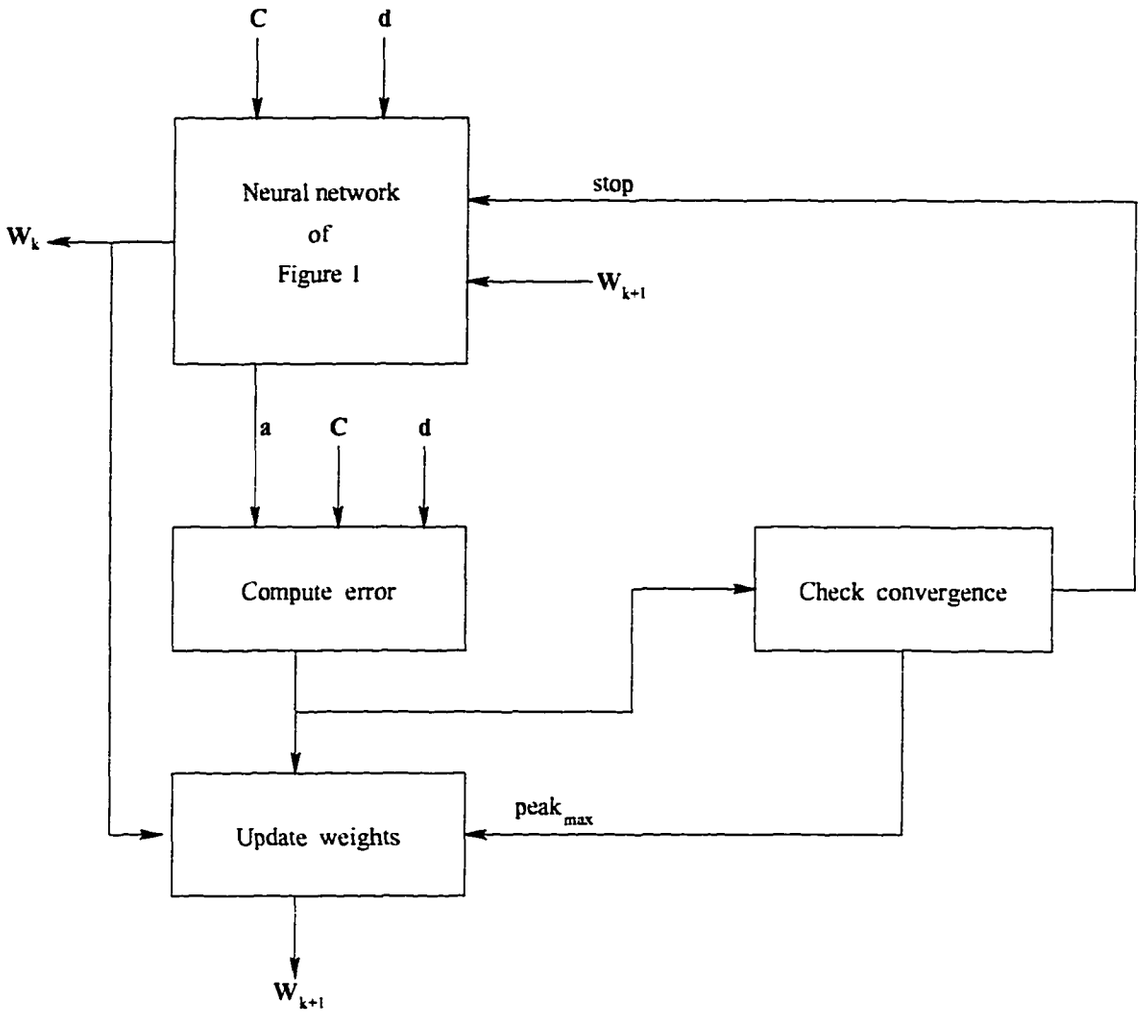


Figure 5.2: Block diagram showing the updating scheme and stopping criterion.

passband and stopband.

The same network was then used to design a digital differentiator with $\omega_p = 0.9\pi$. The number of sample points was taken to be 100 with a filter length of 29. It took 21 iterations with $\alpha = 1.0$. The amplitude response is shown in Figure 5.5(a) and the corresponding error in (b).

In the last example, the network was used to design a Hilbert transformer with bandedges $\omega_l = 0.08\pi$ and $\omega_h = 0.92\pi$. In this case, 100 sample points uniformly distributed in the band of interest were chosen. The parameter α was set to 1 and the filter length was taken to be 31. It took 9 iterations to converge. Figure 5.6(a) shows the amplitude response and (b) shows the error.

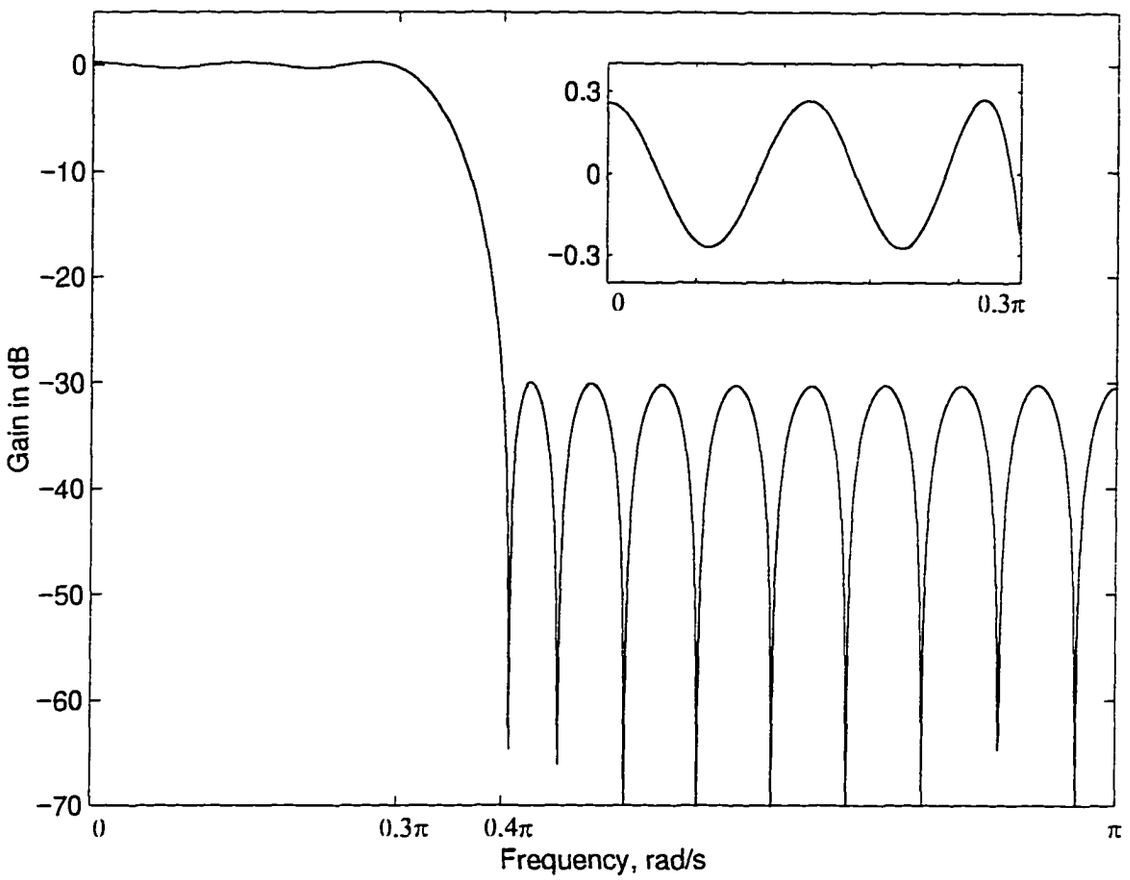


Figure 5.3: Amplitude response of the FIR filter with $\omega_p = 0.3\pi, \omega_s = 0.4\pi$, and filter order is 31. The inset shows the passband ripple.

5.7 Conclusions

A simple yet efficient algorithm for designing 1-D equiripple FIR filters has been proposed. It uses the basic network of chapter 1 for designing 1-D FIR filters for a given amplitude response. A couple of extra multipliers are required by this network to design equiripple FIR filters. The three examples included have shown that this is a viable method for designing equiripple filters. Furthermore it is amenable to analog VLSI implementation.

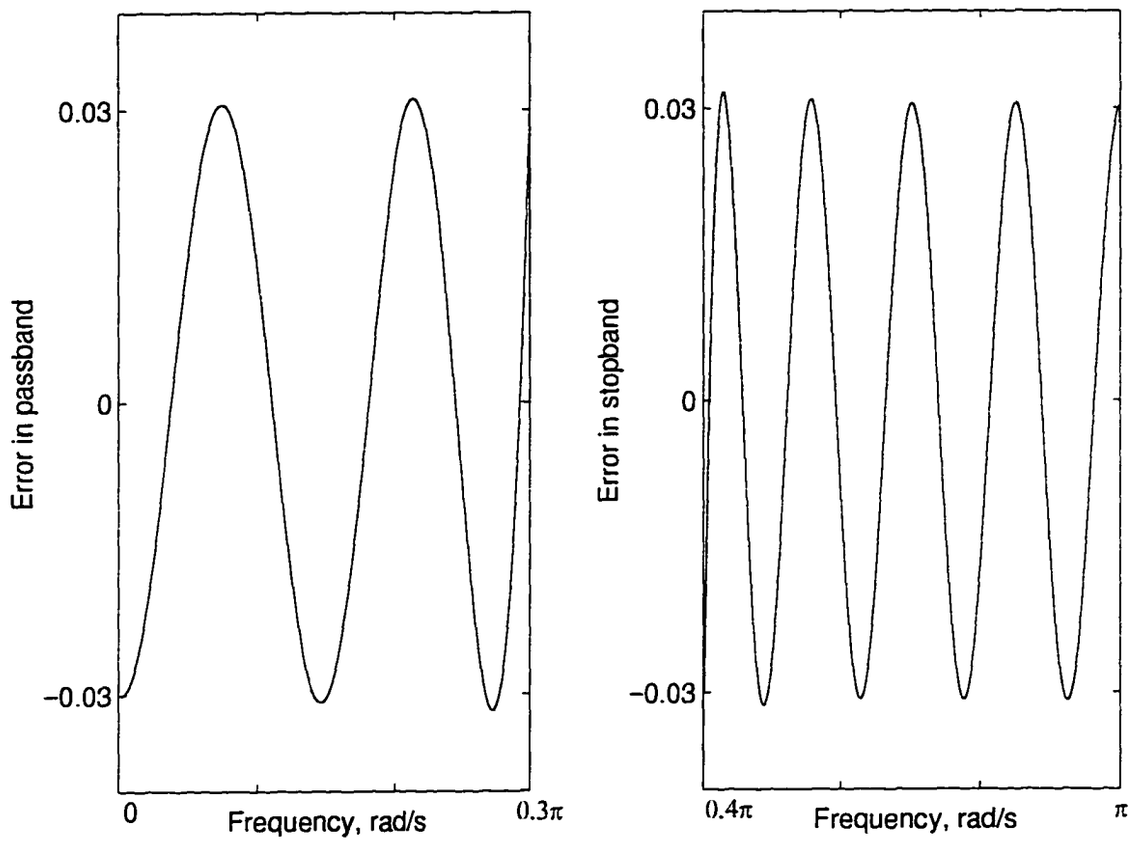


Figure 5.4: (a) Passband error, (b) stopband error of the filter of Figure 5.3

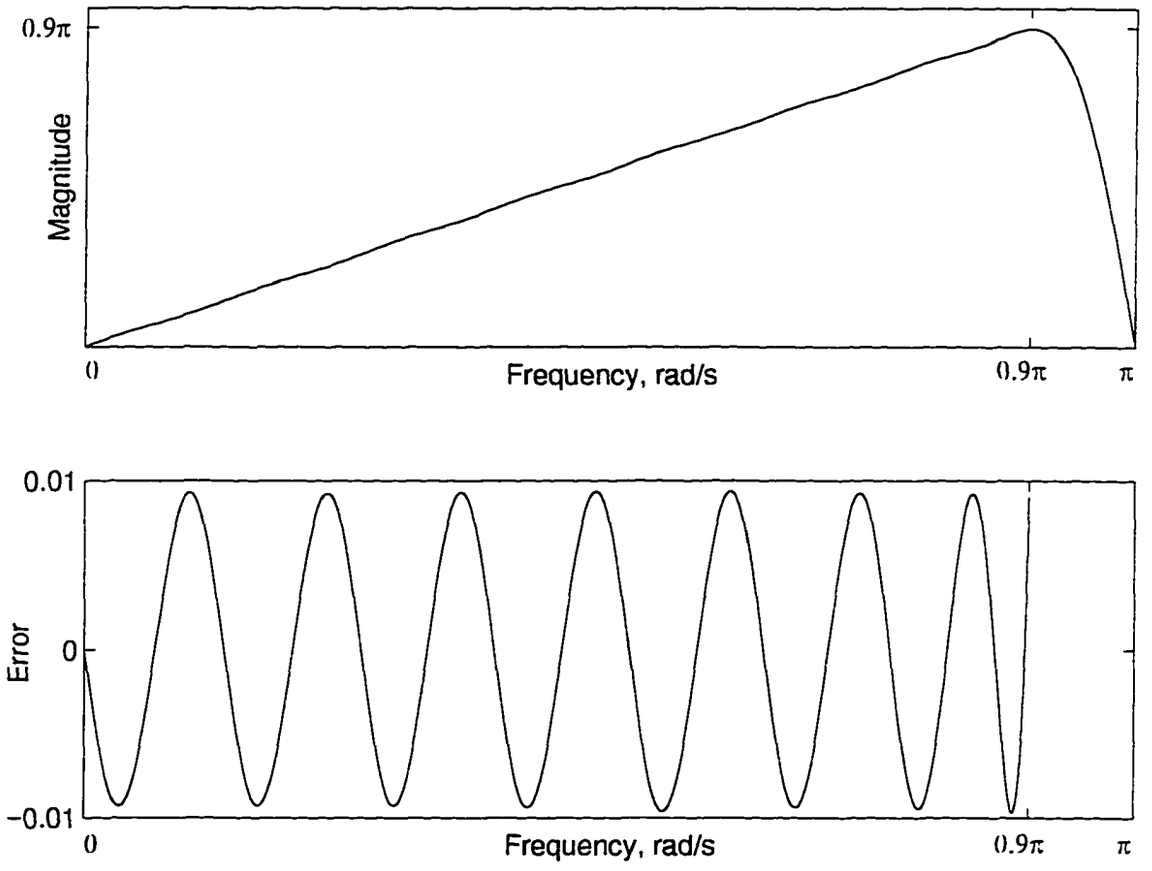


Figure 5.5: (a) Amplitude response of the differentiator with $\omega_p = 0.9\pi$ and $N = 29$. (b) variation of the error with frequency.

