5 Biomedical signal filtering

Chapter 5
Biomedical signal filtering

5.1 Introduction

The electrocardiogram (ECG) is a graphic recording of the electrical potentials produced by cardiac tissue. The heart is unique among the muscles of the body in that it possesses the properties of automatic impulse formation and rhythmic contraction.

Traditionally, a large number of algorithms for noise reduction in ECG's use either spatial or temporal averaging techniques [1]. Temporal averaging method requires a large number of beats or frames for effective noise reduction. Moreover, the averaging causes considerable errors particularly when the time alignment of beats is not accurately known, or especially when premature beats are present [2]. The main drawback of spatial averaging is the physical limitations to placement of a large number of electrodes at the same region.

Chapter provides a comprehensive study of the work done by the researchers using conventional techniques for the signal filtering. The ANN models for the signal filtering are developed and their performance is compared with conventional techniques

5.2 Classical Methods

The section gives overview of proposed techniques for detecting and filtering artifacts generated due to various sources.

5.2.1 Power line interference

Power line interference is the most common of all unwanted artifacts [3]. It causes problems in recording ECG. The cause of this interference may be magnetic induction, displacement current in lead or patient's body. It may also be due to imperfection in the equipment. Proper grounding or using twisted wire pairs can minimize them. Apart from the amplifier design methods, various signal processing procedures for its suppression or elimination were proposed, mostly for the electrocardiogram signal [4-8]. Adaptive noise cancellation techniques are also proposed [9].

A dynamic interference subtraction procedure totally preserves the original signal frequency components [10-13]. Short signal segments are investigated for linearity,

by comparison of differences between samples taken at 2π intervals of the power line interferences, to a selected threshold value M.

When linear segment is detected, averaging the samples in one interference period, thus obtaining interference-free data. The formula of the averaging (comb) filter is:

$$Y_{(n+1)/2} = \frac{\sum_{i=1}^{n} Xi}{n} \qquad \text{.....(5.1)}$$

Where x is the signal before filtration, Y is the filtered one and n is the number of samples in one interference period. In case when the sampling rate is an even multiple of the interference frequency, following correction is introduced [8]:

$$Y_{n/2} = \frac{\sum_{i=1}^{n} Xi}{n} - \frac{X_{n+1}^{2n} X_1}{2n} \qquad \text{.....(5.2)}$$

Once the interference-free data is obtained, the interference amplitudes are calculated for every sample by subtraction of the filtered signal data from the original ones.

In nonlinear segments (QRS complex and some turning points in sharp T waves of high amplitude) the interference can be eliminated by sample-by-sample subtraction of the interference amplitudes computed from corresponding nearest linear segments.

The threshold value M has been chosen as a result of empirical tests [7-10] at 150 μV.

The subtraction procedure was proven to be very efficient, even with changing amplitude and frequency of the interference [12, 13]. Its performance is slightly reduced in cases of continuous well-expressed tremor (electrocardiogram (EMG) interference). This is due to the decreased number of linear segments detected. It results:

(i) In preservation of some of the tremor noise as a part of the 'pure' electrocardiogram (ECG) signal in nonlinear segments, where the subtraction is applied, and
(ii) in relatively infrequent re-calculation of the interference amplitude are present, where these corrections do not fully correspond to the already changed signal.

The purpose of the work in [3] was to improve the subtraction method in order to enhance its efficiency in the presence of non-powerline interference noise and to widen its application to other than ECG biological signals, for example impedance-cardiogram, plethysmogam, electrocephalograph (EEG), etc.

This was achieved by dynamic adaptation of the linearity criterion as a compromise between two contradictory requirements; (i) noise elimination (reduction) and (ii) preservation of the original signal components.

5.2.2 Singular Vale Decomposition

The level of contamination due to artifacts may be such that the ECG signals are completely hidden. A singular value decomposition technique decomposes signal in to two time-orthogonal signal subspaces. One subspace will contain ECG component while artifacts such as base line wander or EMG are in other subspace.

5.2.2.1 Orthogonalization Method

An orthogonalization method can be used to eliminate unwanted signal components in standard 12-lead exercise ECGs [14]. The algorithm developed adapts itself to variation in ECG or unwanted signal components due to artifacts [14].

The cardiac signal morphologies in standard 12-lead ECG signals are highly correlated. Eight channels out of 12 are independent and orthogonalization is applied to these eight channels. They are named VI-V6.

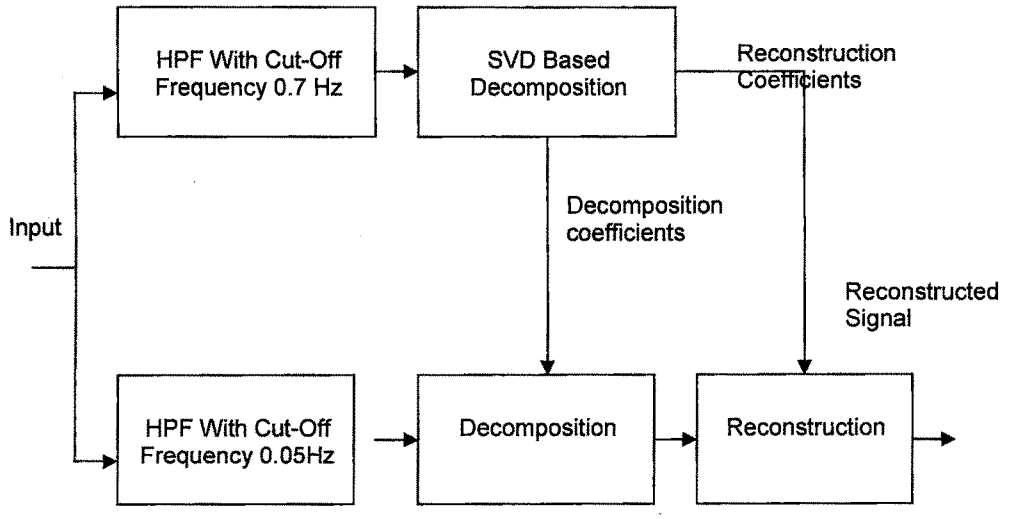

Figure 5.1 Block Diagram of the Process

M is data matrix with each row corresponds to input channel and columns correspond to successive sampling instances in time. The algorithm is based on the online approximation of SVD of the data matrix $M \in R^{pxn}$. SVD of M is [15]:

$$M=U\Sigma V^T \quad \text{.....(5.3)}$$

$$UU^T = U^TU = I_p\ ,\ VV^T = V^TV = I_n \quad \text{.....(5.4)}$$

U and V are left and right singular vectors. Σ is a diagonal matrix whose diagonal entries are the singular values of **M.** If the rank of **M** is r then

$$\tilde{Ur} = \text{span (u1,...,ur)} \quad \text{.....(5.5)}$$

spans range of M and $\tilde{Ur}^T$ projects M onto that subspace. The relation between the eigenvalue decomposition and SVD can be established from (5.3) as

$$\Sigma^2 = U^TMM^TU \quad \text{.....(5.6)}$$

M is available completely only at the end of test. During test each column of of M is received one after the other. The proposed algorithm approximates (3) at every sampling instance.

Algorithm is described as follows:

$U_0=I$, $C_o=0$

For I = 1 to n

$S_i=U_{i-1}^T m^i$

$B_i=\alpha^2 C_{i-1} + s_i s_i^T$

$C_i=Q_i^T B_iQ_i$

$U_i=U_{i-1}Q_i$

$\hat{m}_I = U_{i-1}\hat{s}_I$

m_i = ith column of M

r=rank(M)

$\hat{s} = [s_1 \ldots s_r\ 0 \ldots 0]$

$U=[\tilde{U}_r \tilde{U}_n]$

Q_i = Jacobi rotation matrix

Matrix U corresponds to the left singular matrix. The matrix C is an approximation to Σ^2 , whereas the matrix B is an intermediate matrix and is an updated version of C in each step. C converges to Σ^2 in (5.6). Convergence is determined by α, which should be selected carefully. Its choice describes preference between stability of the algorithm and its adaptability to the variations in the morphology of the beat. Q_i is Jacobi rotation matrix that nullifies an off diagonal element of B_i , whose absolute value is maximum, is chosen to be made zero at each step in the algorithm. Hence at each step, correlation between the most correlated two output channel is decreased. Effectively (5.8) is performed incrementally:

$\hat{Mi} = [\alpha^{i-1} m(t_1), \alpha^{i-2} m(t_2), \ldots m(t_i)]$ .....(5.7)

$C_i = Q_i^T \ldots Q_1^T \hat{Mi}\ \hat{Mi}^T\ Q_1 \ldots Q_i$ .....(5.8)

$\hat{Ui} = Q_1 \ldots Q_i$ .....(5.9)

$\hat{Ui}$ is closer to U_i than $\hat{Ui}_{-1}$

Where $U_i^T M_i M_i^T U_i = \Sigma^2$ .....(5.10)

$M_i = [m(t_1) \ldots m(t_i)]$ .....(5.11)

Reconstruction is given by (13). $\tilde{U}_r$ corresponds to high singular values and spans the signal space. $\tilde{U}_n$ is its orthogonal component and spans the noise space.

[14] Considers exercise ECG data from 23 patients with record length between 9:00 and 21:00 was recorded with sampling rate of 500 samples/s per channel at 12 bit resolution under Bruce protocol. The data of 8 independent channels in standard 12-lead ECG were acquired simultaneously. SVD is sensitive to the average value of input signals. When the derivations contain a non-zero average or very low-frequency components, SVD algorithm decomposes these as orthogonal components. Such low frequency components increase rank of the data matrix and dimensions of the signal spaces. Hence high passed input signals with a filter of cutoff frequency 0.7 hz are used to remove these low frequency components. The decomposition coefficients obtained on this signal are applied to the input signals that are high pass filtered with a first order filter of cutoff frequency 0.05 hz . 0.05 hz is the allowed cutoff frequency in order to preserve all of the clinical data in ECG.

The difficulty in the technique is that data in one or more channels may be lost. However redundancy may be used to recover important part of data using same

decomposition algorithm. Another difficulty is that the record for longer time is required.

5.2.3 Stimulus artifacts

Several adaptive filtering methods have been proposed for detection and identification of the component waves from noisy ECGs , particularly adaptive Gaussian filter for detection of the QRS component from noisy ECG's [7]. Advantage of adaptive signal processing is that conventional operating systems operate in open-loop fashion while adaptive processors operate in a closed loop fashion [8]. Adaptive filtering of the EMG artifact was attempted with limited success. When the QRS complex disturbs the adaptation process, re-adaptation occurs and artifact appears [10]. Modification of algorithm was carried out in [11] but at the cost of reduction of sharp ECG amplitudes. Luo and Tompkins [12] could obtain better results with faster conversion using additional EMG channel. Rossi R. Casteli [13] proposed a low pass comb filter with reduced lobes in the frequency band above 50 Hz by cascading three averaging filters.