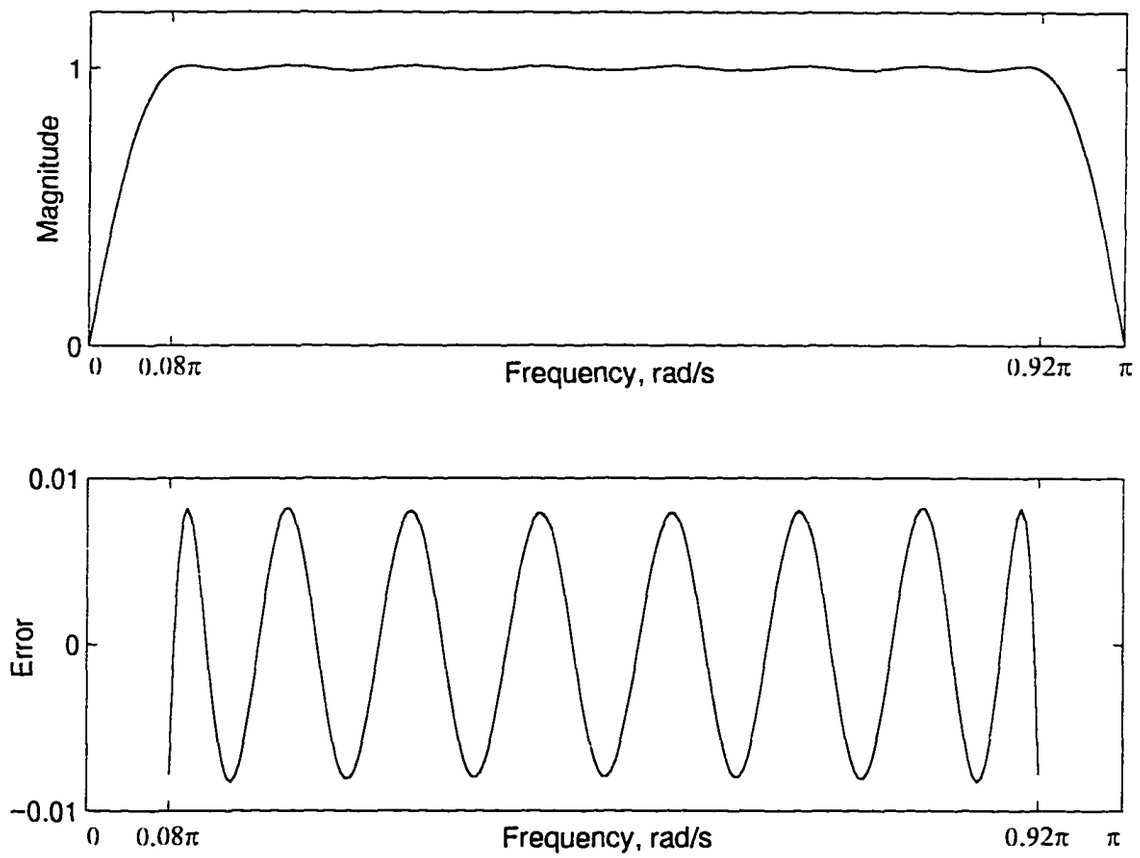


Figure 5.6: (a) Amplitude response of the Hilbert transformer with $\omega_l = 0.08\pi$, $\omega_h = 0.92\pi$ with $N = 31$, (b) variation of the error with frequency.

Chapter 6

Application of Neural Networks

6.1 Introduction

As mentioned in chapter 1, neural networks find numerous applications in signal processing, particularly in image and speech processing. The application of neural networks in emerging techniques like the lidar system is fairly new. Typically, the lidar system is used for mapping sea-bed topography in coastal areas [31], but it can also be used for the detection of fish. For some important commercial species like herring, near-shore waters are the usual habitats of young fish. Because of the problems associated with the use of big ocean-going vessels in coastal areas, lidar technology proves to be one of the most attractive choices for this purpose. However, a reliable method is required by which small variations in the waveform due to the presence of fish can be detected.

This chapter is concerned with two schemes for processing lidar waveforms by neural networks [7], [10]. The first one is for the classification of waveforms in a fishing environment. The lidar waveforms are clustered in an unsupervised manner into various categories depending on some similarity metric. In order to reduce the training time and to assign a particular number to each cluster, a supervised clustering method is also developed. Both algorithms use single-layer neurons with linear activation functions. The spatial results obtained from this method would be useful for the fishery environment. The second scheme is also intended for the classification of waveforms and information extraction from the lidar waveform. It consists of a two-stage multi-layer feedforward neural network employing a supervised learning algorithm. An input waveform is first mapped into some characteristics at the output of stage I. These characteristics are then used in stage II to extract various

types of parameters and to assign a signature number to the waveform. The classification obtained by this method is then compared with the assigned grouping obtained by the first scheme. The chapter is organized as follows: Section 6.2 deals with the preprocessing of the waveforms before being presented to the network. In section 6.3, a detailed analysis of the proposed single-layer neural network is provided, whereas section 6.4 deals with the multi-stage network. Section 6.5 contains results obtained by applying the proposed schemes on real-life lidar data collected off the coast of Vancouver Island and the Canadian Arctic region. A comparative study of the proposed schemes is included and their merits are discussed.

6.2 Preprocessing

Raw lidar waveforms received from the ocean cannot be presented to a neural network directly. It is required that the waveforms be preprocessed suitably so that the network can recognize them, and to ensure at the same time that the processed waveforms reflect subtle variations in the original waveforms properly. One- and two-dimensional discrete-time Fourier transforms are popular for this purpose. However, for lidar waveforms, high-frequency components are typically of reduced magnitude and frequency components generally concentrate in the lower range. In addition, variations in the waveforms due to different conditions are not reflected well in these methods.

A simple preprocessing method based on mapping the waveform on a two-dimensional grid has been developed and will now be described. The waveform is placed on a rectangular grid where the spacings between the horizontal and the vertical grid lines can be set according to the need. As should be expected, higher accuracy of preprocessing can be achieved by higher grid density. However, a higher grid density increases the numerical complexity and the neural network would take more time to process the information, i.e., the choice of grid density is a compromise between accuracy and computational complexity.

For most of the tasks, only a part of the waveform needs to be preprocessed, and the user must decide on the range of sample points that would be relevant. When this is done, each sample point is checked against the grid-crossing points. If a sample point falls within half a grid spacing length on either side in both the horizontal and vertical directions, a one is assigned at that point. Otherwise, a zero is assigned. Each sample point is tested

in the same way and at the end one obtains a matrix of zeros and ones. It is to be noted that the number of ones is much lower than the number of zeros. However, any change in the waveform due to a change in the ocean depth or biomass is faithfully reflected on the matrix.

6.3 Single-layered neural network for waveform clustering

Clustering is the process of grouping similar objects together and separating dissimilar ones [72]. Classes must be found from the correlation of the input data. Once the input data is properly classified, each group is labeled with an appropriate category name or number. This process is popularly known as calibration.

Clustering is important in the fishing environment. Each waveform typically contains the easting and northing sounding locations. After proper clustering, one can display the result with respect to the north and east coordinates. Thus the spatial distribution of the biomass can be easily obtained. This result is very useful for commercial and conservation purposes. In fact, this information, coupled with sea-bed structure gives a good indication of biomass habitat.

A neural network, capable of clustering input data is shown in Figure 6.1. The input and output values of the network are denoted as x_{ij} and y_k, respectively, for $j = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, p$, i being the index of training patterns. An input waveform (or part thereof) is first preprocessed to generate a matrix of zeros and ones. A column vector $\mathbf{x}_i$ is formed using the rows of this matrix. The number of neurons corresponds to the number of categories the input data are clustered into. Let the number of clusters be p. Note that to start the training sequence, it is not required to know which category each input pattern belongs to, but the number of categories can be assumed to be known a priori. Input waveforms are classified into several groups depending on some similarity metric. Several such algorithms exist in the literature [32], [72]. However, Kohonen's winner-take-all learning algorithm [41] was found to be very simple and suitable for our purpose.

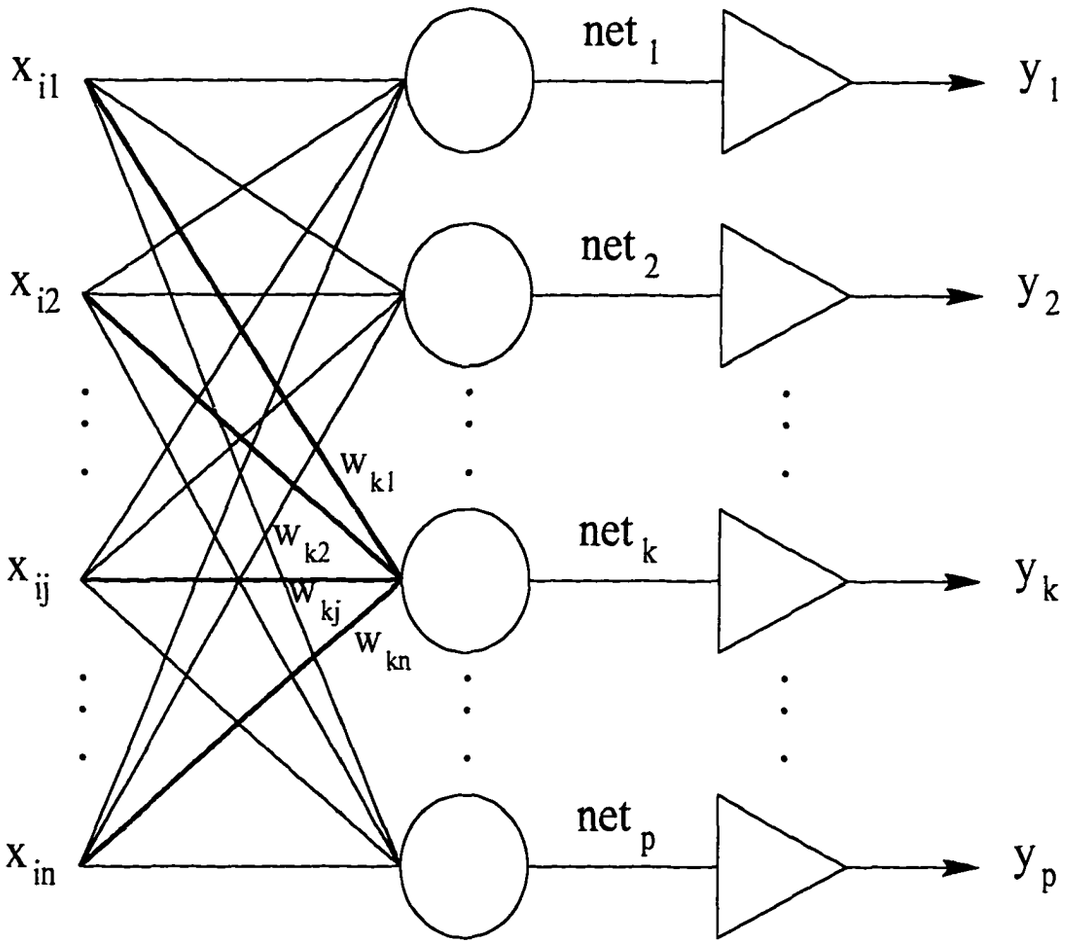


Figure 6.1: Winner-take-all neural network with linear neurons. Neuron k is the winning one with the corresponding weights highlighted.

6.3.1 Unsupervised learning

Let the input patterns be $\{\mathbf{x}_i : i = 1, \ldots, N\}$ where $\mathbf{x}_i = [\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{in}]^T$. Input points are connected to p neurons by the weight matrix $\mathbf{W} \in R^{(p \times n)}$ given by

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ w_{p1} & w_{p2} & \cdots & w_{pn} \end{bmatrix}$$

where element w_{kj} denotes the weight from input j to neuron k. The activation vector of the network for the input pattern i is the sum of the weighted products of the input and is

given by

$$\mathbf{act} = \mathbf{W}\mathbf{x}_i$$

Since linear neurons with unit gain are employed, the output vector is given by

$$\mathbf{y} = \mathbf{act}$$

The essential feature of this learning algorithm is that for each input pattern $\mathbf{x}_i$, one of the p neurons (say the kth one) would have the highest activity. This kth neuron is declared the winner and its weights, fanning in onto the kth neuron, are changed in a suitable manner so that next time the same input occurs, the activity of this neuron would be still higher. In other words, the distance between the input $\mathbf{x}_i$ and the weight vector $\mathbf{w}_k$ where $\mathbf{w}_k = [w_{k1}, w_{k2}, \ldots, w_{kn}]^T$ is minimized.

When an input pattern $\mathbf{x}_i$ is presented to the network, it computes the distance $||\mathbf{x}_i - \mathbf{w}_l||$ for $l = 1, 2, \ldots, p$ and the one with the minimum distance (the kth one) is declared the winner. The way this is done is as follows. The distance can be written as

$$||\mathbf{x}_i - \mathbf{w}_l|| = [\mathbf{x}_i^T\mathbf{x}_i - 2\mathbf{w}_l^T\mathbf{x}_i + 1]^{1/2}$$

Finding the minimum is equivalent to finding the maximum of $\mathbf{w}_l^T\mathbf{x}_i$ for $l = 1, 2, \ldots, p$. In other words, the neuron with the highest activation value is the winning neuron. Now, the weight of this kth neuron is changed along the negative gradient direction, that is,

$$\Delta_{\mathbf{w}_k}||\mathbf{x}_i - \mathbf{w}_k||^2 = -2(\mathbf{x}_i - \mathbf{w}_k)$$

In effect, the weight $\mathbf{w}_k$ is changed as

$$\mathbf{w}_k \leftarrow \mathbf{w}_k + \alpha(\mathbf{x}_i - \mathbf{w}_k) \tag{6.1}$$

where α is a learning constant chosen between 0.1 and 0.8. All other weight vectors are left unchanged.

The above process is repeated for all N input patterns over and over again until the desired accuracy is achieved. When the network is trained, each weight vector points at the centroid of the cluster formed by the input patterns. In order to satisfy the termination criterion, the centroid of each cluster is determined after each training session. The square of the distance of the centroid from the respective weight is computed. If this quantity falls

below a value, say 0.01, for each of the clusters, training is terminated. The initial weights are chosen randomly.

It is to be noted that each training pattern is normalized to start with. Each weight vector is also normalized as

$$\mathbf{w}_l = \frac{\mathbf{w}_l}{||\mathbf{w}_l||}$$

for $l = 1, 2, \ldots, p$. Weights are also normalized after each update. This is required because the purpose of this training method is to move the weight vector towards the centroid of each cluster. This ensures that all input patterns and weight vectors lie on the surface of a unit hypersphere.

In order to distinguish between subtle variations among the input patterns, a so called leaky learning algorithm can also be used [72]. In this method, all the neurons have their weight changed in proportion to their activities. The winning neuron has its weights changed as given in (6.1) whereas the others are changed as

$$\mathbf{w}_l \leftarrow \mathbf{w}_l + 0.01 y_l \alpha(\mathbf{x}_i - \mathbf{w}_l) \tag{6.2}$$

for $l = 1, 2, \ldots, p$ and $l \neq k$. The quantity y_l is the output of neuron l, which is the same as its activity. The factor 0.01 was found to give the winning neuron more chance to succeed compared to the other neurons which may have activities of the same order.

The value of α is chosen to be 0.5 to start with and is decreased monotonically as the learning proceeds. This value was found heuristically and gives good results. The learning algorithm is as follows:

Algorithm A

a) Normalize input vectors $\mathbf{x}_i$ for $i = 1, 2, \ldots, N$.

b) Choose initial weight matrix $\mathbf{W}^0$ randomly and normalize each row with respect to itself. Set $i = 1$.

c) Input $\mathbf{x}_i$ and compute $\mathbf{y}$. The kth neuron is declared winner if $y_k > y_l$ for $l = 1, 2, \ldots, p$ and $l \neq k$.

d) Update $\mathbf{w}_k$ as in (6.1) and $\mathbf{w}_l$ for $l = 1, 2, \ldots, p$ and $l \neq k$ as in (6.2). Normalize the weights.

e) Repeat steps (c) to (e) for $i = 2, \ldots, N$.

f) Compute the square of the distance between the centroid of each cluster and the respective weight vector. If each of these quantities is less than 0.01, save the weight matrix and stop; else set $i = 1$ and go to step (c).

This is an unsupervised learning algorithm where the network learns by itself to cluster the input patterns into a number of categories. In order to capture all the clusters (which may not be known a priori), p can be set to a higher value. These extra clusters would not be detected and the weights corresponding to the actual clusters would only be saved. This is required because in the training mode, those unwanted weights would still be updated and would contain nonzero entries. In order to suppress the triggering of those spurious neurons in the recall mode, it is required to save only those weight vectors that correspond to a successful categorization of the patterns.

Another point that has to be emphasized here is that depending on the initial weight vectors and the sequence of input patterns, patterns will be classified into one of the p clusters. The category into which a particular input pattern is classified is not known a priori. In order to assign a particular category number to any input pattern, a supervised learning scheme can be devised. It has been found that the supervised learning method also accelerates the learning process.

6.3.2 Supervised learning

In the supervised learning method, if an input pattern is classified properly its weight is updated as in (6.1). If its classification is wrong, the corresponding weight is reduced. The amount by which the weight is reduced (β as opposed to α) also gives another parameter to tune up the learning process. This process can further be accelerated if a particular pattern belonging to cluster k is misclassified into any other group (say r). What is being done here is that not only $\mathbf{w}_r$ is reduced, but weight $\mathbf{w}_k$ is also increased appropriately so that next time the same input occurs, the probability of neuron k being the winner is higher. The algorithm now assumes the form:

Algorithm B

a) Normalize input vectors $\mathbf{x}_i$ for $i = 1, 2, \ldots, N$.

b) Choose initial weight matrix $\mathbf{W}^0$ randomly and normalize each row with respect to itself. Set $i = 1$.

c) Input $\mathbf{x}_i$ and compute $\mathbf{y}$. The kth neuron is declared winner if $y_k > y_l$ for $l = 1, 2, \ldots, p$ and $l \neq k$.

d) If $k = \mathbf{d}(i)$, where vector $\mathbf{d}(i)$ contains category numbers, update $\mathbf{w}_k$ as in (6.1); else update weights as

$\mathbf{w}_{\mathbf{d}(i)} \leftarrow \mathbf{w}_{\mathbf{d}(i)} - \beta \times (\mathbf{x}_i - \mathbf{w}_{\mathbf{d}(i)})$

$\mathbf{w}_k \leftarrow \mathbf{w}_k + \alpha \times (\mathbf{x}_i - \mathbf{w}_k)$

Normalize the weights.

e) Repeat steps (c) to (e) for $i = 2, \ldots, N$.

f) Compute the square of the distance between the centroid of each cluster and the respective weight vector. If each of these quantities is less than 0.01, save the weight matrix and stop; else set $i = 1$ and go to step (c).

6.3.3 Recall mode

In the recall mode, waveforms are tested for proper cluster assignment. Waveforms are first preprocessed and for each of the waveforms, the activation and, in turn, the output vector is computed. The number corresponding to the winning neuron is assigned to the waveform. One point to be noted here is that each of the weight vectors is a nonzero quantity and when the nonzero input vector is multiplied by the weight vector and added together, the activation value is typically positive. So, any waveform that does not belong to any of the trained clusters would still produce a winning neuron. However, in this case we have a classic case of a misclassification. In order to suppress this problem, the maximum, average, and minimum values of the neural network outputs in the last round of training are stored and are used as thresholds. Depending on the desired accuracy in the recall mode, the outputs of the neurons are compared with one of these thresholds. If the output is greater than or equal to the threshold value, the group number corresponding to the winning neuron is assigned to the waveform. This method gives the user the flexibility to cluster the waveforms in different groups properly.

6.4 Multi-stage multi-layer neural networks

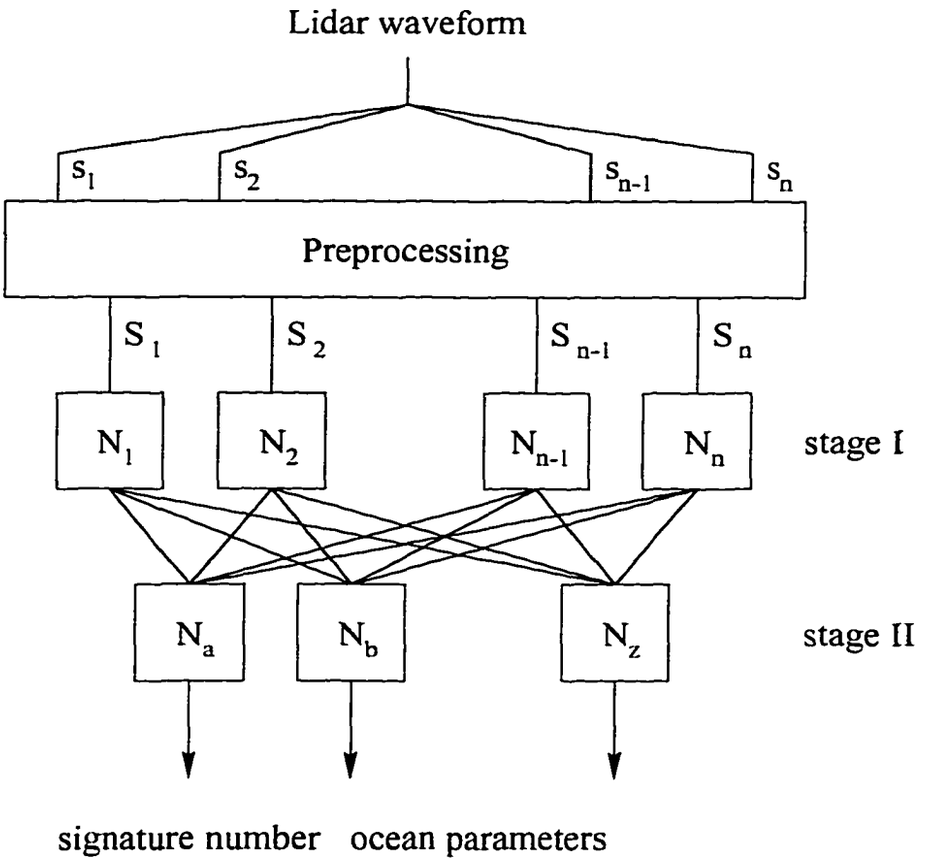
A lidar waveform typically comprises a surface reflection, a bottom reflection, and a back-scattered envelope in between. The position and height of these two reflections contain a lot of information such as depth, sea-bed structure etc. The backscatter envelop, on the

other hand, reflects the clarity of water. It reduces rapidly if the water is very clear. On the other extreme, when the water is very turbid, the rate of reduction is low. This implies that the slope of the envelope contains information about the turbidity of the water. Again, if light encounters any floating or submerged object in its path, a small peak or bulge will appear on the waveform.

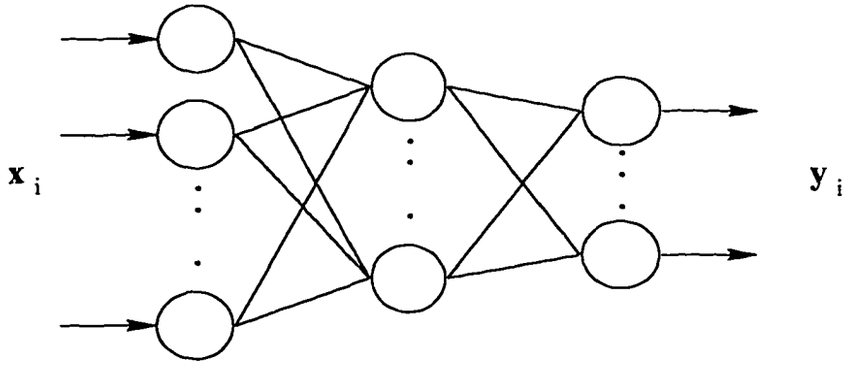
The proposed network is shown in Figure 6.2. Part (a) shows the overall structure and each network depicted as a rectangular block in part (a) is shown in part (b). In stage I, each waveform is divided into segments s_i, $i = 1, 2, \ldots, n$ and preprocessed into a matrix $\mathbf{S}_i$. As before, rows are put next to each other to form the input column vector $\mathbf{x}_i$ which is then presented to as many neural networks N_i. This method has some advantages. First, by having a small number of sample values in each group, preprocessing can be done more accurately. Second, the whole task is broken down into a number of sub-tasks. So, the computational burden can be distributed among all these networks, thereby reducing the learning time significantly. Each of the networks N_i is a three-layer network. The number of neurons in the input layer, the hidden layer and the output layer can vary from segment to segment. Associated with each input vector $\mathbf{x}_i$, there is a target vector $\mathbf{d}_i$. The target vector or the desired response contains information such as presence or absence of the surface reflection, position of the surface reflection, presence or absence of the bottom reflection, its position, slope of the backscatter envelop etc. In other words, the characteristics of each waveform segment are reflected on the desired output.

The neurons in each layer are connected to the neurons in the next layer through weights. Initially, the weights are set in a random order. These connectivities or the weights coupled with the nonlinear neurons in the hidden and output layers nonlinearly map the input space onto the output space. Since the inputs contain either a zero or one, unipolar neurons with a sigmoidal characteristic are being used. A supervised learning scheme such as the back-propagation learning algorithm [59] is employed for training the network in stage I. Only the relevant portion of the training scheme follows, the details being well documented in the literature [72], [59]. In this method, the sum of the instantaneous errors over all the training patterns is minimized, i.e.,

$$\text{minimize } E = \sum_{l=1}^{P} \left\{ \left[\mathbf{d}_i^l - N_i(\mathbf{x}_i^l) \right]^T \left[\mathbf{d}_i^j - N_i(\mathbf{x}_i^l) \right] \right\}$$



(a)



(b)

Figure 6.2: Two-stage multi-layered feedforward neural networks. (a) The overall structure. (b) structure of network N_i.

where P is the number of training patterns. $N(\mathbf{x}_i)$ is the nonlinear mapping of the input $\mathbf{x}_i$ onto the output space. In the training procedure, these weights are updated iteratively in the gradient descent direction given by

$$\Delta w = -\eta \frac{\partial E}{\partial w}$$

Let us assume that the number of neurons in the input, hidden layer, and output layer are given by I, J, and K, respectively. The weight matrix between the input and the hidden layers is $\mathbf{V} \in R^{J \times I}$ and that between the hidden layer and the output layer is $\mathbf{W} \in R^{K \times J}$. The input pattern is $\mathbf{x}$, the output of the hidden layer is $\mathbf{y}$, and the final output is $\mathbf{o}$. All neurons in the hidden and the output layers have activation function given by

$$f(\text{net}) = \frac{1}{1 + \exp(-\lambda \text{net})}$$

where parameter λ defines the slope of the activation function and net is the sum of the weighted product of the inputs with corresponding weights. The weight updating scheme for the output and the hidden layer neurons can be written as

$$w_{kj} \leftarrow w_{kj} + \eta \delta_{ok} y_j$$

$$v_{ji} \leftarrow v_{ji} + \eta \delta_{yj} x_i$$

for $i = 1, 2, \ldots, I$, $j = 1, 2, \ldots, J$, and $k = 1, 2, \ldots, K$. The error signals are defined as

$$\delta_{ok} = \frac{1}{2}(d_k - o_k) o_k (1 - o_k)$$

$$\delta_{yj} = y_j (1 - y_j) \sum_{k=1}^{K} \delta_{ok} w_{kj}$$

Parameter η is the learning rate. To accelerate the training procedure, a momentum term, which is part of the last weight change, can be added in the weight updating expressions.

In stage II, these waveform characteristics are used as the input and the target pattern could be the signature number of the waveform, or the turbidity of the water or any other information that is being extracted from the waveform. Again, a three-stage neural network is employed and is trained by the same back-propagation algorithm. The size of the network in stage II is significantly smaller because the input is a highly encoded form of the waveform. The other advantage is that once the training of stage I is complete for all segments, depending on the task, these results either in their entirety or in parts, can be used for extraction of different types of information. After the training, all the weights are stored for future use.

6.4.1 Recall mode

In the recall mode, input patterns are passed only in the forward direction. In stage I, the input patterns are first submitted to the network and the sum of the weighted product or the net are computed for the hidden layer. The result is passed through the activation function to compute the outputs of the hidden layer. These outputs are, in turn, presented to the weights between the hidden layer and the output layer and the final outputs are computed after passing through the activation function. The outputs from stage I are then presented to the network in stage II and the final outputs are computed as before.

6.5 Results and discussion

6.5.1 Single-layered network

Lidar waveforms recorded by the LARSEN 500 system off the coast of Vancouver Island in March 1995 were used for clustering. We took a profile waveform file which has a sounding pattern in a straight line at two-meter intervals. It was taken on day 65 and line 22 (file Q0650022) with global positioning system (GPS) coordinates of 384932. 5469320 to 393800. 5471703 in the eastern and northern directions. The waveforms received from the surveyed location represent milt of various densities with or without bottom reflections and clear water. The first task was to identify the prototype waveforms for learning purposes. The prototype waveforms from each group are shown in Figure 6.3. In total, 57 waveforms were selected for training purposes and their grouping is given in Table 6.1.

Each waveform contains 256 samples and the easting and the northing sounding locations. It was found that samples 10 – 60 were adequate to represent the characteristics properly. During preprocessing, the number of horizontal and vertical grids were set to 16 and 17. After getting the matrices, the input vectors were formed and normalized. The weight matrix was randomly generated and each row was normalized as stated earlier.

The waveforms were renumbered from 1 to 57, that is, waveforms 1-10 should be in one group, 11-31 in another group, and so on. For unsupervised learning, the number of categories chosen was 7. The network was allowed to train until the desired accuracy was obtained. It took 46 iterations and the number of categories found was 6, one more than

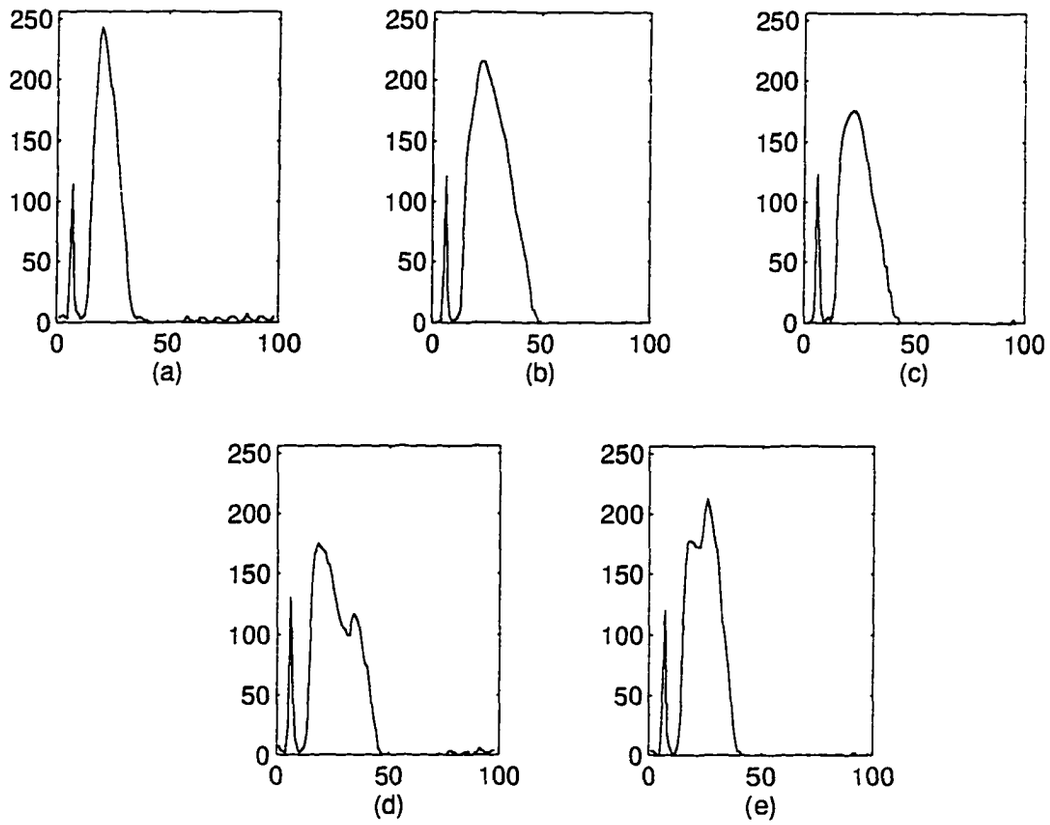


Figure 6.3: (a) - (e) Prototype waveforms (representing groups 1 to 5) used for classifying waveforms corresponding to milt content of varied densities.

what was intended. The initial value of α was set to 0.5 and decreased monotonically. The results are listed in Table 6.2.

The training results are shown in Figure 6.4. Waveforms 11-31 were divided into two groups (group 2 and 3 in Table 6.2) and are shown in Figure 6.4 (b) and (c). As can be seen in the figure, the waveforms in groups 2 and 3 are identical except for the height of the surface returns and the network clustered these waveforms in two separate groups. All other waveforms were clustered as intended.