5.2.3.1 Adaptive filtering of Stimulus Artifact

Noninvasive measurements of somatosensory evoked potentials have both clinical and research applications [16]. The electrical artifact which results from the stimulus is an interference which can distort the evoked signal, and introduce errors in response onset timing estimation. Given that this interference is synchronous with the evoked signal, it cannot be reduced by the conventional technique of ensemble averaging. The technique of adaptive noise canceling has potential in this regard however, and bas been used effectively in other similar problems, An adaptive noise canceling filter which uses a neural networks as the adaptive element is investigated in this application. In [13] the filter is implemented and performance determined in the canceling of artifact for in *vivo* measurements on the median nerve. A technique of segmented neural network training is proposed in which the network is trained on that segment of the record time window which does not contain the evoked signal. The neural network is found to generalize well from this training to include the segment of the window containing the evoked signal. Both quantitative and qualitative measures show that significant stimulus artifact reduction is achieved.

Somatosensory evoked potentials (SEP's) are used widely in clinical and research applications, and include peripheral spinal, and cortical potentials. For reasons of convenience, comfort, and safety the preferred measurement techniques are noninvasive using surface electrical stimulation and recording electrodes. The recorded SEP is thus small in amplitude-typically on the order of a microvolt peak.

[17] While an electrical stimulus evokes the desired response from the nervous process of interest, it also results in an interfering potential referred to as the stimulus artifact (SA).

The stimulus artifact is a particularly troublesome form of interference. The amplitude of the SA is typically orders of magnitude larger than the SEP, [18] SA is a synchronous or coherent with the SEP and thus cannot be reduced by ensemble averaging – the common method of reducing other forms of interference, [19] In most cases the SA overlaps the SEP in both the time and frequency domain, such that conventional time windowing and frequency filtering are not capable of removing the SA without distortion of the SEP,[20] The mechanisms by which the electrical stimulus is coupled into the recording are varied complex and distributed, making minimization of the SA at the source problematic [21].

ANC **(Figure 5.2)** can be used for effective reduction of stimulus artifact. An estimate of the SA is obtained via an adaptive filter and the estimate subtracted from the composite SA plus SEP signal. ANC filters have been used in other biological signal acquisition applications. Ye and choy [22] used such a filter for the reduction of respiratory artifact in rhneopneumography measurements. Sadasivan and Narayana [23] developed an ANC filter for the reduction of electroculogram in electroencephalography measurements. Thakor and Zhu [6] have used an ANC filter for the removal of the QRS complex in ECG in order to improve detection of the P-wave; Suzuki et al. [24] developed a real time ANC filter for suppression of ambient noise in lung sound measurements.

In the application of noise canceling to SA reduction Mc Gill et. al[20] estimated the artifact waveform sub threshold stimulation and subtracted the estimate from the suprathreshold stimulus response. Nonlinear adaptive ANC filters were not considered in this study. Work by one of the present authors investigated the application and performance of linear and nonlinear finite impulse response (FIR) ANC filters for SA reduction in SEP measurements [25], [26]. The performance of linear FIR ANC filters was found to be rather poor [25] suggesting that the input – output relationship of tissue to electrical stimulus is nonlinear. Indeed the source of this nonlinearity is at least in part, the nonlinear voltage-current characteristic of the skin [27] A second order truncated Volterra series nonlinear FIR ANC filter was shown to have substantially improved performance [25].

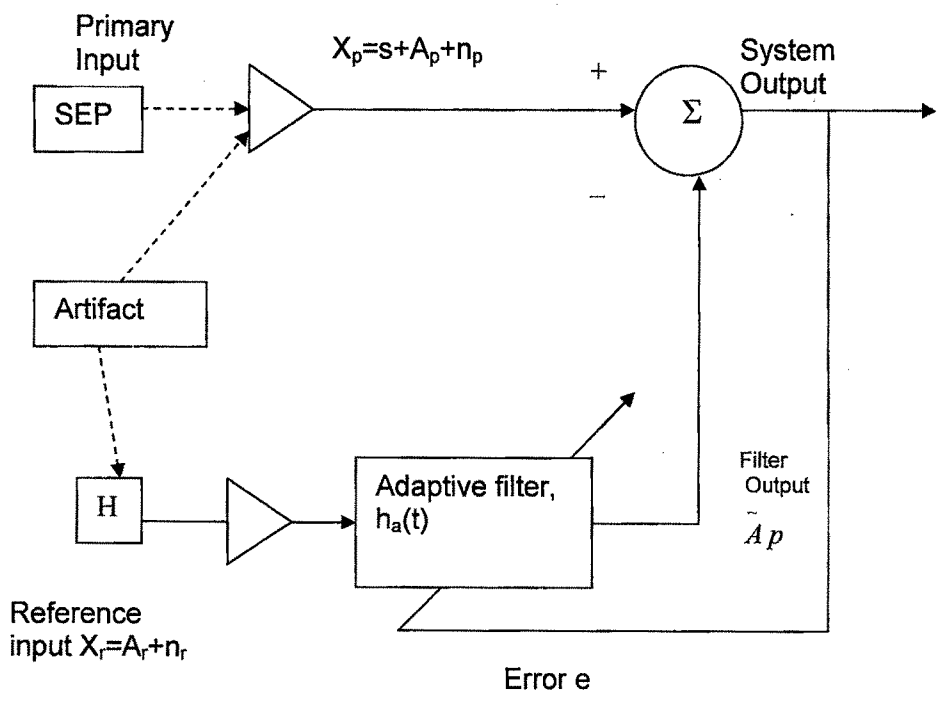

Figure 5.2 Configuration: Adaptive Noise Cancellation (ANC) Filter

The limitations of this method are:

1) It cannot model nonlinearities of order higher than two and
2) The number of filter coefficients grows as the square of the FIR filter length.

- **Event Synchronous Cancellation (ESC):** ESC pursues the concepts of the "event synchronous adaptive interference canceller" (ESAIC)". It uses a simple adaptive gain control (AGC) instead of the complex adaptive filter of the ANC [28].

5.2.4 Electromyogram artifacts

Electromyogram (EMG) artifacts often contaminate the electrocardiogram (ECG). They are more difficult to suppress or eliminate compared for example to the power line interference, due to their random character and to the considerable overlapping of the frequency spectra of ECG and EMG signals obtained from the same pair of electrodes[29]. The usually applied low-pass filtering (cutoff frequency of minimum 35 Hz) results in limited suppression of the EMG artifact and considerable reduction of sharp Q, R and S ECG wave amplitudes. A solution to this problem is proposed by applying approximation filtering with dynamically varied number of samples and weighting coefficients, depending on the ECG signal slope. The slope measure used is

the absolute value of the product of the tilts of two adjacent 10 ms segments sliding along the signal. The results obtained show a slight widening of some sharper QRS complexes, but a virtual preservation of their amplitudes and a considerable reduction of the EMG artifact

Various artifacts often contaminate electrocardiogram (ECG) recording, of which the most common are powerline interference and baseline drift. They were subjected to extensive research and many successful solutions have been found and applied.

The electromyogram (EMG) artifacts, as obtained from the same electrodes as the ECG are difficult to remove, due to considerable overlapping of the frequency spectra of these two types of signals. EMG artifacts in ECG are quite common in subjects with uncontrollable tremor. In disabled persons having to exert efforts in maintaining position of their extremities or a body posture, in children, etc.

Attempts at filtering out the EMG were only partially successful. Therefore, a compromise was universally accepted and postulated in standards and recommendations, namely the applications of a low pass filter with a cut off frequency of minimum 35 Hz [30]. The result is a limited suppression of the EMG artifact and a reduction in the amplitudes of sharp ECG waves, such as Q,R and S. This fact precludes accurate ECG diagnostic measurements if the EMG filter was applied [31].

Adaptive filtering of the EMG artifact was attempted with limited success. When the QRS complex disturb the adaptation process, re adaptation occurs and the artifact appears [6]. A modification of the algorithm could deal with this inconvenient, but the price again is a reduction of the sharp ECG amplitudes [32], Better results with a faster convergence were obtained by Luo and Tompkins [33], but at the price of using an additional EMG channel.

Rossi et al. [34] proposed a low pass comb filter with reduced lobes in the frequency band above 50 Hz by cascading three averaging filters. In order to improve the reduced frequency response, they added a compensating filter gradually enhancing components from 10 to 100 Hz and having first zero at 200 Hz. The goal was to obtain the 3 db cutoff at 35 Hz, as required by the standard. Unfortunately, no data on the compensating filter were given and no results with real signals were presented.

Levkov [35] also tried to use comb filters for simultaneous suppression of 50 Hz interference and EMG artifacts. However, reaching -3 db at 35 Hz resulted in an enhanced lobe with a maximum of 95% at 70 Hz.

5.2.4.1 Approximation Filtering

To improve the traditional compromise between efficient EMG artifact suppression and preservation of the ECG waveform, approximation filtering, adds dynamic modification of the approximation function parameters depending on the ECG signals slew rate.

The characteristics of dynamic filter are dynamically changed depending on signal slope. For this purpose a function called "wings" [36] is applied. A wing function is formulated by multiplying slopes of two adjacent segments of equal length having common point. A relationship between the slopes and number of samples is established.

The advantage of this method is that the drift problem is eliminated, as the base line fluctuations do not exceed 0.5Hz. The backward filtering [13] can be used to remove drift if required. The method improves the compromise between EMG suppression and preservation of ECG waveform.

5.2.5 Direct Cosine Transform

The filter Proposed in [37] assumes the noisy electrocardiography to be modeled as a signal of deterministic nature, corrupted by additive muscle noise artifact. The muscle noise component is treated to be stationary with known second-order characteristics.

Noise-free ECG is shown to possess a narrow-band structure in discrete cosine transform (DCT) domain. The second order statistical properties of the additive noise component is preserved due to the orthogonality property of DCT, noise suppression is easily accomplished via subspace decomposition in the transform domain. The subspace decomposition is performed using singular value decomposition (SVD)

Method [37] uses the properties of the discrete cosine transform (DCT) in conjunction with conventional smoothing algorithms using singular value decomposition (SVD) for enhancement of the SNR This method uses a single lead information with a limited amount of data for processing.

The order of the transform domain SVD filter required to achieve the desired degree of noise suppression is compared to that of a suboptimal Wiener filter using DCT.

Since the Wiener filter assumes both the signal and noise structures to be statistical, with a priori known second order characteristics, it yields a biased estimate of the ECG beat as compared to the SVD filter for a given value of mean square error (MSE). The filter order required for performing the subspace smoothing is shown to exceed a certain minimal value for which the MSE profile of the SVD filter follows the minimum mean square error (MMSE) performance warranted by the suboptimal Wiener filter. The effective filter order required for reproducing clinically significant features in the noisy ECG is than set by an upper bound derived by means of a finite precision linear perturbation model. A significant advantage resulting from the application of the proposed SVD filter lies in its ability to perform noise suppression independently on a single lead ECG record with only a limited number of data samples.

ECGs recorded under exercise conditions are often corrupted by extraneous disturbances due to muscular activity (electromyographic (EMG) noise) and respiration. The EMG noise is random in nature and has a frequency content existing over a wide range. Under exercise conditions, the level of these interfering signals, particularly the muscle noise component become large enough thereby mutilating the signal characteristics which are clinically significant. For example, the small amplitude P-wave are hidden by the larger amplitude muscle artifacts and render it difficult for the physician to locate the presence or absence of these waves, which in turn provides significant clinical information required to diagnose certain abnormalities associated with the functioning of the heart. Moreover, since the spectral content of muscle noise overlaps that of the ECG, improvement of signal to noise ratio (SNR) solely by means of digital filtering is not possible without introducing considerable distortion in the ST segment regions. In most situations the approximate shape of the component wave, expected to be present in the noisy signal, may be known and the requirement is to estimate its time of occurrence and its exact shape. The two different approaches used to solve this problem are those which employ the structural features of the component wave and secondly the method that use template matching technique [38] Algorithm based on the first approach are of heuristic nature and are selective to the particular type of component wave being searched for (for example, the QRS complex) In the second approach. The approximate knowledge on the shape of the component wave is used to generate a template which is determined by means of correlation, matched filtering, or other pattern recognition techniques.

Traditionally, a large number of algorithms for noise reduction in ECG's use either spatial or temporal averaging techniques. Assuming the noise to be random and stationary, the noise reduction by the temporal averaging method is proportional to the square root of the number of frames or beats averaged [39],[40]. As is evident, the temporal averaging method requires a large number of beats or frames for effective noise reduction. Moreover, the averaging may cause considerable errors particularly when the time alignment of beats beats is not accurately known, or especially when premature beats are present [41] In the spatial averaging method, potentials from 4-16 independent electrode pairs are acquired and averaged Initial work using spatial averaging was attempted by flowers et al [42] and EL Sherif et al. [43] The main drawback of spatial averaging is the physical limitation to placement of a large number of electrodes at the same region.

Several adaptive filtering methods have been proposed for detection and identification of the component waves from noisy ECG's of particular interest is the adaptive Guassian filter for detection of the QRS component from noisy ECG's [44] The adaptive tuning is performed on the frequency response of the Gaussian filter in order to minimize the distortion of the undisturbed signal by the filter. A second method for adaptive noise cancellation of ECG recordings has been cited in [6] and works on the principal that EMG noise recorded using two different orthonormal limb leads are uncorrelated. The inherent limitation of this scheme is the requirement of generating a signal whise noise component is orthonormal to that in the noisy ECG.

Single lead ECG's are more common in an exercise testing than in resting ECG's and the one leadedness of the method is an additional feature suitable for its application to exercise ECG's in particular. The method does not require a prior knowledge of onset and offset points or temporal alignment of beats as required in other methods based on signal averaging. [37] describes the formulation of noise removal problem and SVD smoothing algorithm.

The performance of the SVD filter was evaluated [37] using simulated signals and compared to that of a suboptimal Wiener filter using DCT. The choice of a minimal filter order is decided by choosing the minimum value of the matrix size which results in a MSE performance close to the MMSE of the suboptimal Wiener filter using DCT. The transform domain SVD filter is capable of reproducing clinically significant morphological features from ECG waveforms with an SNR of up to 10 dB by using a filter order exceeding the minimal value and bounded by an upper limit decided by the linear perturbation model. The superiority of the proposed method lies

in the fact that it does not require any prior information about the onset and offset points or the knowledge of beat intervals for its satisfactory performance. Moreover, the capability of the method to work with a limited amount of data is an added advantage.

5.2.6 Multi reference Adaptive Noise Cancellation

Noise cancellation using LMS adaptive filters often leads to large steady-state residual noise, as the desired signal impedes the estimation procedure. In a recent paper, the knowledge of the desired signal statistics has been exploited to formulate a whitening mechanism which reduces the residual noise variance. In [45] is shown how proposed algorithm could be unstable in the absence of a strictly positive real condition on the whitening filter, suggest a globally stable modification, prove its convergence, and demonstrate its efficacy through simulation.

Adaptive identification with the LMS and related algorithm has become a popular solution to many problem [46-53]. Consider echo cancellation [48-50]. The received signal contains not only the incoming message, but also the weighted and delayed versions of the outgoing signal. If we estimate these weights and apply them to a copy of the outgoing signal, we have an estimated echo. If we subtract this synthetic echo from the received signal, we are more likely to understand the incoming message. The quality of the estimated echo improves with more accurate weight estimates. Similar ideas can be applied to the general problem of noise removal [46], [51-53] by treating the noise in the same manner as one treats echoes in the above problem.

The LMS algorithm is used to adaptively estimate the weights. Exact convergence of the estimated weights to their modeled values is possible, provided the parameter update is guided by the precise residual noise (echo) However, this noise is not directly measurable, so the system output (the desired signal plus residual noise) is generally used in its place. Therefore, when the errors get sufficiently small, the desired signal becomes the dominant component of the system output and obstructs further improvement of the weight estimates. The result is a steady state error variance proportional to the update gain and desired signal energy [58].

It may be possible to reduce the steady state error with out significantly reducing the value of the update gain, and thus slowing convergence. One way to do so is to subtract an estimate of the desired signal from the output, to get a better estimate of the residual noise. This requires prior information about the desired signal.

In [54], the technique of multi reference adaptive noise canceling MRANC (Figure **5.3**) is applied to enhance transient nonstationaries in the electroencephalogram (EEG) with the adaption implemented by means of a multilayer-perception artificial neural network (ANN).

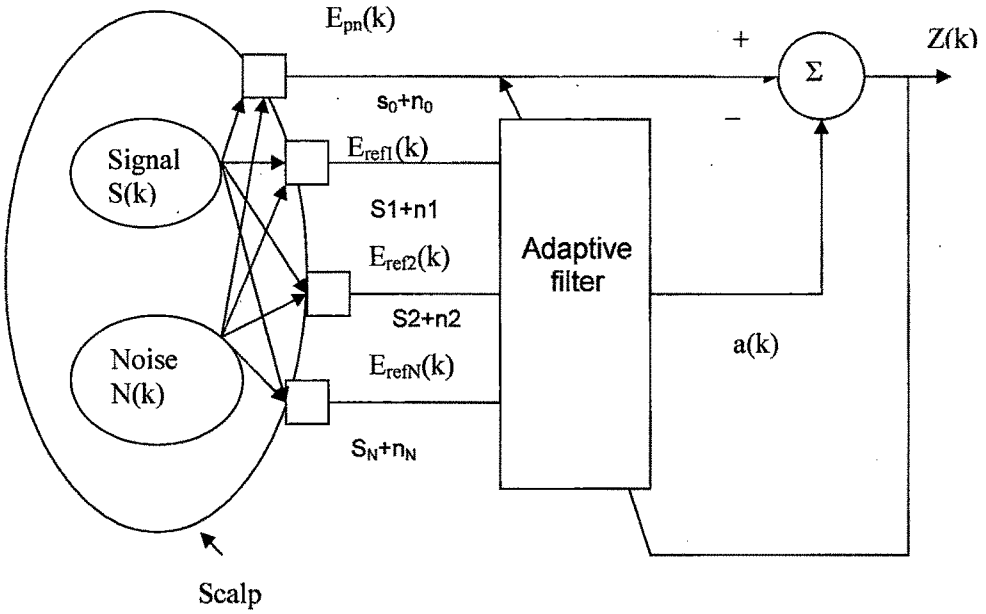

Figure 5.3 Multi Reference ANC

The method was applied to recorded EEG segments and the performance on documented nonstationarities recorded. The result show that the neural network (nonlinear) gives an improvement in performance (i.e. signal to-noise ratio (SNR) of the nonstationarities) compared to a linear implementation of MRANC. In both cases an improvement in the SNR was obtained. The advantage of the spatial filtering aspect of MRANC is highlighted. Performance of MRANC is compared to that of the inverse auto successive filtering of the EEG.

The electroencephalogram EEG can be considered to consist of an underlying background process (assumed stationary and ergodic) with superimposed transient nonstationarities (TNS's) such as spike and sharp-waves (SSW's) electrode "pop" eye-blinks, and muscle artifacts. The detection of SSW's in the EEG is of particular importance in the diagnosis of epilepsy.

Methods for detecting SSW's nave included mimetic methods [55],[56] and the use of template matching [57]. The lack of any definition of a SSW other than "transient

clearly distinguished from background activity with pointed peaks at conventional paper speeds" [58] means that what constitutes the "ideal" SSW can vary amongst researchers. Instead of matching a single template, several authors have employed an artificial neural network (ANN) by training the ANN on a large number of known SSW's [59],[60] Lopes da silva et al. [61] used the method of modeling the stationary background EEG with an autoagressive (AR) prediction filter and detecting TNS's by examining the rediction error; the AR filter was calculated from a segment of the background EEG which is assumed to be stationary. The major drawback is that the stationary assumption may not always hold true, leading to a large number of false detections.

The method described in [54] comprises the first stage of a ANN based system designed to detect SSW's in the interictal EEG. The system makes use of multi reference adaptive noise canceling (MIRANC) as described by widrow et al [62], The background EEG on other channels in the multi channel EEG recording is used to adaptively cancel the background EEG on the channel under investigation. The use of a multilayer ANN to implement the MRANC filter provides the opportunity to model the EEG spatial distribution as nonlinear and leads to improved performance over the linear case.

The EEG signal is assumed to consist of a signal s_o (here, modeling the TNS) contaminated by noise n_o (here, modeling the background EEG) which is assumed to be uncorrelated with the signal. Each reference input $E_{ref}(k)$ contains a noise signal n_i which is uncorrelated with s_0, but correlated with n_0. The adaptive filter adapts its parameters so as to produce an output signal which is as close as possible to n_0. This output is then subtracted from the primary input, canceling the noise content n_0 but leaving signal s_0 intact. The adaptive filter continuously adjusts to minimize the output z. Any suitable adaptive algorithm which minimizes the output can be used; in particular, the least mean square (LMS) adaptive algorithm [62] can be used if the system is assumed to be linear. The LMS algorithm is employed in the work reported here to compare with the nonlinear ANN described below. The reference inputs to the adaptive noise canceller may contain some signal components, which are correlated to the signal at the primary input (**Figure 5.3**).