The same waveforms were then used for supervised learning as listed in algorithm B. As before, α was set to 0.5 and decreased gradually. The classification results are shown in Table 6.3. Waveforms 1-10 were designated as group 1, 11-31 as group 2, and so on. It took 34 iterations and all the waveforms were clustered properly. The results are shown in

Table 6.1: Details of training waveform

Category name	No. of waveforms	Waveform numbers	Assigned waveform numbers
clear water (group 1)	10	2 - 11	1 - 10
high density milt (group 2)	21	1540 - 1560	11 - 31
low density milt (group 3)	9	1940, 1942 - 1946 1948-1950	32 - 40
low density milt with bottom peak (group 4)	6	2470 - 2475	41 - 46
shallow water (group 5)	11	2505 - 2515	47 - 57

Table 6.2: Grouping of waveforms in unsupervised mode

Waveform numbers	Group number
1 - 10	group 4
11- 19 30, 31	group 2
20 - 29	group 3
32 - 40	group 6
41 - 46	group 1
47 - 57	group 5

Figure 6.5.

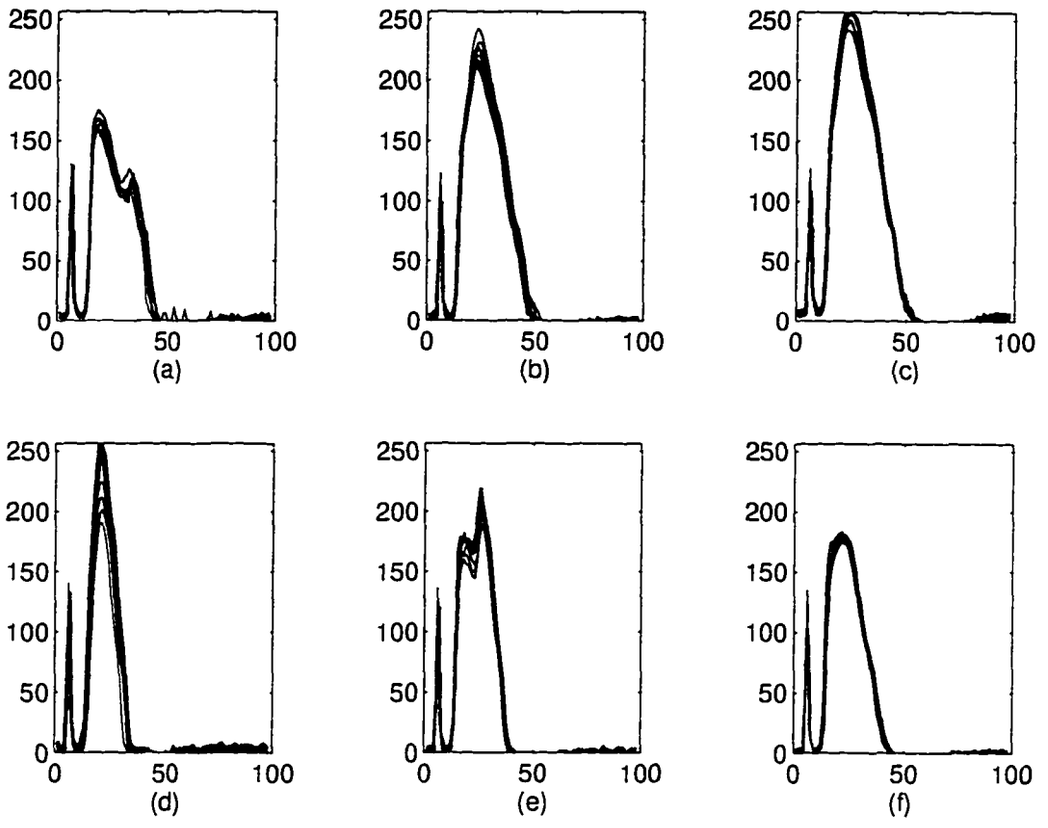


Figure 6.4: Clustering of waveforms obtained through unsupervised learning. The number of clusters is more than intended with arbitrary group number assignments as shown in Table 6.2.

As explained before, after the supervised training, the weight and the threshold values were stored. The original waveform file contained 3162 waveforms. About 40% of the waveforms were contaminated with digitizer noise and were eliminated to leave 1920 waveforms. These waveforms were then preprocessed and ran with the final set of weights. The average threshold value was used and the waveforms were classified into five different groups as intended.

Table 6.3: Grouping of waveforms in supervised mode

Waveform numbers	Group number
1 - 10	group 1
11 - 31	group 2
32 - 40	group 3
41 - 46	group 4
47 - 57	group 5

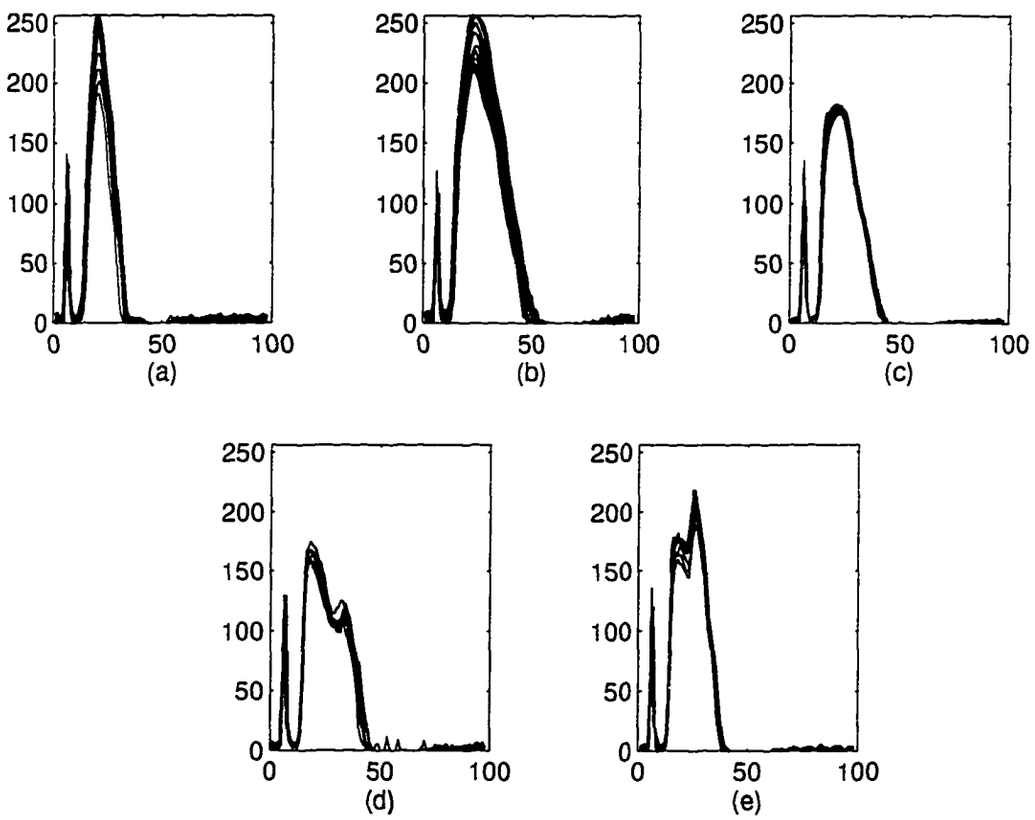


Figure 6.5: Clustering of waveforms in supervised mode with anticipated results.

Figure 6.6 (a) to (e) shows the distribution of each cluster over the same coordinates. Figure 6.6 (f) shows the distribution of all the waveforms together. The distribution looks

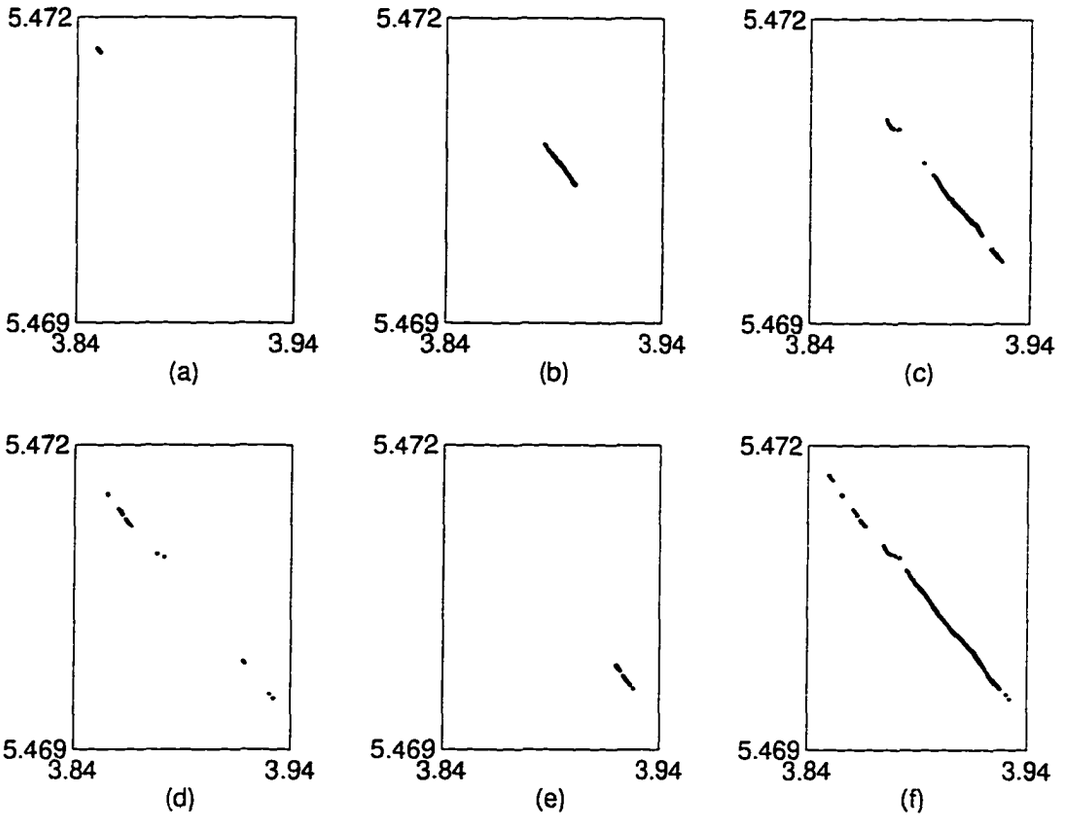


Figure 6.6: Spatial distribution of file Q0650022 obtained by supervised learning. (a)-(e) Distribution of each category of waveforms with respect to the same normalized GPS coordinates. (f) distribution of all waveforms together.

very consistent.

The same weight and threshold values were used on another file (Q0650023), which is line 23 and spans from 384728, 5469303 to 394545, 5471785 in the eastern and northern directions. The results are shown in Figure 6.7. Spatial distributions of waveforms in each group are plotted against the same coordinates. Figure 6.7 (a) to (e) shows the distribution of waveforms in each group whereas the last one shows the overall distribution of all the classified waveforms.

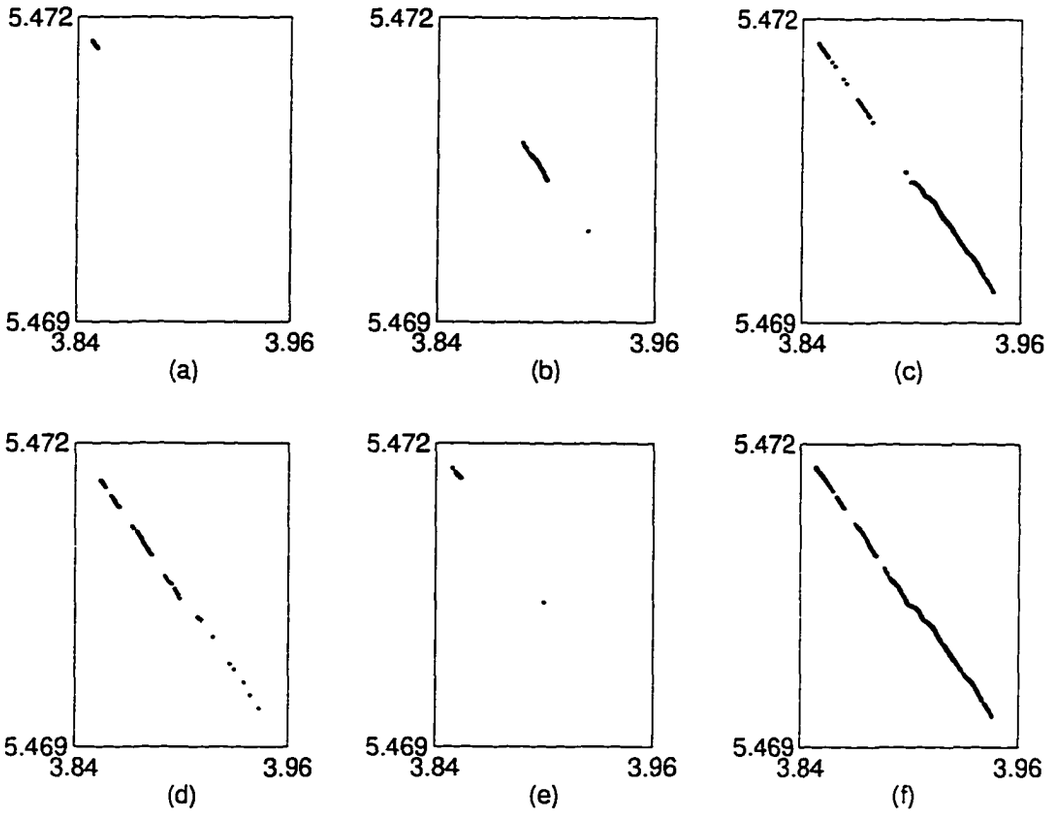


Figure 6.7: Spatial distribution of file Q0650023 obtained by supervised learning. (a)-(e) Distribution of each category of waveforms with respect to the same normalized GPS coordinates, (f) distribution of all waveforms together.

6.5.2 Multi-layered supervised network

As a first example using the two-stage multi-layered feedforward neural network, the waveforms in the previous subsection were assigned a signature number. In stage I, input waveforms were mapped into a set of waveform characteristics chosen suitably for the task at hand. Only one segment of the training waveforms (samples 10 to 60) was used for the learning purpose. The target vectors were classified into 7 different fields as defined in the Table 6.4. In stage II, these characteristics were consolidated into an appropriate signature number. The stage I network contained 272 input neurons, 12 hidden neurons, and 17 output neurons. The number of input neurons corresponds to the total number of entries in matrix S (16×17) and the number of output neurons is the sum of neurons assigned to each of the 7 fields. The number of iterations needed to converge was 94.

Table 6.4: Details of each field in the target vector for stage I

Field no.	No. of neurons	Description
1	1	presence/absence of surface reflection
2	3	amplitude range of surface reflection
3	2	peak position
4	1	presence/absence of bottom reflection
5	4	amplitude range of bottom reflection
6	3	peak position
7	3	position where zero

The stage II network had 17 input neurons, 4 hidden layer neurons, and 3 output neurons. The outputs of stage I constituted the inputs, and the target patterns were defined as 0 0 1 to 1 0 1 for five different groups. It took 73 iterations to converge.

To test the results, 50 different waveforms were picked up from the clusters provided by the single-layer clustering network. The groupings of the waveforms are given in Table 6.5. The waveforms were preprocessed and tested with the trained weights. The results are listed in Table 6.6. As can be seen, the networks assigned the signature numbers properly. Next, the whole file (after removing the undesirable waveforms) was tested with the stored weights. The spatial distributions of the different waveforms are shown in Figure 6.8. It is to be noted that only those waveforms which were successfully classified by the single-layered network were plotted for the purpose of comparing these two methods. A point to be noted here is that in this method, the user does not have any control of accepting or rejecting any given input pattern on the basis of the outputs as is possible in the single-layer case by using the threshold. A waveform which looks entirely different would still produce an acceptable set of outputs and there is no way to decide whether it is a genuine misclassification or the input pattern is entirely different from the trained patterns.