5.2.7 Optimal Multichannel Filtering

The artifacts presented by precordial compressions during cardiopulmonary resuscitation (CPR) could be removed from the human electrocardiogram (ECG) using a filtering approach [63]. This study was to allow analysis and defibrillator charging during ongoing precordial compressions yielding a very important clinical

improvement to the treatment of cardiac arrest patients. In this investigation authors started with noise-free human ECG with ventricular fibrillation (VF) and ventricular tachycardia (VT) records. To simulate a realistic resuscitation situation, they added a weighted artifact signal to the human ECG, where the weight factor was chosen to provide the desired signal –to-noise ratio (SNR) level. As artifact signals they used ECG's recorded from animals in asystole during precordial compressions at rates 60,90, and 120 compressions/min. The compression depth and the thorax impedance were also recorded.

An adaptive multichannel Wiener filter was used to construct an estimate of the artifact signal. The estimate was subtracted from the noisy human ECG signal. The method was used for CPR artifact and it improves SNR and rhythmic classification.

Early defibrillation is the most important factor for restoration of spontaneous circulation (ROSC) in settings of sudden deaths, in which the majority of instances are due to ventricular fibrillation (VF) [64], [65]. In the current operation of automated external defibrillators, substantial time is consumed in the "hands-off" interval during which precordial compressions are discontinued to allow for automated rhythm analysis before delivery of the electric counter-shock. Current guidelines for cardiopulmonary resuscitation require that chest compression and ventilation must be interrupted prior to any shock, to avoid the effects of artifacts on the electrocardiogram (ECG) analysis [66],[67]. During this period, there is no circulation of heart muscle or brain with a rapid deterioration of the metabolic state of the tissues [68],[69], which can be partially reversed by chest compressions and ventilations [70],[71]. In agreement with this, Sato et al. found that the interruption of chest compressions before a defibrillation attempt reduced the defibrillation success rate 24-h survival was significantly reduced with a 20-s delay [72]. They concluded that automated defibrillators are likely to be maximally effective if they are programmed to secure minimal "hands-off" delay before delivery of the electric counter-shock. Furthermore, the removal of cardiopulmonary resuscitation (CPR) artifacts in VF would make it possible to assess the CPR effect on the myocardium as indicated by VF changes. In a study of 883 shocks in patients during out of hospital resuscitation, these episodes were of that duration with a median of 20 s [73]. In the clinical study only 10% of the shock given to patients with VF or VT resulted in a rhythm with a pulse and the patients received median six shocks with significantly fewer shocks given to survivors than non survivors [73]. Others have reported median 3-6 shock/patient with fewer shocks given to survivors than non survivors [74-77]. Thus, most patients with VF and VT have many episodes where chest compressions

and ventilation cannot be administered due to signal analysis of the ECG and defibrillator charging. This can also be seen in connection with the fact that the success rate for a defibrillation attempts depends on the frequency configuration of the VF [78]-[79], and this configuration is affected by the CPR [80]. In agreement with this Cobb et al [92] have recently reported an improved success rate when a period of chest compressions and ventilations were given before a defibrillation attempt in patients.

In [75] hypothesis was that artifact removal by filtering would allow the maintenance of precordial compressions during automatic rhythm analysis. Without interrupting the sensitivity of simultaneous rhythm classification, artifact removal has been done successfully on animal ECGs applying high pass digital filters with fixed coefficients [78],[79]. In human ECG however, the frequency component of the artifacts overlap with the frequency components of the desired signal, which renders separation by such filters infeasible.

In an attempt to solve this problem, the following strategy was put forward in [81]: The ECG artifact is modeled as a sum of several sources; the main source being the mechanical stimulation of the heart itself. For each source the idea is to obtain a reference signal correlated to the associated artifact component. In case of artifacts due to chest compressions one such reference is the compression depth. The reference signal facilitate removal of the artifact component using an adaptive, digital filter.

Animal experiments were carried out in [81] for collection reference signals and the corresponding ECG artifact signals. Collection of artifact information was performed on well-established pig model [82],[83]. During precordial compression on domestic anaesthetized pigs in cardiac arrest, the ECG with artifacts and corresponding references were continuously recorded during different compression rates with constant compression depth. Data were collected at different compression rates 60,90 and 120 min^{-1} – the latter because evidence favors a compression rate up to 120 min^{-1} [84].

The setup in [81] was used for signal collection. The data was combined with human ECG in order to stimulate an artifact contaminated human ECG signal. The algorithm [63] was tested for SNR improvement and sensitivity.

5.3 Artificial Neural Network based filtering

An Artificial Neural Network (ANN) is an attempt to mimic the action of the brain using simple structure. The ANN is built up using a class of adaptive machine that

perform computation through process of learning. The large number of artificial neurons are interconnected to form the network. Thus neural network consists of massively parallel distributed processors which have a neural propensity for storing experienced knowledge and making it available for use. ANN can be programmed or trained to store, recognize and associatively retrieve patterns or database entries; to solve combinational optimization problems, in summary, to estimate sampled functions when we do not know the form of the functions. The overall network behaves as an adaptive function estimator. This can be useful to carry out signal processing task like filtering, averaging, compression, detection etc.

An adaptive filter automatically adjusts its own impulse response. In [85] adaptive noise canceller and adaptive signal enhancer systems are implemented using feed forward and recurrent neural networks using back propagation algorithm and real time recurrent learning algorithm respectively for training. Their performances are compared with conventional adaptive filtering techniques using LMS and RLS algorithms. The recurrent neural network employing RTRL algorithm which functions better than the other algorithms is studied further by varying the number of nodes, adding a bias to the neurons, adding a momentum term for learning and varying the momentum term and learning rate for better convergence.

Conventional signal processing systems operate in an open-loop fashion. Adaptive processors operate in a closed-loop fashion. The adaptive filter estimates the static of the incoming signal and adjusts its own response in such a way that some cost function is minimized. This cost function may be derived in different ways based on the application. Adaptive filters find wide applications in different fields such as communications, control, radar and seismology. In this paper adaptive noise cancellation and adaptive signal enhancement systems are considered for implementation.

Artificial neural networks are parallel computational models comprised of interconnected adaptive processing ants. A very important characteristic of these networks is their adaptive ability where "learning by example" replaces "programming" in solving problems. Neural networks can offer the computational power of non-linear techniques. Signal processing techniques have been improved by the application of artificial neural networks [86] Neural network based adaptive filters find wide applications in biomedical signal processing [87-88] In this section feed forward and recurrent neural networks are used to implement adaptive noise canceller and adaptive signal enhancer. Their performances are compared with those

implemented using conventional LMS (Least Mean Square) and RLS (Recursive Least square) algorithms.

A new global optimization strategy for training adaptive systems such as neural networks and adaptive filters with finite or infinite impulse response (FIR or IIR) is proposed [89]. Instead of adding random noise to the weights as proposed in the past, additive random noise is injected directly into the desired signal. Experimental results show that this procedure also speeds up greatly the back propagation algorithm. The method is very easy to implement in practice, preserving the back propagation algorithm and requiring a single random generator with a monotonically decreasing step per output channel. Hence, this is an ideal strategy to speed up supervised learning and avoid local minima entrapment when the noise variance is appropriately scheduled.

It is well known that the two major problems associated with back propagation learning are the slow convergence for complex problems and local minima entrapment. For the first problem, several improvements such as quick prop, momentum learning adaptive step sizes, etc. have been proposed [90],[91]. Local minimum entrapment can be solved by simulated annealing or related techniques which include the Langevin algorithm and the diffusion optimization method [92],[93],[94]

The common point of these methods is the injection of a noise term of controlled variance into each weight vector. These methods have a very slow convergence but they can theoretically overcome local minima. Another weakness is that one has to control a lot of internal variables (noise terms for each weights) which is not very efficient during learning one wishes to adjust only the external variables such as input, desired signal, and step size. From a pragmatic point of view on line algorithms, i.e., algorithms where the weights are updated with every sample are highly desired but in simulated annealing due to the stochastic nature of the updates on-line methods cannot be efficiently implemented. Motivated by these results, we propose here to add noise to the desired signal and experimentally investigate the advantages of such procedure.

Noise has been used with gradient descent procedures. Holmstrom analyzed the generalization ability of the static back propagation (BP) algorithm [95] when random noise was injected into the external signals. They showed that the generalization can be improved using additive noise in the training data. Bishop shows that injection of

noise in the input is equivalent to regularization [96] several other papers also showed that noise injection into the input and weights can also improve the generalization [97],[98],[99],[100],[101]. All of these papers concentrate mostly on the network generalization ability and did not study the effect of noise in the learning speed and the ability of escaping from local minima. We will address this issue in this paper and propose to inject the noise in the desired response, not at the input, due to the linearly of the dual network. A theoretical and experimental performance study of a method for time delay estimation (TDE) based on the signal integral (TDS-SI) is presented in [102] The TDE-SI method considers the delay between two transient signals as the difference between the centers of mass of these signals. Estimations are validated by stimulation result with artificially generated signals and by real so called QRS complex waves (ventricular activity) from an electrocardiographic (ECG) signal.

The Hopfield, MPLNN and RBFNN model have been proposed for filtering.

5.3.1 Hopfield Neural Network based filtering (LS Algorithm)

5.3.1.1 Filter Design

The basic promise behind optimal filtering is that we must have knowledge of both the signal and noise characteristics. Building adaptive digital filters can perform noise cancellation and signal extraction. Adaptive techniques are advantageous because they do not require a prior knowledge of the signal or the noise characteristics. Adaptive filters employ a method of learning through an estimated synthesis of a desired signal and error feedback to modify the filter parameters.