Table 6.5: Test waveforms for multi-layered networks and their target cluster numbers

Waveform no.	Assigned no.	Group no.
23 - 30	1 - 8	1
725 - 731 733 - 737	9 - 20	2
1029 - 1034 1060 - 1067	21 - 34	3
1594 - 1596 401 - 403	35 - 40	4
1642 - 1651	41 - 50	5

Table 6.6: Test results showing outputs of stage I and stage II neurons

Waveform	Outputs of stage I (decimal values)	Outputs of stage II
1 - 8	1 6 3 0 0 0 0	0 0 1
9 - 20	1 6 3 0 0 0 6	0 1 0
21 - 34	1 2 3 0 0 0 3	0 1 1
35 - 40	1 2 2 1 3 4 4	1 0 0
41 - 50	1 2 2 1 8 2 2	1 0 1

In the last example, waveforms recorded in Dolphin Union Straits of the Canadian Arctic in 1990 were used for extracting information about water quality. In all 200 waveforms were used for training purposes. The details of the training waveforms along with the desired final outputs are given in Table 6.7. Samples 25-176 were used in two segments. Samples 25-100 were used in segment 1 and samples 101-176 were used in segment 2. The size of the

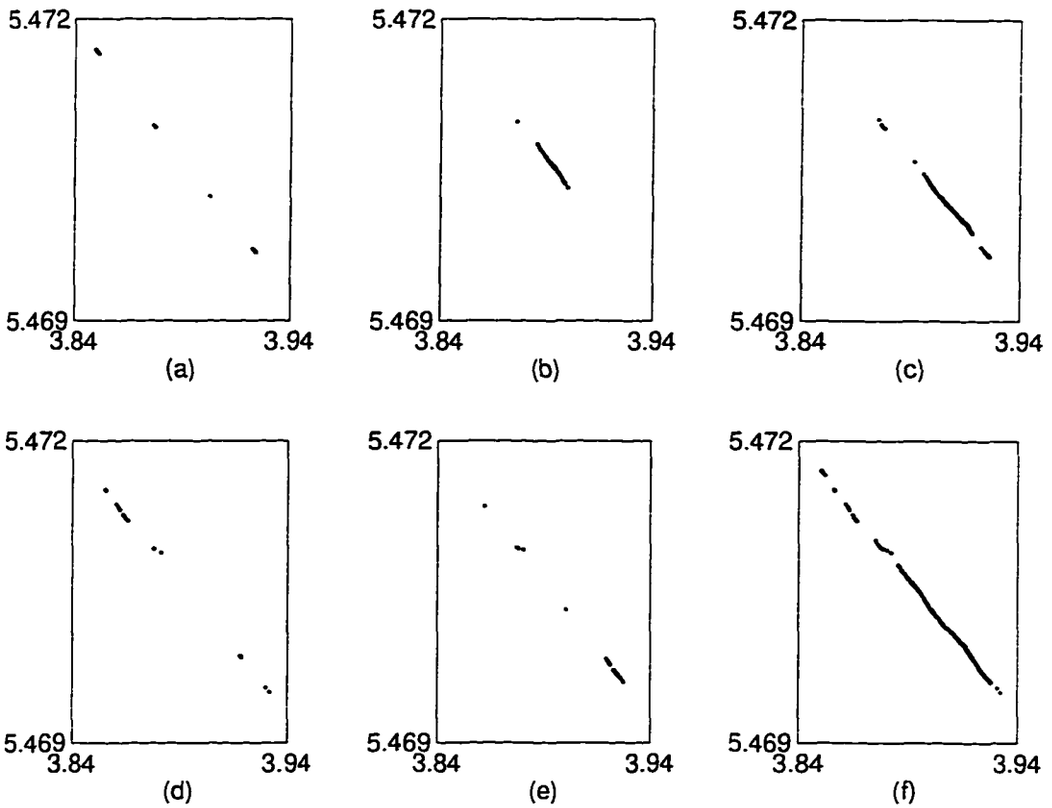


Figure 6.8: Spatial distribution of file Q0650022 obtained by multi-layered neural networks. (a)-(e) Distribution of each category of waveforms with respect to the same normalized GPS coordinates. (f) distribution of all waveforms together.

S matrices in each segment was fixed to 16 × 11. The details of the waveform characteristics which are the output of stage I for each of the segments are given in Table 6.8 and 6.9. A neural network of size 176-12-14 was used for segment 1 in stage I. It took 34 iterations to converge. A momentum term of 0.9 was used. For segment 2, a network of size 176-12-13 was used. It took 46 iterations with a momentum term of 0.9. Outputs from both the segments in stage I were then used as inputs for stage II. The network size used in stage II was 27-4-3 and it converged in 23 iterations with no momentum term. The network was tested with 99 waveforms, which were different from the training waveforms from the same files. It grouped all the waveforms properly. The waveforms are plotted in Figure 6.9.

It is well known that the results depend on the choice of initial weight matrices and the

Table 6.7: Number of different waveforms used for training and corresponding target output

Category name	No. of waveforms	Assigned output
clear water	40	1
slight turbid	40	2
clear, shallow	20	4
clear, medium shallow	20	5
turbid, medium shallow	20	6
turbid, deep	40	7

Table 6.8: Description of each field and number of assigned neurons in stage I for segment 1 for estimation of water quality

. Field no.	No. of neurons	Description
1	1	presence/absence of surface peak
2	2	amplitude of surface peak
3	2	peak position
4	1	presence/absence of bottom peak
5	3	amplitude of base of bottom peak
6	3	position of base of bottom peak
7	2	amplitude of bottom peak

order in which waveforms are presented for training. The results obtained by the two-stage multi-layered network were quite satisfactory. The choice of waveforms was rather arbitrary and some waveforms may have more representatives than waveforms from other groups.

The results obtained by the two methods for the classification of lidar waveforms repre-

Table 6.9: Description of each field and number of assigned neurons in stage I for segment 2 for estimation of water quality

Field no.	No. of neurons	Description
1	1	presence/absence of bottom peak
2	3	amplitude of base of bottom peak
3	5	position of base of bottom peak
4	4	amplitude of bottom peak

senting different milt contents are comparable. The main difference pertains to the networks themselves. The single-layered network is simple and takes much less time to train than the multi-layered one. However, the latter one takes into account all the important characteristics of the waveforms for training, which is not the case for the single-layer network. On top of that, the single-layered network can classify patterns which are linearly separable [72] through hyperplanes passing through the origin. One can possibly think about nonlinear mapping in these neurons as well. The reason that this network successfully clustered the waveforms is that we are looking for only 5 or 6 clusters in a huge 272-dimensional hyperspace and hence, the chance of their being linearly separable is very high. On the other hand, the multi-layered network can classify any patterns. Use of the results is two-fold. First, waveforms can be classified based on certain criteria for spatial distribution purposes. The other use is to group waveforms into various clusters so that particular signal processing algorithms can be used on them. Both the networks are incremental in the sense that waveforms can be added to them in future. The best use one can think of is to use all different types of waveforms for learning and use the trained network directly in the production phase.

6.6 Conclusions

Two different neural network schemes for the classification and extraction of information from waveforms recorded by the LARSEN 500 system have been proposed. The first scheme

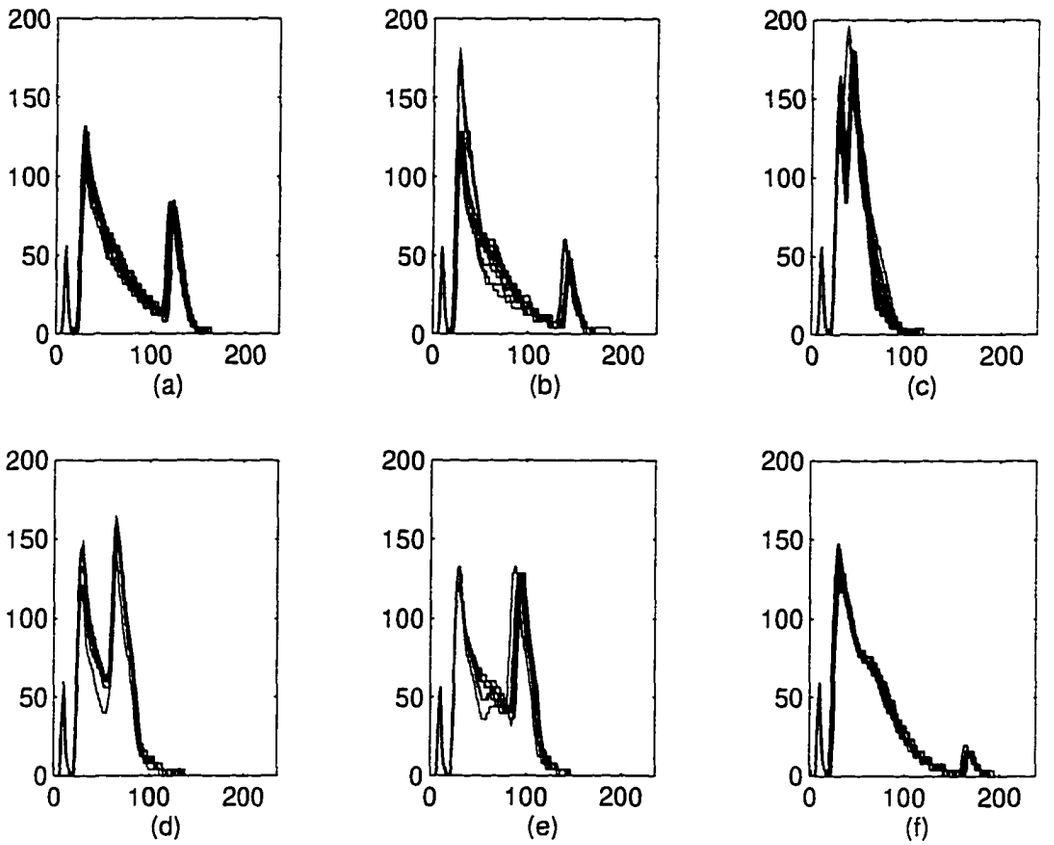


Figure 6.9: Grouping of waveforms representing different depth and water quality obtained from the Canadian Arctic region.

uses a single-layered linear neural network for classifying waveforms into different groups. Both unsupervised and supervised learning schemes have been included and applied for classification of waveforms for different milt content in the water. The spatial distribution of the milt is consistent with that obtained from the observed data. This method is much faster compared to the 2-stage multilayered feedforward neural network. The latter one transforms waveforms into characteristics in the first stage and uses this information in the second stage to generate final outputs. Actual waveform characteristics are used here for the final results. The performance of the two networks is comparable and the results are almost identical.

Chapter 7

Implementation

7.1 Introduction

The earliest implementations of neural networks used to be typically in terms of computer simulation. The simulation process is inherently sequential and the real power of neural networks lies in the use of parallel processing of information. One can exploit the usefulness of neural nets only if they are implemented in specialized hardware. There has been tremendous development in the area of VLSI in recent years and it is now possible to implement a whole network in silicon.

Electronic neural networks are based on simplified models of biological neurons. It is generally assumed that the computing power of neural systems, electronic or biological, arise from a collective behavior of large, highly interconnected, fine-grained networks [29]. In fact, an individual node does very little computation. It accepts inputs from all other connecting neurons and multiplies them by the respective weights. This sum of the weighted product is then passed through a decision-making or nonlinear function. Traditionally the weights heve been represented by resistors. If V_j is the output of neuron j and T_{ij} is the conductance between neurons i and j, then the sum of the weighted product is given by

$$I_i = \sum_{i=1 i \neq j} V_j T_{ij}$$

and the output of neuron i is given by

$$V_i = f(I_i)$$

As is apparent from the neural equations, there are three major components that one needs to implement electronic neural networks. First is the multiplier to multiply the input

voltage by the conductance, the second being an adder to compute the sum of all the weighted products. And the last one is a nonlinear amplifier. Other blocks involving the learning process may require additional components but more often than not, the learning is implemented off chip and learned weights are downloaded to the neural networks as and when needed [30].

Another important issue that one has to resolve is to decide whether to implement the circuits in analog or digital domain. The choice would certainly be influenced by the problem one intends to solve by the network. In the analog domain, one can think of current summation or charge summation. This provides an easy and compact way for adding a large number of values. If built digitally, accumulators are large and slow because the whole operation is spread over a large set of time steps. On the other hand, multipliers can be built very compactly in analog domain. However, analog circuits are smaller in size only when the resolution of computation is not too high [30]. The other drawback of analog circuits is that they are susceptible to process parameter variations. The noise immunity of analog circuits is much lower compared to that in their digital counterparts [33].

With reference to the networks that have been described in the previous chapters, we see that we need four-quadrant multipliers with one operand as a sine or cosine term. The other operand is either a filter coefficient or an error term. Filter coefficients are typically less than or equal to ± 1 and so are the trigonometric terms. The error values are high to start with and go down gradually with time. So an analog multiplier with somewhat lower dynamic range would solve our purpose. If the output of the multiplier is current, addition is even simpler. We can tie the outputs of all the multipliers together to achieve the summation. Of course, we have to be careful about the saturation problem, particularly when the network size is large. The decision-making function for our case is linear and a high-gain operational amplifier would do the job.

In the next section, the design of the multiplier and the amplifier will be considered. We will discuss the strategies that we have adopted to make these circuits suitable for our task. In the last section, a simple network will be implemented using these blocks which can be used to design FIR filters. Circuit development and simulations have been carried out using the CAD package Cadence.

7.2 Building blocks

7.2.1 Multiplier

Several analog multipliers based on the variable transconductance technique [57], the quarter square technique [55], and the square-algebraic identity [62] have been reported in the literature. The Gilbert multiplier is well established in bipolar technology. However, it is not possible to realize an MOS version of a Gilbert multiplier by simply substituting BJT's by MOS transistors. This is due to the fact that the output current in the MOS source-coupled differential pair depends nonlinearly on the bias current I_{ss} and input signal V_i [27]. Consequently, the linear range of such amplifiers is narrow. An analog multiplier based on the Gilbert cell has been described in [57]. Differential active attenuators have been used in the front end to increase signal swing capabilities.

A number of other multiplier circuits have been reported in [27] and [39]. Most of the designs have two drawbacks, namely, differential inputs and outputs. Some circuits need inputs of the form $V_c \pm V_1/2$ and $V_c \pm V_2/2$. For our case, generating the negatives of the filter coefficients or error values would require extra hardware. Ideally, we should be able to ground one of the differential inputs. One of the input terminals in the circuit presented in [57] can grounded and hence that circuit has been chosen for our task. The schematic of the MOS version of the Gilbert cell is given in Figure 7.1.

We would assume that all transistors are biased in saturation and they are perfectly matched so that the transconductances are related such that

$$K_1 = K_2 = K_3 = K_4 = K_a$$
$$K_5 = K_6 = K_b$$

Now the drain current through M_5 can be written as

$$I_5 = K_b (V_{2-} - V_T)^2$$

where it is assumed that all transistors have a threshold voltage of V_T. V_{2-} is the negative terminal for the input V_2. Similarly, I_6 is given by

$$I_6 = K_b (V_{2+} - V_T)^2$$

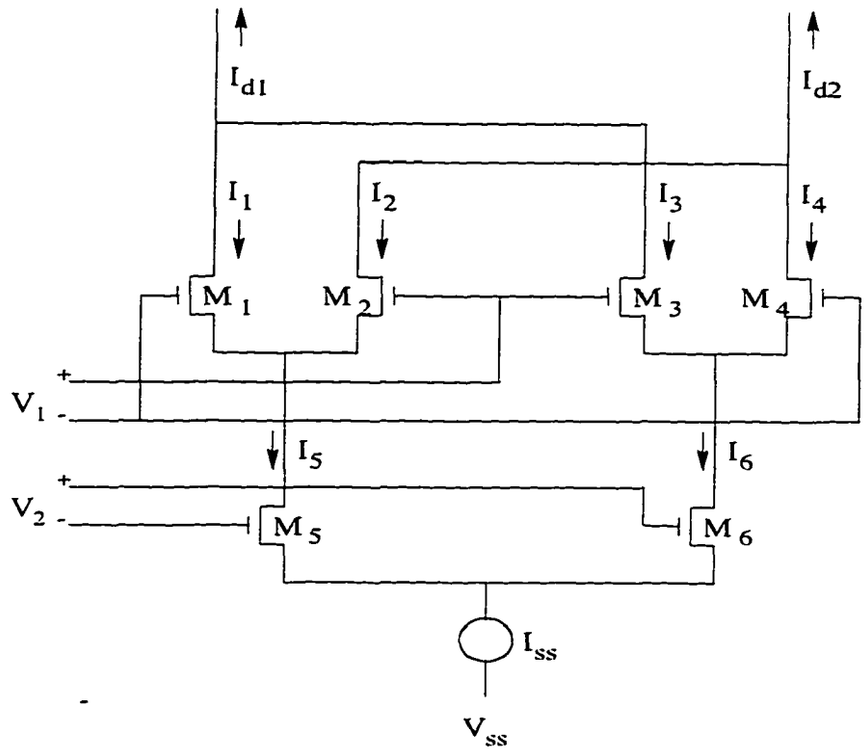


Figure 7.1: Schematic of the CMOS Gilbert cell.