In adaptive filtering, it is absolutely necessary to compute the weight coefficients of the filter in real time so as to make the corresponding transform function trace the characteristics of the input signals in the desired way. However the computational complexity of the available algorithms for computing the weight coefficients is very intensive. Although many methods have been proposed to decrease the computational complexity, it is still difficult to deliver the desired real - time performance if conventional digital and sequential methods are used. Continuous time Hopfield Neural Networks is used to solve the above problem. The principle of an adaptive filter is shown in **figure 5.4.**

5.3.1.2 Least Square Algorithm

The Least Squares (LS) algorithm for computing the weight coefficients of the adaptive filter is regarded as

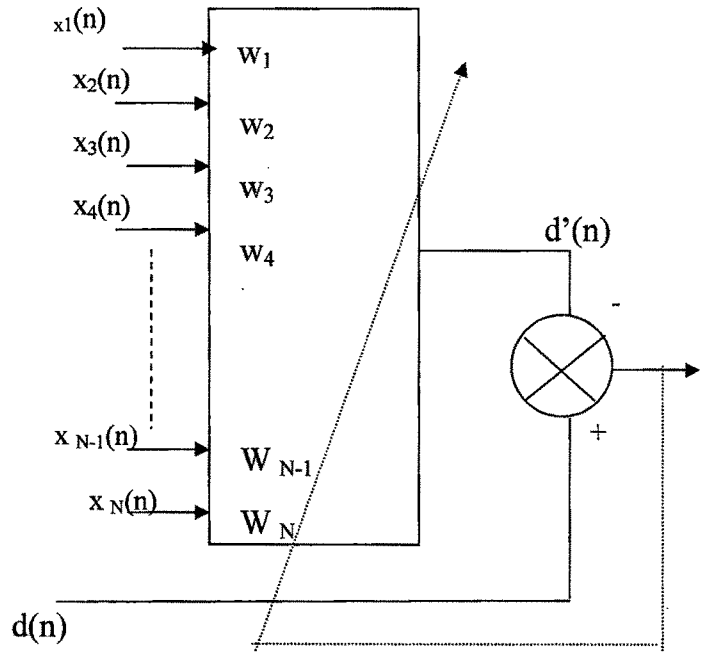

Figure 5.4 Adaptive Filter

$$\underset{W}{\text{Min}} \sum_{n=1}^{M} [d(n) - X^T(n)W]^2 \quad \text{.....(5.12)}$$

The problem (5.12) is written in a matrix form

$$\underset{W}{\text{Min}} \, ||\mathbf{d} - \mathbf{XW}||^2 \quad \text{.....(5.13)}$$

The solution of (5.13) is

$$W_{LS} = \mathbf{X}^+\mathbf{d} = \lim_{\alpha \to 0} (\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1}\mathbf{X}^T\mathbf{d}. \quad \text{.....(5.14)}$$

Where **X+** is the pseudo inverse of **X.**

Because (5.14) involves a matrix pseudo inversion, the computational complexity is intensive. Hopfield Neural Network can be employed to solve for the complexity. Such model is presented in figure **5.5.** Comparing with **figure 5.4,** let connection strength matrix, **T** = available noisy signal matrix, **X** and bias current vector, **B** = **noisy** signal matrix, **d.** For simplification, let $f(u) = K_1u$ and $g(u)=K_2u$.

In terms of Kirchhoff's laws, one can have the following relationship for the proposed model

$$C\frac{du_i(t)}{dt} = \sum_{j=1}^{M} T_{ji} f\left(\sum_{k=1}^{N} T_{jk}\upsilon_k(t) - b_j\right) - \frac{u_i(t)}{R} \quad \text{.....(5.15)}$$

For this model, we define an energy function as follows:

$$E(t) = \sum_{j=1}^{M} F\left(\sum_{i=1}^{N} T_{ji}\upsilon_i(t) - b_i\right) + \sum_{i=1}^{N} \frac{1}{R}\int_0^{\upsilon_i(t)} g^{-1}(\upsilon)d\upsilon. \quad \text{.....(5.16)}$$

Where F(.) is the indefinite integral of the function $f(u)$.
By writing (5.16) in matrix form and making these simplifications, we have,

$$C\frac{dU(t)}{d(t)} = -\frac{U(t)}{R} - X^T Q(t) \quad \text{.....(5.17)}$$

Where, $Q(t) = K_1(XV(t)-d)$

$$= K_1(K_2XU(t)-d) \quad \text{.....(5.18)}$$

Also, $$\frac{dU(t)}{dt} = -(1/RC + K_1K_2\mathbf{X^TX}/C)\mathbf{U}(t) + K_1\mathbf{X^T}d/C \quad \text{.....(5.19)}$$

The input vector U_f and the output vector V_f in the stationary state can be obtained by letting

$$\frac{dU(t)}{dt} = 0 \text{ for all i.}$$

$$\mathbf{Uf} = \lim_{t\to\alpha} \mathbf{U}(t) = K_1(\mathbf{I}/R + K_1K_2\mathbf{X^TX})^{-1}\mathbf{X^Td} \quad \text{.....(5.20)}$$

$$\mathbf{Vf} = \lim_{t\to\alpha} \mathbf{V(t)} = (\mathbf{I}/RK_1K_2 + \mathbf{X^TX})^{-1}\mathbf{X^Td} \quad \text{.....(5.21)}$$

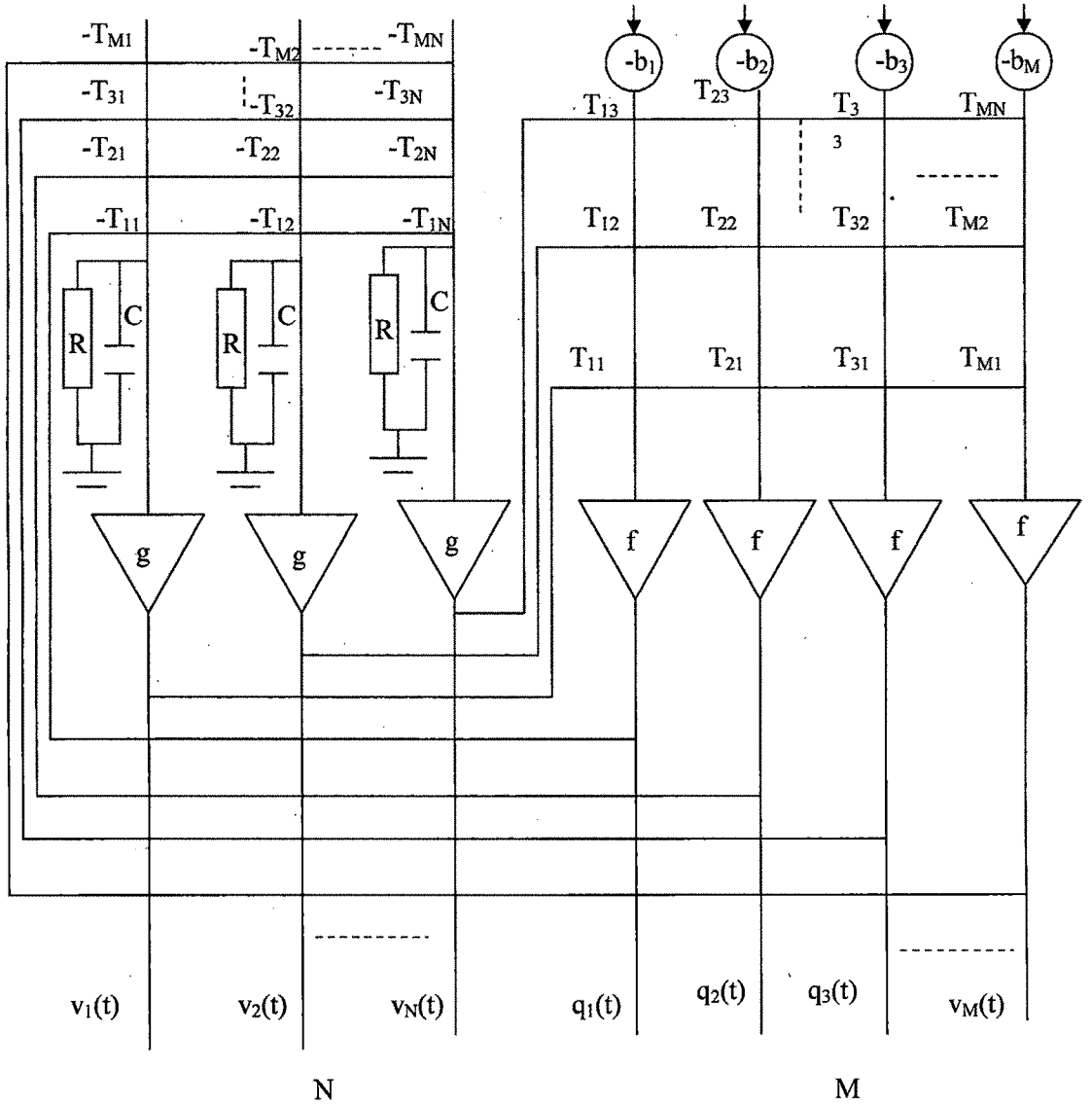

Figure 5.5 Hopfield Model for Least Square Algorithm

Equation (5.21) is an approximation of the LS solution **W.** It is implemented using MATLAB. Since there are N weights, N equations are required to be solved. For M samples dimension of the matrix is M x N.

5.3.1.3 Terminology

d(1), d(2) .....,d(M) : set of known quantities and dimension: M X 1

$\underline{\mathbf{X(n)= \{x_1(n),x_2(n),.....,x_N(n)\}^T \ (n = 1,2,.....,M):}}$ vectors that stand for the input samples of the filter dimension : M X N

$$X = \begin{bmatrix} x_1(1) & X_2(1)\ldots & X_N(1) \\ X_1(2) & X_2(2)\ldots & X_N(2) \\ . & . & . \\ . & . & . \\ X_1(M) & X_2(M)\ldots & X_N(M) \end{bmatrix}$$

$\underline{W = [W_1, W_2,\ldots,W_N]^T}$: **N X 1** unknown weight coefficient vector of the filter .

g(u) and *f*(u) : The input-output relationship of the neurons in the left and right parts of network

N: Number of neurons in the left part

M: Number of neurons in the right part

$\underline{R_i}$ **and** $\underline{C_i}$ **(fot i = 1,2,....,N) :** The input resistance and capacitance of the i'th neuron in the left part, respectively (for Mathematical convenience , here we let Ri = R and Ci =C).

$\underline{T = \{T_{ij}\}}$ (for j = 1,2,..,M; i = 1,2,...,N) is the connection strength matrix.

$\underline{U(t) = [u_1(t), u_2(t), \ldots\ldots u_N(t)]^T}$ = input voltage vector of the left part
$\underline{Q(t) = [q_1(t), q_2(t), \ldots\ldots q_N(t)]^T}$ = input voltage vector of the right part
$\underline{B = [b_1, b_2,\ldots, b_M]^T}$: The bias current vector of right part
$\underline{\upsilon_i(t)}$ **and** $\underline{q_i(t)}$ **(for i = 1,2,...,N; j = 1,2,...,M)** : The neuron output of the left and right parts of the network, respectively.

5.3.1.4 ECG data

The samples of biosignals used in all algorithms consist of two ECG records and one EMG (noise) records. These are 8 channel records of two peripheral and 6 precordial leads, taken with respect to the left leg electrode. Sampling rate is 400 Hz, 12 bit ADC records (one byte least significant bits; 1 byte (half – full) most significant bits), with resolution of 4.88 mV/bit.