Input voltage V_2 can be represented as

$$V_2 = V_{2+} - V_{2-} = \frac{1}{\sqrt{K_b}}\left(\sqrt{I_6} - \sqrt{I_5}\right) \tag{7.1}$$

Similarly, V_1 can be written as

$$V_1 = \frac{1}{\sqrt{K_a}}\left(\sqrt{I_2} - \sqrt{I_1}\right) = \frac{1}{\sqrt{K_a}}\left(\sqrt{I_3} - \sqrt{I_4}\right) \tag{7.2}$$

Now

$$\begin{aligned} I_1 + I_2 &= I_5 \\ I_3 + I_4 &= I_6 \end{aligned} \tag{7.3}$$

From (7.2) and (7.3), we get

$$\sqrt{I_2} - \sqrt{I_5 - I_2} = \sqrt{K_a}V_1 \tag{7.4}$$

Squaring (7.4), we have

$$I_2^2 - I_2 I_5 + \frac{\left(I_5 - K_a V_1^2\right)^2}{4} = 0 \tag{7.5}$$

Solving (7.5) for I_2, we get

$$I_2 = \frac{1}{2}\left[I_5 \pm \sqrt{K_a I_5} V_1 \sqrt{2 - \frac{K_a V_1^2}{I_5}}\right] \tag{7.6}$$

Similarly,

$$I_3 = \frac{1}{2}\left[I_6 \pm \sqrt{K_a I_6} V_1 \sqrt{2 - \frac{K_a V_1^2}{I_6}}\right] \tag{7.7}$$

The differential output current is given by

$$\begin{aligned} I_{od} &= I_{d2} - I_{d1} = -I_2 - I_4 + I_1 + I_4 \\ &= I_5 - I_6 - 2I_2 + 2I_3 \end{aligned} \tag{7.8}$$

Here we have used (7.3). Substituting (7.6) and (7.7) in (7.8) and noting that

$$\frac{K_a V_1^2}{2I_5, I_6} << 1$$

the differential output current can be written as

$$I_{od} = \sqrt{2K_a}\left(\sqrt{I_6} - \sqrt{I_5}\right) V_1 \tag{7.9}$$

Substituting (7.1) in (7.9), the output current can be expressed as

$$I_{od} = \sqrt{2K_a K_b} V_1 V_2 \tag{7.10}$$

As we can see from (7.10), the output current of the multiplier is a weighted product of the input voltages. By adding two resistors, a differential voltage can also be obtained. Input voltages V_1 and V_2 are passed through two attenuators before being connected to the multiplier cell. This enables one to use a higher input voltage swing. For biasing purpose, a level shifter is used at the output of the attenuator. The schematic diagram of the multiplier is shown in Figure 7.2.

7.2.2 Amplifier

Amplifiers are required to implement the neurons. For our case, the neurons are constant gain amplifiers. So an operational amplifier with a sufficiently high gain-bandwidth product would do the job. A number of schemes are reported in [4] and [27]. A moderately fast two-stage operational amplifier with a gain-bandwidth product of 1 MHz has been reported

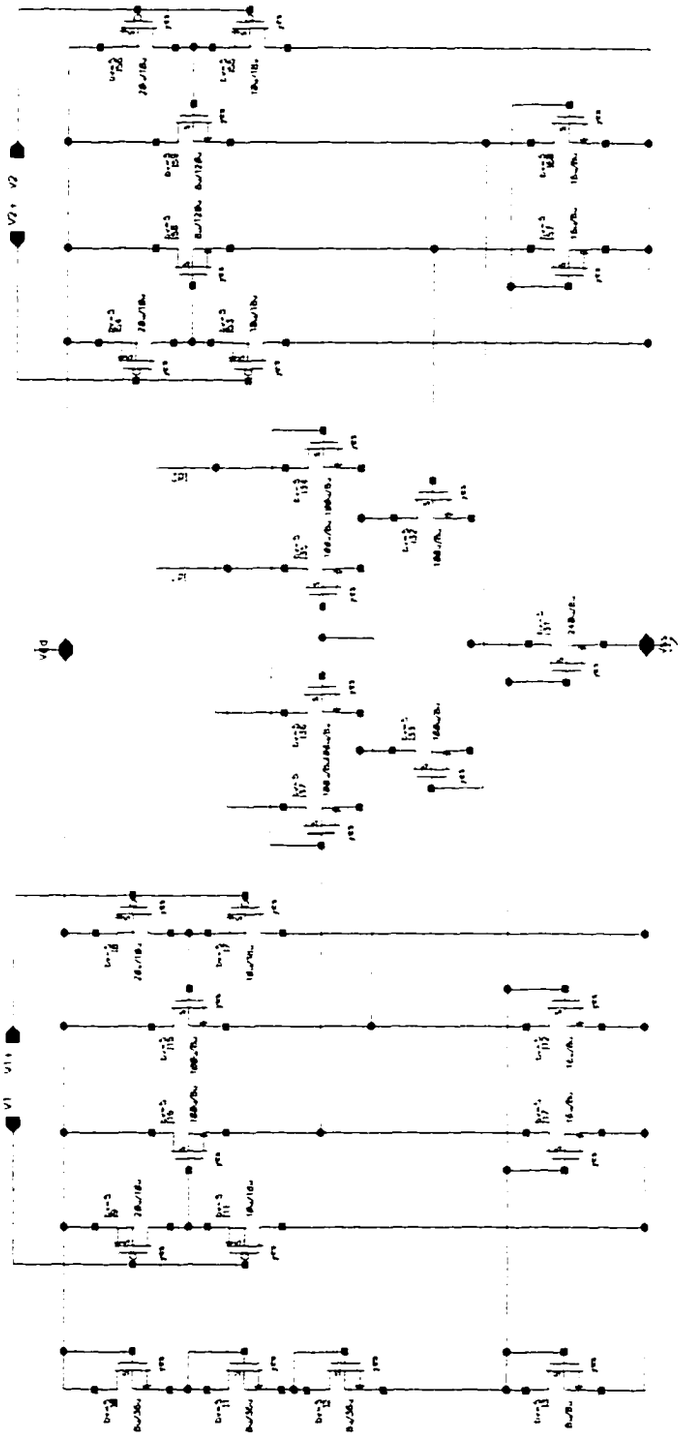


Figure 7.2: Schematic of the CMOS four-quadrant analog multiplier.

Table 7.1: Specifications of the operational amplifier

Open-loop gain > 4000	Load capacitor = 20 pF	$V_{ss} = -5$ V
Gain-bandwidth = 1 MHz	Slew rate > 2 V μs	O/p voltage range = ± 4 V
Input common range = ± 3 V	$V_{dd} = 5$ V	Power dissipation < 10 mW

in [4] and has been adopted for designing out network. A detailed design procedure can be found in [4]. The specifications of the amplifier are given in Table 7.1. The schematic diagram of the amplifier is shown in Figure 7.3.

7.3 Design strategies

Ideally our multiplier would produce a single ended current output which will be summed over a line and would be dumped into the capacitor C_g in parallel with r_g. The current equation then can be written as

$$i_{\text{total}} = \frac{u_i}{r_g} + C_g \frac{du_i}{dt}$$

In order to achieve this effect, all positive and negative terminals of the multipliers are tied to a positive bus and a negative bus, respectively, to achieve the current summation. Ideally, the multiplier should have 1000 Ω resistors from the output nodes to V_{dd}. However, when a number of multipliers are connected in parallel, the amount of current that would be flowing through the resistors would increase linearly. Consequently, the differential voltage that would be available would decrease steadily. In order to have a substantial differential output voltage, 700 Ω resistors are added from the buses to the supply voltage generating a differential voltage. As will be apparent in the next section, the output voltage from the multiplier is actually a scaled version of the product of the two voltages. Again, the total weighted product is also multiplied by a factor (10^{-5}) to avoid saturation. Part of this scaling is obtained by the multiplier and the rest is achieved by the voltage-to-current

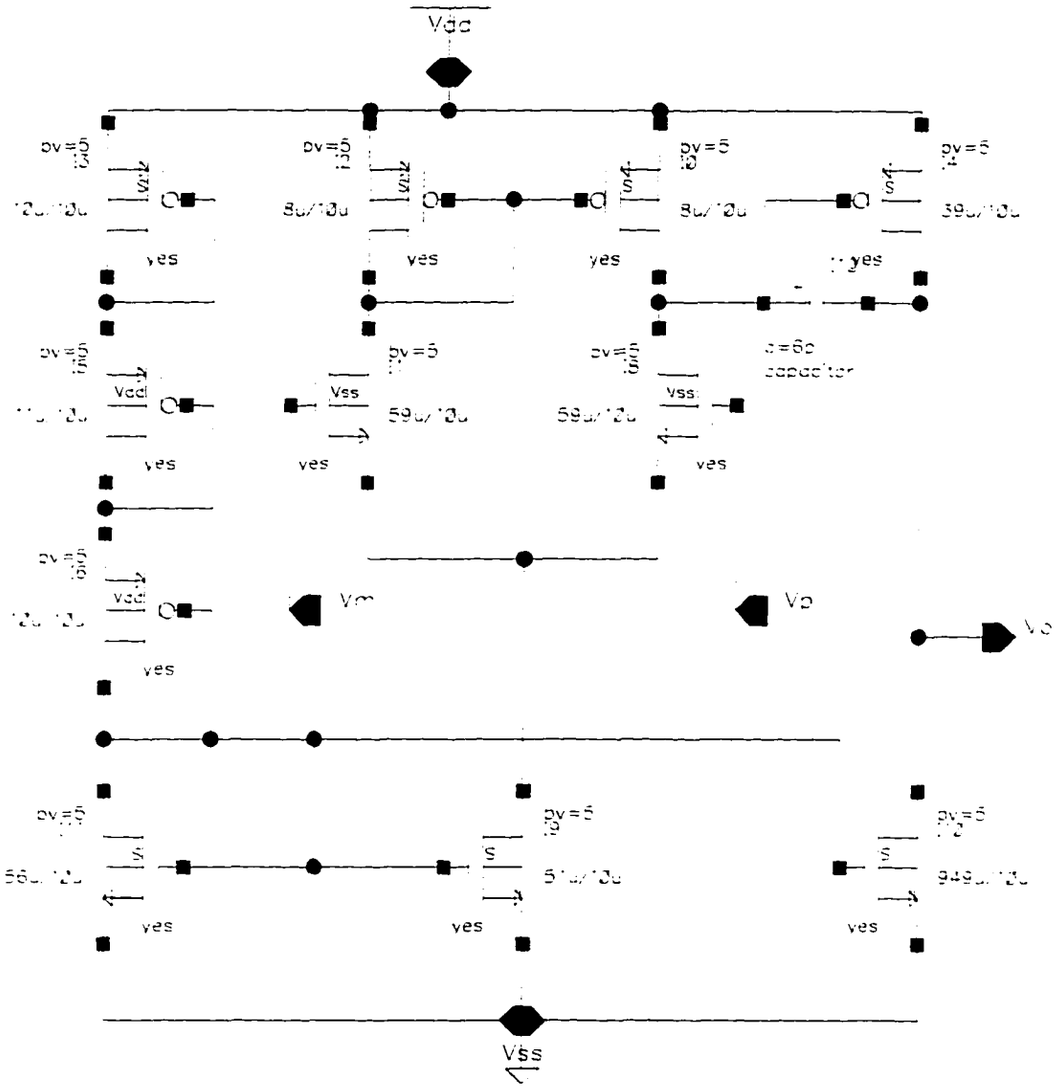


Figure 7.3: Schematic of the CMOS operational amplifier.

converter which has been used from the SPICE library. The current is then fed to capacitor C'_g in parallel with r_g. The voltage developed across C'_g and r_g is then amplified by the neurons. The proposed scheme is shown in Figure 7.4.

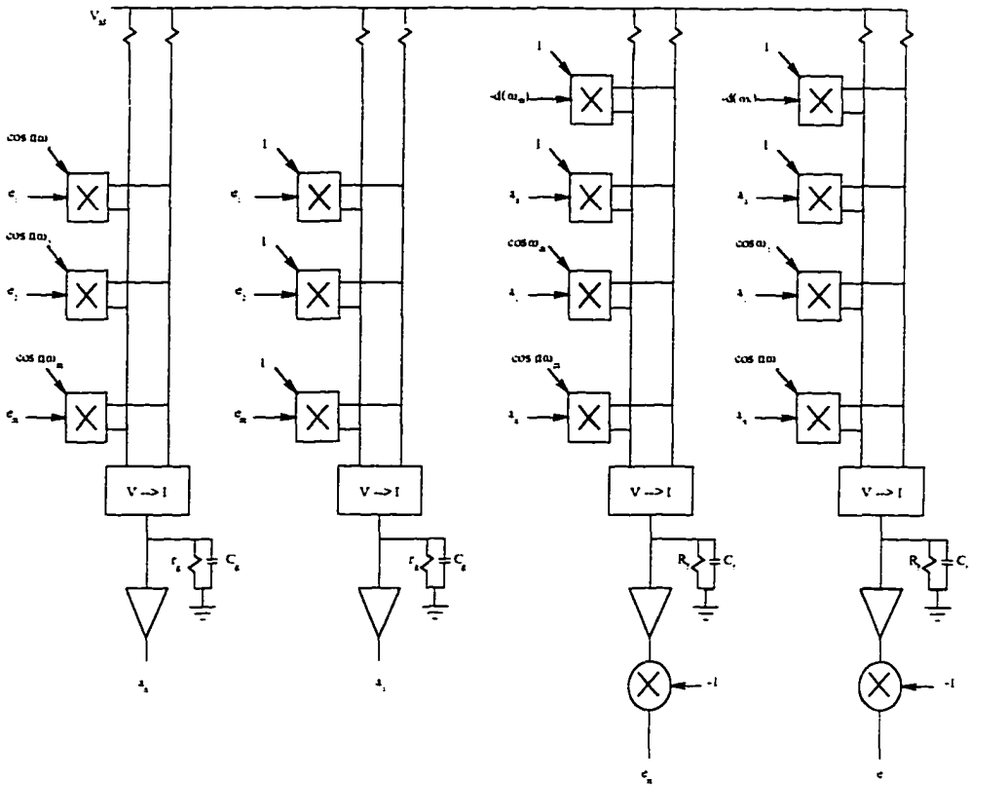


Figure 7.4: The schematic of the network.

7.4 Simulation

Since the analog simulators actually solve a set of differential equations to find out the solution of any circuit, the simulation time is really lengthy. In order to optimize the time and at the same time to show that our network does solve optimization problems, we have chosen a simple example. It is to design an FIR filter with $\omega_p = 0.4\pi$, $\omega_a = 0.6\pi$, and $\omega_s = 2\pi$. Only 6 sample points were chosen, three each in the passband and stopband and equally spaced. The filter length was set to 12 which requires 6 g neurons. The number of f neurons is 6 and the total number of multipliers is then 78, 36 in the connection matrix of g neurons and 42 for f neurons. The extra 6 are required to generate the current for the desired response. The schematic of the whole scheme is shown in Figure 7.4.

The transfer characteristic of the multiplier is shown in Figure 7.5. The differential voltage is obtained across a couple of 1000 Ω resistors. It can be seen that the multiplier output is linear and the output is scaled by a factor of 1:0.00831. In the network itself, we are stacking six multipliers for the g neurons and seven for the f neurons terminated with

700 Ω resistors. Since we are using only six (seven for f neurons) terms to generate the sum of the products, a scaling factor of 10^{-2} is used. The stack of multipliers itself attenuates the signal by a factor of 1:0.0053. So the voltage-to-current converter adds another scale factor of 30/16 to scale the total current by 10^{-2}. For the f neurons, the scaling factor for the voltage to current converter is 70/2.63e1.