EMG signals were obtained from one ECG electrodes placed on one forearm. The ECG amplifier was used and the recordings were made during sustained voluntary effort. The artifacts thus obtained were weighted and additively mixed with the different ECG signals subjected to processing.

5.3.1.5 MATLAB code for filter

The steps followed to implement filter are:

1. The ECG data file created through ADC add-on card for patient is read.
2. From the knowledge of data structure for ECG storage, data items for segments are separated.
3. The EMG data file is also read. **Step 2** is repeted for the EMG signal.
4. The EMG noise is added to ECG to obtain corrupted signal.
5. The signals are plotted.
6. The filtering LS filtering algorithm is used.
7. Filtered signal are plotted.

5.3.1.5(A) MATLAB code for reading ADC Data and store in files

This MATLAB code for reading ECG/EMG Data is common for all the applications developed and hence will not be repeated again.

```
fid = fopen('N0001.adc..adc','r');          %one of the ECG file
text=fread(fid,2048,'char');
d=fread(fid,'int16');
l=3200;              % length of a lead
%-------------------------------------------------------------------------------
Lecg=d(1:l);         % 3200 length first piece is LECG
Recg=d(l+1:2*l);     % Similarly next 3200 samples are for RECG
C1ecg=d(2*l+1:3*l);
C2ecg=d(3*l+1:4*l);
C3ecg=d(4*l+1:5*l);
C4ecg=d(5*l+1:6*l);
C5ecg=d(6*l+1:7*l);
C6ecg=d(7*l+1:8*l);
Recg=Recg*0.00488;       %Multiplication with 1 bit value for all samples
Lecg=Lecg*0.00488;
C1ecg=C1ecg*0.00488;
C2ecg=C2ecg*0.00488;
C3ecg=C3ecg*0.00488;
C4ecg=C4ecg*0.00488;
C5ecg=C5ecg*0.00488;
C6ecg=C6ecg*0.00488;
%-------------------------------------------------------------------------------
fid = fopen('emg01.adc..adc','r');
text=fread(fid,2048,'char');
d=fread(fid,'int16');
l=3200;              % length of a lead
%-------------------------------------------------------------------------------
```

```
Lemg=d(1:l) ;          %Similar to ECG signals, first 3200 samples are for LEMG

Remg=d(l+1:2*l);
C1emg=d(2*l+1:3*l);
C2emg=d(3*l+1:4*l);
C3emg=d(4*l+1:5*l);
C4emg=d(5*l+1:6*l);
C5emg=d(6*l+1:7*l);
C6emg=d(7*l+1:8*l);
%-------------------------------------------------------------------------------------------
Remg=Remg*0.00488;
Lemg=Lemg*0.00488;
C1emg=C1emg*0.00488;
C2emg=C2emg*0.00488;
C3emg=C3emg*0.00488;
C4emg=C4emg*0.00488;
C5emg=C5emg*0.00488;
C6emg=C6emg*0.00488;
%-------------------------------------------------------------------------------------------
Recgn=    [Recg;0;0]+[0;0;Remg];%+ [0;0;0;0;0;Rp;0;0;0;0;0]; %Signal and noise
are mixed up
Lecgn=   [Lecg;0;0]+[0;0;Lemg];%+ [0;0;0;0;0;Lp;0;0;0;0;0];
C1ecgn=  [C1ecg;0;0]+[0;0;C1emg];%+[0;0;0;0;0;C1p;0;0;0;0;0];
C2ecgn=  [C2ecg;0;0;]+[0;0;C2emg];%+[0;0;0;0;0;C2p;0;0;0;0;0];
C3ecgn=  [C3ecg;0;0]+[0;0;C3emg];%+[0;0;0;0;0;C3p;0;0;0;0;0];
C4ecgn=  [C4ecg;0;0]+[0;0;C4emg];%+[0;0;0;0;0;C4p;0;0;0;0;0];
C5ecgn=  [C5ecg;0;0]+[0;0;C5emg];%+[0;0;0;0;0;C5p;0;0;0;0;0];
C6ecgn=  [C6ecg;0;0]+[0;0;C6emg];%+[0;0;0;0;0;C6p;0;0;0;0;0];
%-------------------------------------------------------------------------------------------
fid=fopen('Recg.dat','w');  %Records are written in respective files, which will be read
later
fprintf(fid,'%f\n',Recg);
fclose(fid);
fid=fopen('Recgn.dat','w');
fprintf(fid,'%f\n',Recgn);
fclose(fid);
fid=fopen('Remg.dat','w');
fprintf(fid,'%f\n',Remg);
```

```
fid=fopen('Lecg.dat','w');
fprintf(fid,'%f\n',Lecg);
fclose(fid);
fid=fopen('Lecgn.dat','w');
fprintf(fid,'%f\n',Lecgn);
fclose(fid);
fid=fopen('Lemg.dat','w');
fprintf(fid,'%f\n',Lemg);
fclose(fid);
fid=fopen('C1ecg.dat','w');
fprintf(fid,'%f\n',C1ecg);
fclose(fid);
fid=fopen('C1ecgn.dat','w');
fprintf(fid,'%f\n',C1ecgn);
fclose(fid);
fid=fopen('C1emg.dat','w');
fprintf(fid,'%f\n',C1emg);
fclose(fid);
fid=fopen('C2ecg.dat','w');
fprintf(fid,'%f\n',C2ecg);
fclose(fid);
fid=fopen('C2ecgn.dat','w');
fprintf(fid,'%f\n',C2ecgn);
fclose(fid);
fid=fopen('C2emg.dat','w');
fprintf(fid,'%f\n',C2emg);
fclose(fid);
fid=fopen('C3ecg.dat','w');
fprintf(fid,'%f\n',C3ecg);
fclose(fid);
fid=fopen('C3ecgn.dat','w');
fprintf(fid,'%f\n',C3ecgn);
fclose(fid);
fid=fopen('C3emg.dat','w');
fprintf(fid,'%f\n',C3emg);
fclose(fid);
fid=fopen('C4ecg.dat','w');
fprintf(fid,'%f\n',C4ecg);
```

```
fclose(fid);
fid=fopen('C4ecgn.dat','w');
fprintf(fid,'%f\n',C4ecgn);
fclose(fid);
fid=fopen('C4emg.dat','w');
fprintf(fid,'%f\n',C4emg);
fclose(fid);
fid=fopen('C5ecg.dat','w');
fprintf(fid,'%f\n',C5ecg);
fclose(fid);
fid=fopen('C5ecgn.dat','w');
fprintf(fid,'%f\n',C5ecgn);
fclose(fid);
fid=fopen('C5emg.dat','w');
fprintf(fid,'%f\n',C5emg);
fclose(fid);
fid=fopen('C6ecg.dat','w');
fprintf(fid,'%f\n',C6ecg);
fclose(fid);
fid=fopen('C6ecgn.dat','w');
fprintf(fid,'%f\n',C6ecgn);
fclose(fid);
fid=fopen('C6emg.dat','w');
fprintf(fid,'%f\n',C6emg);
fclose(fid);
fid = fopen('N0001.adc..adc','r');          %one of the ECG file
text=fread(fid,2048,'char');
d=fread(fid,'int16');
l=3200;                 % length of a lead
%-------------------------------------------------------------------------------
Lecg=d(1:l);            % 3200 length first piece is LECG
Recg=d(l+1:2*l);        % Similarly next 3200 samples are for RECG
C1ecg=d(2*l+1:3*l);
C2ecg=d(3*l+1:4*l);
C3ecg=d(4*l+1:5*l);
C4ecg=d(5*l+1:6*l);
C5ecg=d(6*l+1:7*l);
C6ecg=d(7*l+1:8*l);
```

```
Recg=Recg*0.00488;          %Multiplication with 1 bit value for all samples
Lecg=Lecg*0.00488;
C1ecg=C1ecg*0.00488;
C2ecg=C2ecg*0.00488;
C3ecg=C3ecg*0.00488;
C4ecg=C4ecg*0.00488;
C5ecg=C5ecg*0.00488;
C6ecg=C6ecg*0.00488;
%-------------------------------------------------------------------------------------------
fid = fopen('emg01.adc..adc','r');
text=fread(fid,2048,'char');
d=fread(fid,'int16');
l=3200;                     % length of a lead
%-------------------------------------------------------------------------------------------
Lemg=d(1:l) ;               %Similar to ECG signals, first 3200 samples are for LEMG

Remg=d(l+1:2*l);
C1emg=d(2*l+1:3*l);
C2emg=d(3*l+1:4*l);
C3emg=d(4*l+1:5*l);
C4emg=d(5*l+1:6*l);
C5emg=d(6*l+1:7*l);
C6emg=d(7*l+1:8*l);
%-------------------------------------------------------------------------------------------
Remg=Remg*0.00488;
Lemg=Lemg*0.00488;
C1emg=C1emg*0.00488;
C2emg=C2emg*0.00488;
C3emg=C3emg*0.00488;
C4emg=C4emg*0.00488;
C5emg=C5emg*0.00488;
C6emg=C6emg*0.00488;
%-------------------------------------------------------------------------------------------
Recgn=    [Recg;0;0]+[0;0;Remg];%+ [0;0;0;0;0;Rp;0;0;0;0;0];  %Signal and noise
are mixed up
Lecgn=    [Lecg;0;0]+[0;0;Lemg];%+ [0;0;0;0;0;Lp;0;0;0;0;0];
C1ecgn=   [C1ecg;0;0]+[0;0;C1emg];%+[0;0;0;0;0;C1p;0;0;0;0;0];
C2ecgn=   [C2ecg;0;0;]+[0;0;C2emg];%+[0;0;0;0;0;C2p;0;0;0;0;0];
```

```
C3ecgn=  [C3ecg;0;0]+[0;0;C3emg];%+[0;0;0;0;0;C3p;0;0;0;0;0];
C4ecgn=  [C4ecg;0;0]+[0;0;C4emg];%+[0;0;0;0;0;C4p;0;0;0;0;0];
C5ecgn=  [C5ecg;0;0]+[0;0;C5emg];%+[0;0;0;0;0;C5p;0;0;0;0;0];
C6ecgn=  [C6ecg;0;0]+[0;0;C6emg];%+[0;0;0;0;0;C6p;0;0;0;0;0];
fid=fopen('Recg.dat','w');  %Records are written in respective files, which will be
read later
fprintf(fid,'%f\n',Recg);
fclose(fid);
fid=fopen('Recgn.dat','w');
fprintf(fid,'%f\n',Recgn);
fclose(fid);
fid=fopen('Remg.dat','w');
fprintf(fid,'%f\n',Remg);
fid=fopen('Lecg.dat','w');
fprintf(fid,'%f\n',Lecg);
fclose(fid);
fid=fopen('Lecgn.dat','w');
fprintf(fid,'%f\n',Lecgn);
fclose(fid);
fid=fopen('Lemg.dat','w');
fprintf(fid,'%f\n',Lemg);
fclose(fid);
fid=fopen('C1ecg.dat','w');
fprintf(fid,'%f\n',C1ecg);
fclose(fid);
fid=fopen('C1ecgn.dat','w');
fprintf(fid,'%f\n',C1ecgn);
fclose(fid);
fid=fopen('C1emg.dat','w');
fprintf(fid,'%f\n',C1emg);
fclose(fid);
fid=fopen('C2ecg.dat','w');
fprintf(fid,'%f\n',C2ecg);
fclose(fid);
fid=fopen('C2ecgn.dat','w');
fprintf(fid,'%f\n',C2ecgn);
fclose(fid);
fid=fopen('C2emg.dat','w');
```

```
fprintf(fid,'%f\n',C2emg);
fclose(fid);
fid=fopen('C3ecg.dat','w');
fprintf(fid,'%f\n',C3ecg);
fclose(fid);
fid=fopen('C3ecgn.dat','w');
fprintf(fid,'%f\n',C3ecgn);
fclose(fid);
fid=fopen('C3emg.dat','w');
fprintf(fid,'%f\n',C3emg);
fclose(fid);
fid=fopen('C4ecg.dat','w');
fprintf(fid,'%f\n',C4ecg);
fclose(fid);
fid=fopen('C4ecgn.dat','w');
fprintf(fid,'%f\n',C4ecgn);
fclose(fid);
fid=fopen('C4emg.dat','w');
fprintf(fid,'%f\n',C4emg);
fclose(fid);
fid=fopen('C5ecg.dat','w');
fprintf(fid,'%f\n',C5ecg);
fclose(fid);
fid=fopen('C5ecgn.dat','w');
fprintf(fid,'%f\n',C5ecgn);
fclose(fid);
fid=fopen('C5emg.dat','w');
fprintf(fid,'%f\n',C5emg);
fclose(fid);
fid=fopen('C6ecg.dat','w');
fprintf(fid,'%f\n',C6ecg);
fclose(fid);
fid=fopen('C6ecgn.dat','w');
fprintf(fid,'%f\n',C6ecgn);
fclose(fid);
fid=fopen('C6emg.dat','w');
fprintf(fid,'%f\n',C6emg);
fclose(fid);
```