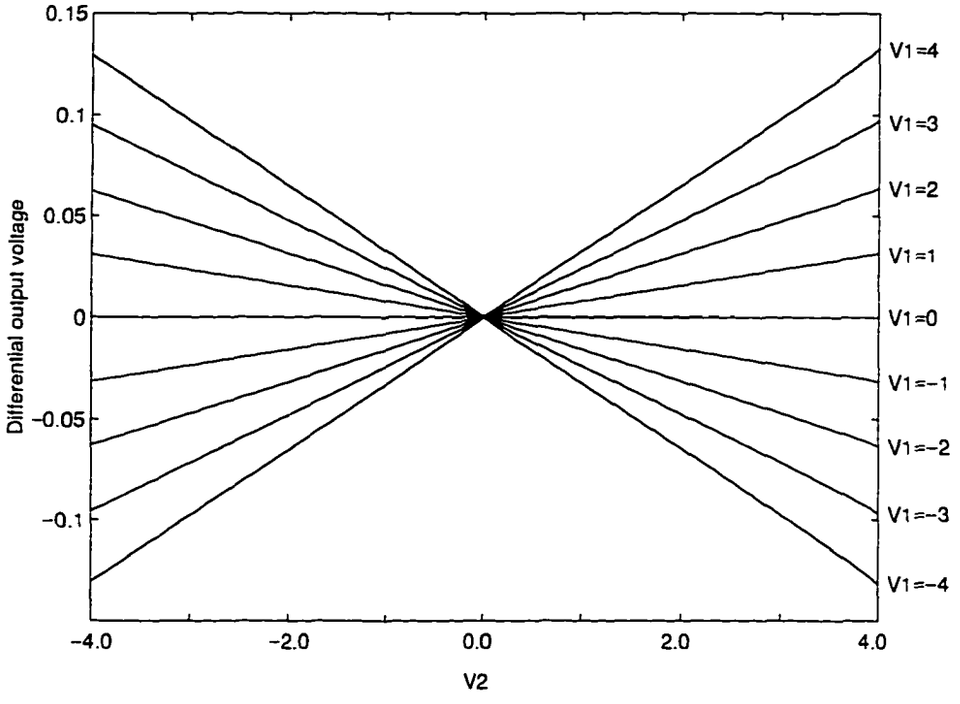


Figure 7.5: Differential output voltage of the multiplier for different values of V_1 when V_2 varies from -4 to +4 V.

As was mentioned in earlier chapters, the gain of the neurons was set to 2000. Since our scaling factor is 10^{-2}, gain is also reduced by the same factor, i.e., a gain of 2 is used. The reasons for doing so are as follows. Since our amplifier has a gain-bandwidth product of 1 MHz, a high gain means we need to use a number of stages or else will loose the speed. However, a gain of 2 can easily be obtained in one stage giving rise to a maximum speed of 2 μs. The other point that is noteworthy is that an f neuron computes the error between the desired response and the response of the filter being designed. The input voltage of the

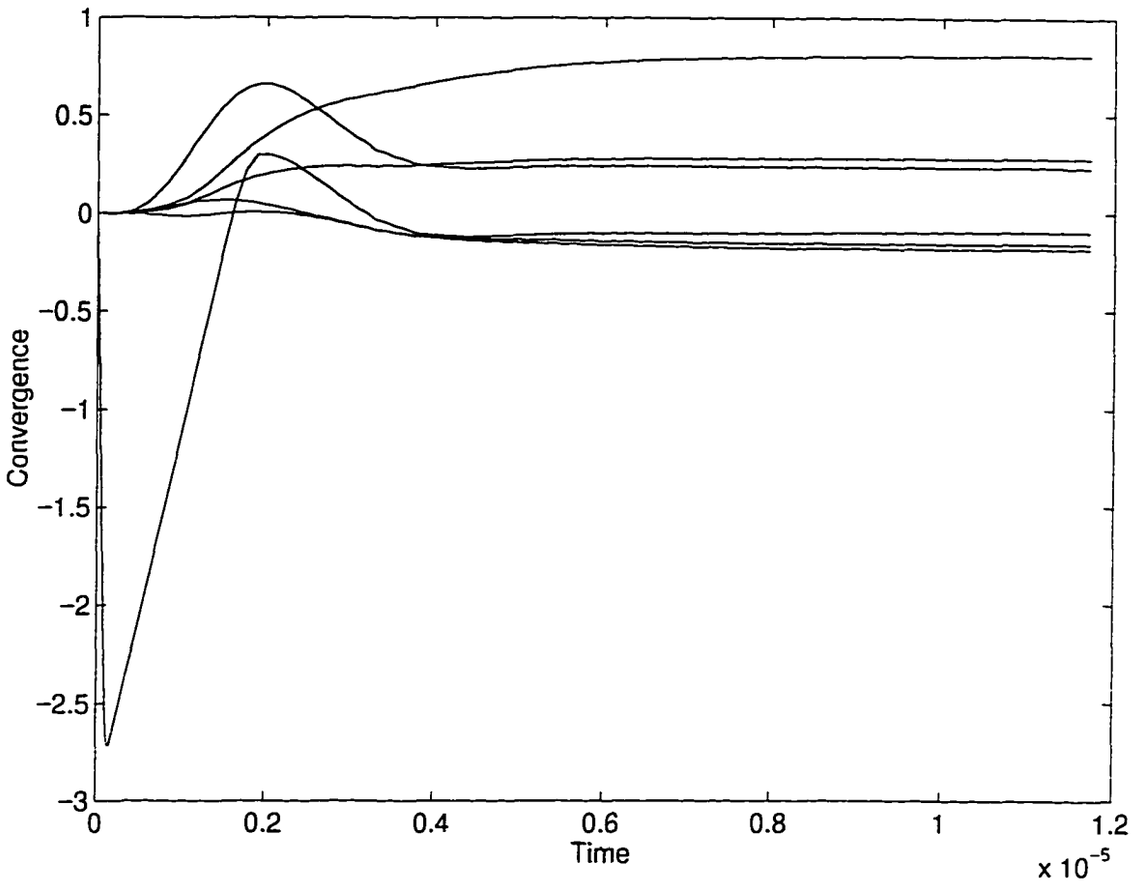


Figure 7.6: Convergence of the g neurons over time.

jth f neuron is given by

$$v_j = d(\omega_j) - \sum_{i=0}^{(N-1)/2} a_k \cos i\omega_j$$

Instead, in the network we compute the negative of the right hand side. Hence the f neurons employ a inverting amplifier. The g neurons employ a noninverting amplifier with the same gain. The network was simulated for 12 μs and the convergence of the filter coefficients are shown in Figure 7.6. The convergence of the error terms are shown in Figure 7.7. It can be seen that the convergence has been achieved in less than 10 μ secs. Next, from the outputs of the g neurons, the filter was designed. The amplitude response of the filter is shown in Figure 7.8. It can be seen that the designed filter meets the specifications. Because of the low filter order, the stopband attenuation is low.

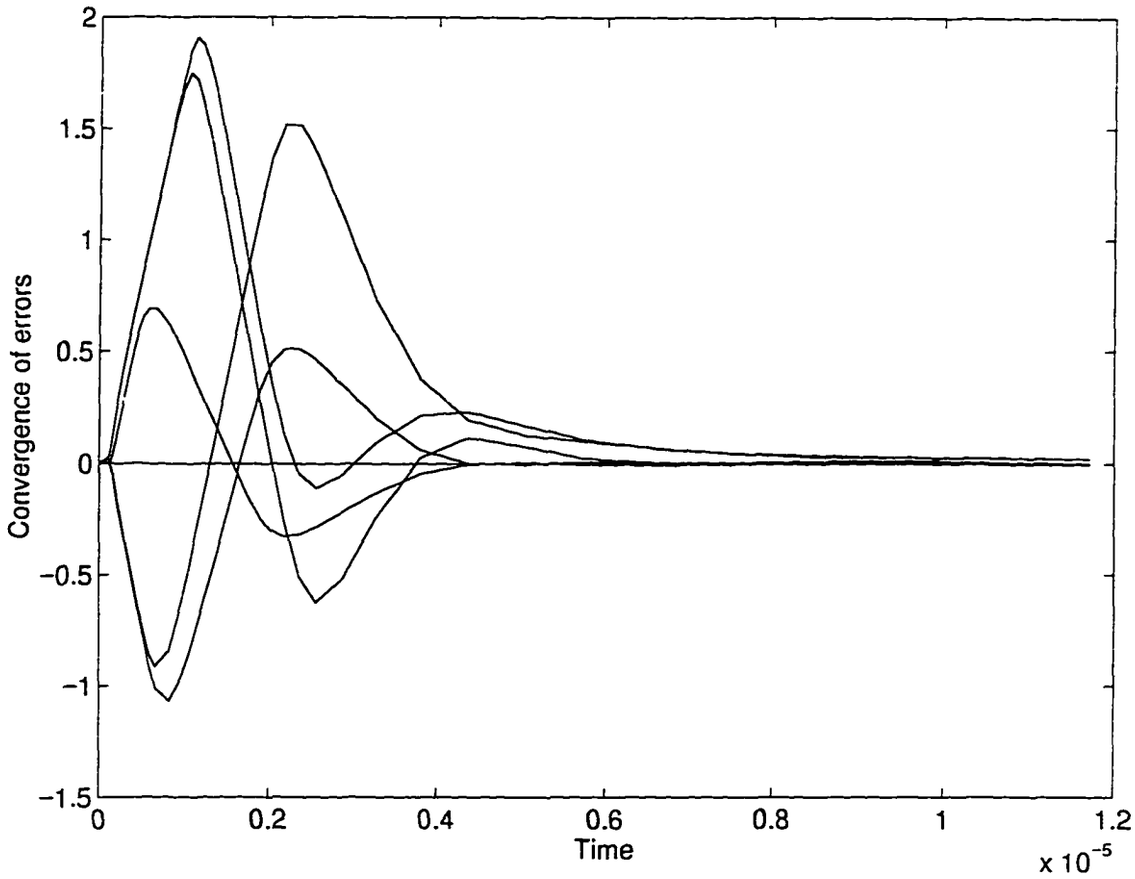


Figure 7.7: Convergence of the f neurons representing the errors.

7.5 Conclusion

The basic building blocks for designing the neural network have been identified. The specific requirements of each block have been addressed and existing circuits from the literature have been used to design the network. The network was then simulated to design an FIR filter of very low order and with a few sample points. Simulation results show that the proposed method is suitable for designing FIR filters and can be implemented in silicon very easily. The same network was simulated with randomly varied circuit parameters, e.g., amplifier gains, multiplier characteristics, and identical results were obtained in each of the cases.

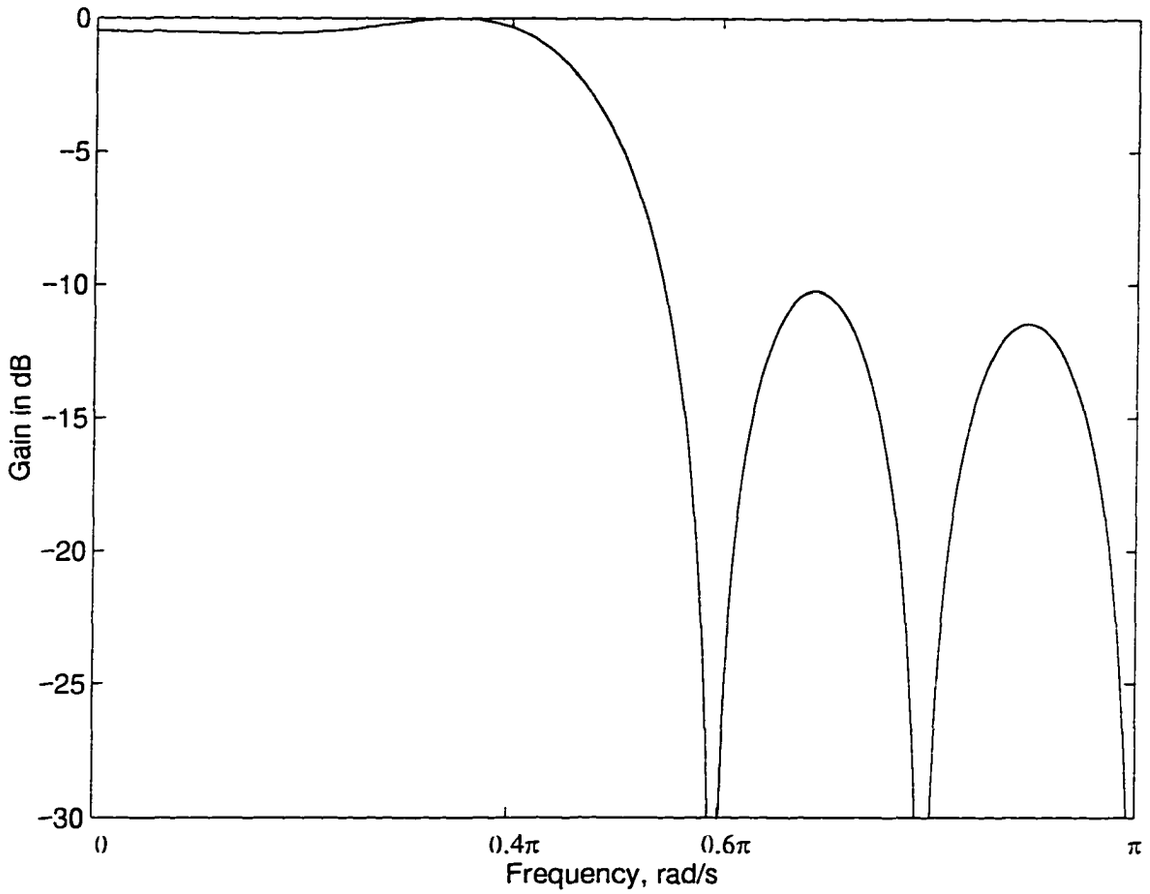


Figure 7.8: Amplitude response of the filter obtained by simulating the neural network.

The simulation and design have been carried out using the CAD package Cadence.

Chapter 8

Conclusions and Suggestions for Future Work

This chapter summarizes the conclusions of the thesis and outlines research directions that can be undertaken in the future.

8.1 Conclusions

The objective of this dissertation was to identify areas in signal processing where neural networks can be used successfully and to design those networks. Two major areas have been identified and proper neural networks have been developed. In the first area, studies have been done on the use of neural networks as a tool for the design of digital filters. In the second area, neural networks have been used successfully for the processing of bathymetric data.

In chapter 1, the purpose of this research work has been outlined and then the basics of neural networks have been introduced. The first part of the work is based on Hopfield's linear programming network and, consequently, the rest of the chapter was devoted to that and other related work.

In chapter 2, three schemes for designing 1-D FIR filters have been proposed. In the first part, a neural network was developed to design an FIR filter on the basis of a given amplitude response. In the following section, this work was extended to design filters on the basis of given amplitude and phase responses. In the last section, another network was developed for designing filters satisfying prescribed specifications. Detailed analysis of the

networks has been included and the convergence properties have been studied and suitable examples have been included. All three networks have been used for designing FIR filters of odd lengths and can easily be used for even lengths as well. The same networks can also be used for designing digital differentiators and Hilbert transformers.

In chapter 3, two networks for designing 2-D FIR filters have been developed. The first network can be used for designing filters on the basis of a given amplitude response and the second can be used to achieve prescribed specification. Only filters with quadrantal symmetry and linear phase have been considered. However, the work can be easily extended to other symmetries.

In chapter 4, two schemes for designing 1-D IIR filters were proposed. In the first scheme, an IIR filter was designed using a neural network in the magnitude-squared domain. In the other scheme, a neural network was developed to design IIR filters on the basis of arbitrary amplitude and phase responses.

In chapter 5, a new updating algorithm for designing equiripple FIR filters has been developed. The basic network is the same as that used in chapter 2 for the design of 1-D FIR filters. It was then shown that with some extra blocks, the resulting network can be used for the design of 1-D equiripple FIR filters, differentiators and Hilbert transformers.