5.3.1.5(B) MATLAB code for filtering noise removal from actual ECG signal

```
% MATLAB Script for filtering
close all;
clear all;
M=1000                    %number of taps
fid = fopen('recg.dat','r');
[x1,count]=  fscanf(fid, '%f',M);
fclose(fid);
%time = 0:0.01:2;
%x1=sin(5*time);
figure(1);
plot(x1);
ylabel(' information signal');
title ('Plot for ECG Signal(Training):');
%d=sin(50*time);
fid = fopen('remg.dat','r');
[d,count]=  fscanf(fid, '%f',M);
fclose(fid);
figure(2);
plot(d);
ylabel('noise:');
title('Plot for EMG noise signal(Training):');
x1=x1';
d=d';
x2=1.0*x1+ 8*d;
x3=1.1*x1+ 7*d;
x4=1.2*x1+ 6*d;
x5=1.3*x1+ 4*d;
x6=1.4*x1+ 5*d;
figure(3)
plot(x2);
ylabel('noisy signal');
title ('Plot for noisy signal(Training):');
x=[x2;x3;x4;x5;x6]';
r=10000;
c=0.00000000001;
k1=1;
```

```
k2=1;
y=x'*x;
z=eye(5,5)/(r*k1*k2);
p= z+y;
vf= (eye(5,5)/p)*(x'*d');
disp(d);
s=(x*vf)'-d;
figure(4)
plot(s);
ylabel('Filtered recovered signal');
title('plot for recovered signal(Training):');
%-------------------------------------------------------------------------------------------
%Testing:
fid = fopen('lecg.dat','r');
[x1,count]= fscanf(fid, '%f',M);
fclose(fid);
figure(5);
plot(x1);
ylabel('information signal');
title('Plot for ECG Signal(Testing):');
fid = fopen('lemg.dat','r');
[d,count]= fscanf(fid, '%f',M);
 fclose(fid);
figure(6);
plot(d);
ylabel('noise:');
title('Plot for EMG noise signal(Testing):');
x1=x1';
d=d';
x2=1.0*x1+ 8*d;
x3=1.1*x1+ 7*d;
x4=1.2*x1+ 6*d;
x5=1.3*x1+ 4*d;
x6=1.4*x1+ 5*d;
figure(7)
plot(x2);
ylabel('noisy signal');
title('Plot for noisy signal(Testing):');
```

```
x=[x2;x3;x4;x5;x6]';
r=10000;
c=0.00000000001;
k1=1;
k2=1;
s=(x*vf)'-d;    %LS segment
figure(8)
plot(s);
ylabel('Filtered recovered signal');
TITLE('plot for recovered signal(Testing):');
```