Chapter 6 deals with the direct use of neural networks for signal processing. Two different neural networks have been proposed to classify lidar waveforms for various tasks. The first network is a single-layer network using Kohonen's winner-take-all learning algorithm. Both supervised and unsupervised learning schemes have been used. The performance of the supervised learning scheme has further been extended by changing the weights of both the wrong winning neuron and the right one which should have won but did not. The idea of using a threshold value has also been introduced to give the user a choice of accepting or rejecting a waveform in the recall mode. In the second scheme, a two-stage three-layer neural network has been developed. In the first stage, the input waveforms are first transformed into a set of waveform characteristics and these results are used in the second stage to assign a signature number to the waveforms. This method can also be used for extracting various ocean parameters from the waveforms. Both the networks were used for classifying waveforms representing different milt contents.

In chapter 7, three basic building blocks for implementing the neural network in silicon have been identified. Existing circuits from the literatures have been used and a simple

network was constructed to design a 1-D FIR filter. The circuit was simulated and the result shows that the network can be easily implemented in analog VLSI. Analog CMOS circuits were chosen because they are more compact than digital ones. Because of the connectivities, the network was found to be very much tolerant to parameter variations as one would normally expect in analog circuits.

8.2 Suggestions for future work

The research pursued in this thesis can be extended in several directions. One-dimensional FIR filters were designed using direct structures. One can design networks based on transfer functions in parallel structures. The same idea can also be extended for 2-D FIR filters. The design of equiripple 2-D FIR filters is another area that requires further study. On the other hand, for the IIR filter case, a number of strategies can be adopted to realize a stable filter. Out of the two schemes that have been proposed here, no constraints have been put on the position of the poles. In the design based on the magnitude-squared domain, poles occur in pairs, poles inside the unit circle and their mirror images with respect to the unit circle. Rejecting the poles outside the unit circle results in a stable filter. On the other hand, for the design of IIR filters on the basis of given amplitude and phase responses, there is no control over the position of the poles. Instead of working on the transfer function in the direct form, we can possibly use a cascade or parallel structure and put a constraint on the filter coefficients so that the poles would always lie within the unit circle. One can even consider working on the transfer function with only poles and put a constraint so that they always lie within the unit circle. Nothing has been done on the design of equiripple 1-D IIR filters or 2-D IIR filters and that certainly leaves ample scope in these areas too.

Another area that needs further investigation is the use of different kinds of cost functions. In our work, we have used least-squares cost functions for the design of the different kinds of filters. In [26], some non-LS cost functions have been discussed in conjunction with adaptive FIR filtering. Research involving different cost functions can be carried out and compared to find out the best one. Similarly, the f neurons we used are perfectly linear although they are not in Hopfield's linear programming network [65]. The effect of nonlinearity can also be investigated.

The networks developed for classifying lidar waveforms representing varied milt contents

can be used for the detection of fish. The presence of fish is represented by a small bulge and the nature and position of this bulge determines the species and location of the fish. A better preprocessing scheme for transforming lidar waveforms into binary streams can be developed for detecting these small variations in the waveforms. The existing networks can be modified, if required, for classifying waveforms on the basis of sea-bed structures.

Last but not the least is the VLSI implementation. As mentioned earlier, we have used components that already exist in the literature. The multiplier used is very fast and can be scaled properly for existing technologies, e.g., 0.8 μ or 1.2 μ. The amplifier is slow and one can consider designing a faster version. There is also scope in developing standard cells for these components which would be very useful for the implementation of the whole network in silicon.

References

[1] Aarts E. and Korst J., *Simulated Annealing and Boltzman Machines*, John Wiley, 1989, Chapter 7, pp. 117-128.

[2] Algazi V. R., Suk M., and Rim C.-S., "Design of almost minimax FIR filters in one and two dimensions by WLS techniques," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 6, June 1986, pp. 590-596.

[3] Alkhairy A., "On IIR filter design," *Proceedings of IEEE International Symposium on Circuits and Systems*, Seattle, USA, April-May 1995, pp 862-864.

[4] Allen P. E. and Holberg D. R., *CMOS Analog Circuit Design*, Holt, Rinehart, and Winston, 1987.

[5] Antoniou A., *Digital Filters: Analysis, Design, and Applications*, 2nd edition, McGraw-Hill, 1993.

[6] Avitabile G., Forti M., Manetti S., and Marini M., "On a class of nonsymmetrical neural networks with application to ADC," *IEEE Transactions on Circuits and Systems*, vol. 38, February 1991, pp. 202-209.

[7] Bhattacharya D., Pillai R., and Antoniou A., "Waveform classification and information extraction from lidar data by neural networks," submitted to *IEEE Transactions on Geoscience and Remote Sensing*.

[8] Bhattacharya D. and Antoniou A., "Real-time design of FIR filters by feedback neural networks," to be published in *IEEE Signal Processing Letters*.

[9] Bhattacharya D. and Antoniou A., "Design of IIR filters with arbitrary amplitude and phase specifications by feedback neural networks," to be presented at the *IEEE International Symposium on Circuits and Systems*, Atlanta, Georgia, USA, May 1996.

[10] Bhattacharya D., Pillai R., and Antoniou A., "Classification of lidar waveforms by neural networks," to be presented at the *IEEE International Symposium on Circuits and Systems*, Atlanta, Georgia, USA, May 1996.

[11] Bhattacharya D. and Antoniou A., "Design of equiripple digital differentiators and Hilbert transformers by feedback neural networks," *Proceedings of European Conference on Circuit Theory and Designs*, Istanbul, Turkey, August 1995, pp. 203-206.

[12] Bhattacharya D. and Antoniou A., "Design of equiripple linear phase FIR filters by feedback neural networks," presented at *38th IEEE Midewest Conference*, Brazil 1995.

[13] Bhattacharya D. and Antoniou A., "Design of digital differentiators and Hilbert transformers by feedback neural networks," *Proceedings of IEEE Pacific Rim Conference*, Victoria, BC, May 1995, pp. 389-392.

[14] Bhattacharya D. and Antoniou A., "Design of 2-D FIR filters by feedback neural networks," *Proceedings of IEEE International Symposium on Circuits and Systems*, Seattle, USA, April-May 1995, pp. 1297-1300.

[15] Bhattacharya D. and Antoniou A., "Real-time synthesis of FIR filters in frequency domain by feedback neural networks," *Proceedings of 37th IEEE Midwest Conference*, Lafayette, Louisiana, August 1994, pp. 1069-1072.

[16] Burr D., "Experiments on neural net recognition of spoken and written text," *IEEE Transactions on ASSP*, no. 7, vol. 36, 1988, pp. 1162-1168.

[17] Chen J., Shanblatt M. A., and Maa C.-Y., "Improved neural networks for linear and nonlinear programming," *International Journal of Neural Systems*, vol. 2, 1992, pp. 331-339.

[18] Cichocki A. and Unbehauen R., "Simplified neural networks for solving linear least squares and total least squares problems in real time," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, November 1994, pp. 910-923.

[19] Crosara S. and Mian G., "A note on the design of IIR filters by the differential-correction algorithm," *IEEE Transactions on Circuits and Systems*, vol. 30, no. 12, December 1983, pp. 898-903.

[20] Deitz W. E., Kiech E. L., and Ali M., "Jet and rocket engine fault diagnosis in real time," *Journal of Neural Network Computing*, Summer 1989, pp. 5-18.

[21] Dong G. and Ling X., "Combinatorial neural net for adaptive filtering," *Proceedings of IEEE International Symposium on Circuits and Systems*, June 1991, pp. 1392-1395.

[22] Forti M., Manetti S., and Marini M., "Neural network for adaptive FIR filtering," *Electronics Letters*, vol. 26, July 1990, pp. 1018-1019.

[23] Forti M., Liberatore A., and Manetti S., "On a new class of neural adaptive FIR filters," *Proceedings of IEEE International Symposium on Circuits and Systems*, June 1991, pp. 1400-1403.

[24] Forti M., Liberatore A., Minetti S., and Marini M., "Global asymtotic stability for a class of neural networks," *Proceedings of IEEE International Symposium on Circuits and Systems*, 1993, pp. 2580-2583.

[25] Forti M., Manetti S., and Marini M., "A condition for global convergence of a class of symmetric neural circuits," *IEEE Transactions on Circuits and Systems - I*, vol. 39, no. 6, June 1992, pp. 480-483.

[26] Forti M., Manetti S., and Marini M., "Neural networks for optimization of non-quadratic cost functions with applications to adaptive signal processing," *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 6, May 1992, pp. 2909-2912.

[27] Geiger R. L., Allen P. E., and Strader N. R., *VLSI Design Techniques for Analog and Digital Circuits*, McGraw-Hill, 1990.

[28] Gorman R. and Sejnowski T., "Learned classification of sonar targets using a massively parallel network," *IEEE Transactions on ASSP*, vol. 36, no. 7, 1988, pp. 1135-1140.

[29] Graf H. P. and Jackel L. D., "Analog electronic neural network circuits," *IEEE Circuits and Devices Magazine*, July 1989, pp. 44-49 and 55.

[30] Graf H. P., Sackinger E., and Jackel A. D., "Recent developments of electronic neural nets in North America," *Journal of VLSI Signal Processing*, no. 5, 1993, pp. 19-31.

[31] Guenther G. C., *Airborne Laser Hydrography: System Design and Performance Factors*, NOAA Professional Paper Series, 1985.

[32] Hecht-Nielsen R., "Applications of counterpropagation networks," *Neural Networks*, vol. 1, 1988, pp. 131-139.

[33] Hirai Y., "Recent VLSI neural networks in Japan," *Journal of VLSI Signal Processing*, no. 6, 1993, pp. 7-18.

[34] Hopfield J. J., "Artificial neural networks," *IEEE Circuits and Devices Magazine*, September 1988, pp 3-10.

[35] Hopfield J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proc. Natl. Acad. Sci. USA*, vol. 79, April 1982, pp. 2554-2558.

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

[37] Hopfield J. J. and Tank D. W., "Neural compostion of decisions optimization problems," *Biol. Cybern.*, vol. 52, 1985, pp. 141-152.

[38] Hush D. R. and Horne B. G., "Progress in supervised neural networks, what's new since Lippman?," *IEEE Signal Processing Magazine*, January 1993, pp. 8-39.

[39] Ismail M. and Fiez T., *Analog VLSI: Signal and Information Processing*, McGraw-Hill, 1994.

[40] Kennedy M. P. and Chua L. O., "Neural Networks for Nonlinear Programming," *IEEE Transactions on Circuits and Systems*, vol. 35, May 1988, pp. 554-562.

[41] Kohonen T., "The 'neural' phonetic typewritter," *IEEE Computer*, vol. 23, no. 3, 1988, pp. 11-22.

[42] Lee Bang W. and Sheu Bing J., "An investigation of local minima of Hopfield network for optimization circuits," *Proceedings of the IEEE International Conference on Neural Networks*, San Diego, pp. I-45–I-51.

[43] Lennon S. and Ambikairajah E., "A two-layer Kohonen neural network using a cochlear models as front-end processor for a speech recognition system," *Neural Networks for Signal Processing II. Proceedings of the 1992 IEEE-SP Workshop*, pp. 139-148.

[44] Lim Y. C., Lee J.-H., and Yang R.-H., "A weighted least squares algorithm for quasi equiripple FIR and IIR digital filter design," *IEEE Transactions on Signal Processing*, vol. 40, no. 3, March 1992, pp. 551-558.

[45] Lippmann R. P., "An introduction to computing with neural nets," *IEEE ASSP Magazine*, April 1987, pp. 4-22.

[46] Lu W.-S. and Antoniou A., *Two-Dimensional Digital Filters*, Marcel Dekker, New York, 1992.

[47] Maa C.-Y. and Shanblan M. A., "Improved Linear Programming Neural Networks," *Proceedings of the 32nd Midwest Symposium on Circuits and Systems*, August 1989, pp. 748-751.

[48] Martinelli G. and Perfetti R., "Neural network for real time synthesis of FIR filters", *Electronics Letters*, vol. 25, no. 17, August 1989, pp. 1199-1200.

[49] Michel A. N., Farrell J. A., and Porod W., "Qualitative analysis of neural networks," *IEEE Transactions on Circuits and Systems*, vol. 36, February 1989, pp. 229-243.

[50] Narendra K. S. and Parthasarathy K., "Identification and control of dynamical sytems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, March 1990, pp. 4-27.

[51] Narendra K. S. and Parthasarathy K., "Gradient methods for the optimization of dynamical systems containing neural networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, March 1991, pp. 252-262.

[52] Oppenheim A. V. and Schafer R. W., *Discrete-Time Signal Processing*, Prentice-Hall, 1992.

[53] Park S., "A geometric analysis of Hopfield neural network for optimizations", *Proceedings of 1989 Southeastcon*, pp 58-62.

[54] Park S., "Signal space interpretations of Hopfield neural networks for optimization," *Proceedings of IEEE International Symposium on Circuits and Systems*, Portland, Oregon, May 1989, pp. 2181-2184.

[55] Peña-Finol J. S. and Connelly J. A., "A MOS four-quadrant analog multiplier using the quarter-square technique," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 6, December 1987, pp. 1064-1073.

[56] Proakis J. G. and Manolakis D. G., *Introduction to Digital Signal Processing*, Macmillan, N.Y., 1988.

[57] Qin S.-C. and Geiger R. L., "A ±5-V CMOS analog multiplier," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 6, December 1987, pp. 1143-1146.

[58] Rabiner L. R., Graham N. and Helms H., "Linear programming design of IIR digital filters with arbitrary magnitude function," *IEEE Transactions on ASSP*, vol. 22, no. 2, April 1974, pp. 117-123.

[59] Rumelhart D. E., McClelland J. L., and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Macrostructure of Cognition, Volume I: Foundations*, MIT Press, Massachusetts, 1986.

[60] Salam F. M., Wang Y., and Choi M.-R., "On the analysis of dynamic feedback neural nets," *IEEE Transactions on Circuits and Systems*, vol. 38, February 1991, pp. 196-201.

[61] Schweizer L., Parladori G., and Sicuranza G. L., "Globally trained neural network architecture for image compression," *Neural Networks for Signal Processing II, Proceedings of the 1992 IEEE-SP Workshop*, pp. 289-295.

[62] Song H.-S. and Kim C.-K., "An MOS four-quadrant analog multiplier using simple two-input squaring circuits with source followers," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 3, June 1990, pp. 841-847.

[63] Sunder S. and Ramachandran V., "Design of equiripple nonrecursive digital differentiators and Hilbert transformers using a weighted least-squares technique," *IEEE Transactions on Signal Processing*, vol. 42, no. 9, September 1994, pp. 2504-2509.

[64] Tagliarini G. A., Christ F., and Page E. W., "Optimization using neural networks," *IEEE Transactions on Computers*, vol. 40, December 1991, pp. 1347-1358.

[65] Tank D. W. and Hopfield J., "Simple Neural Optimization Networks : An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit", *IEEE Transactions on Circuits and Systems*, vol. CAS-33, May 1986, pp. 533-541.

[66] Vazquez A. R.,Castro R. D., Rueda A., Huertas J. L., and Sinencio E.S., "Nonlinear Switched-Capacitor "Neural" Networks for Optimization Problems," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 3, March 1990, pp. 384-398.

[67] Wong H., "Signal Processing Techniques for Airborne Laser Bathymetry," Ph.D. Thesis, University of Victoria, 1993.

[68] Wong H. and Antoniou A., "Characterization and decomposition of waveforms for LARSEN 500 airborne system," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 29, no. 6, November 1991, pp. 912-921.

[69] Wong H. and Antoniou A., "A multiple neural-network approach for classification of waveforms for LARSEN 500 airborne system," *Proceedings of First International Remote Sensing Conference and Exhibition*, Strasbourg, France, September 1994.

[70] Wong H. and Antoniou A., "One-dimensional signal processing techniques for airborne laser bathymetry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 32, January 1994, pp. 35-46.

[71] Wong H. and Antoniou A., "Reconstruction and enhancement of sea-bed topography by using 2-D signal processing," *Proceedings of the Oceans'93 Conference*, vol. 2, October 1993, pp. 375-380.

[72] Zurada J. M., *Introduction to Artificial Neural Systems*, West Publishing Company, 1992.