5.3.1.6 Results of filtering

The results of LS filtering algorithm using are shown in the **figures 5.6**

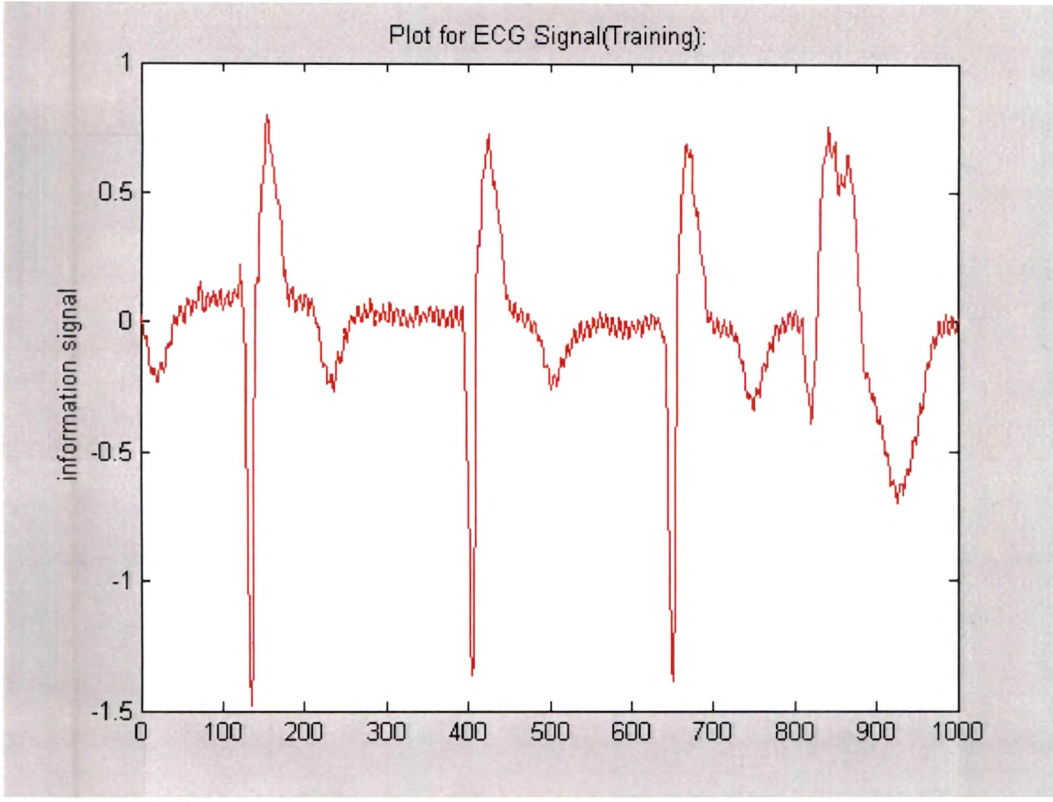

Figure 5.6 (a) ECG signal used for training

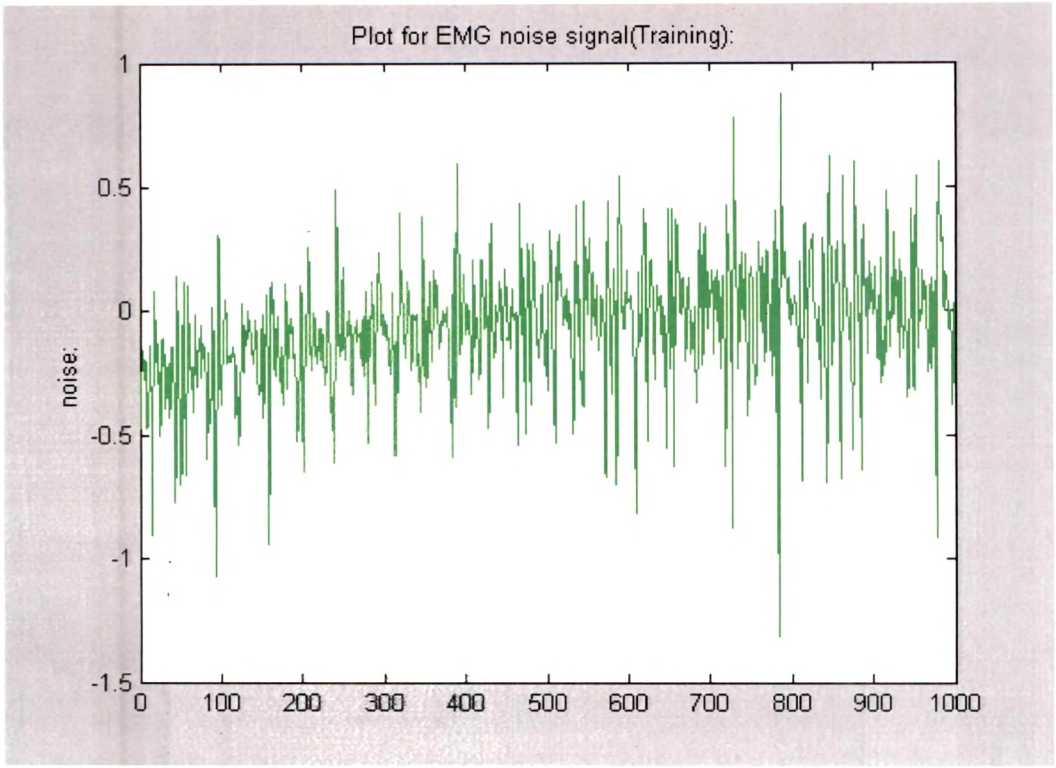

Figure 5.6 (b) Noise signal used to train Hopfield NN

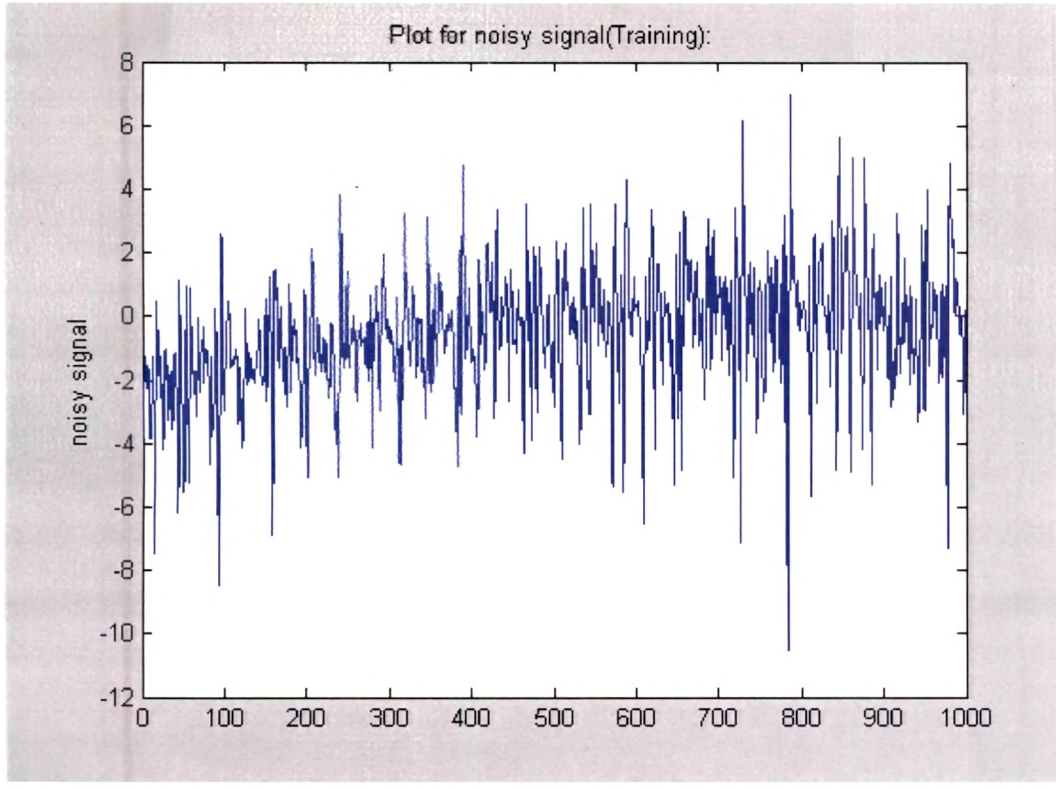

Figure 5.6 (c) Corrupted ECG signal (Actual input): Training phase

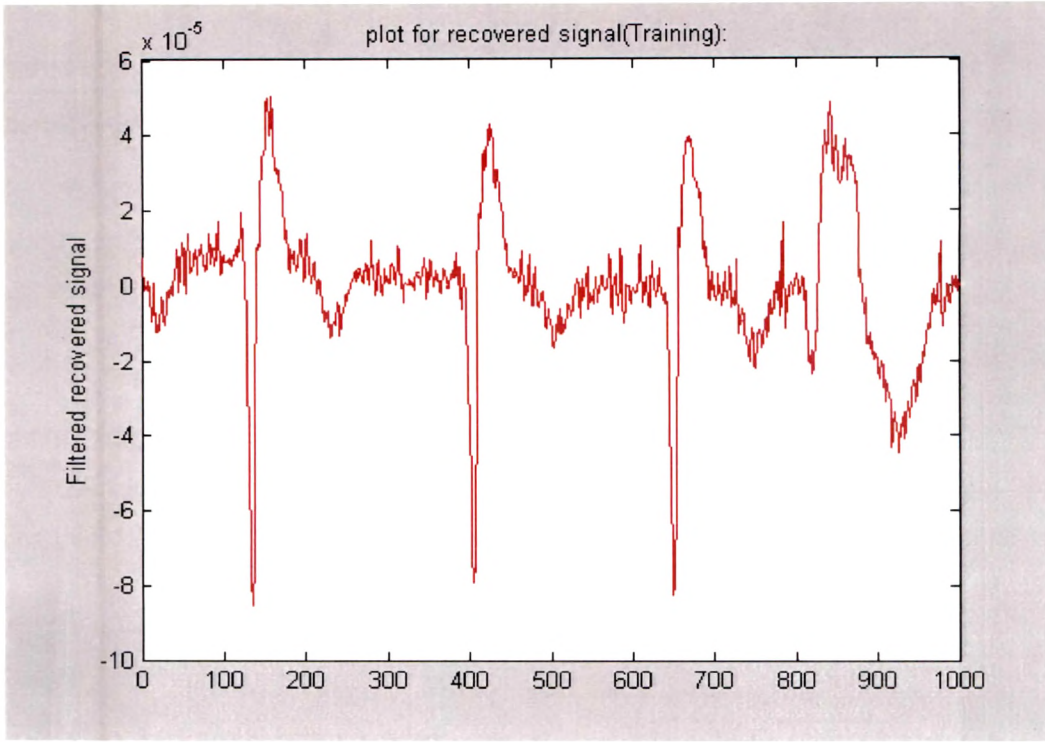

Figure 5.6 (d) Filtered ECG (Output): Training phase

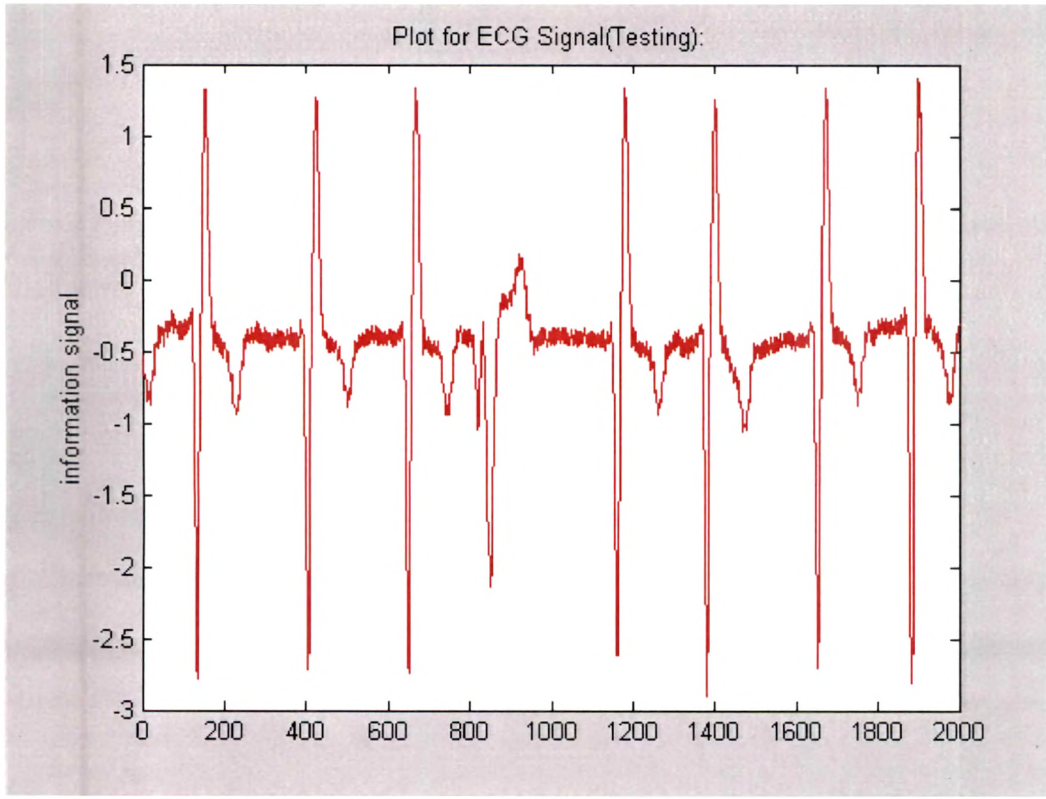

Figure 5.6 (e) ECG signal used for testing

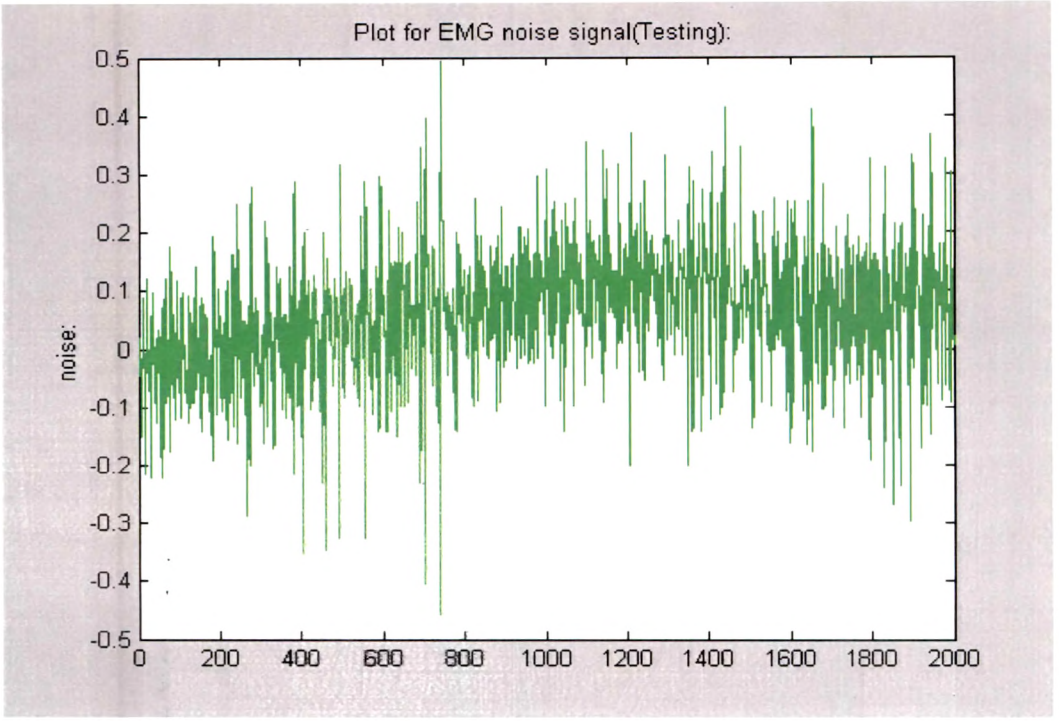

Figure 5.6 (f) Noise signal used to test Hopfield NN

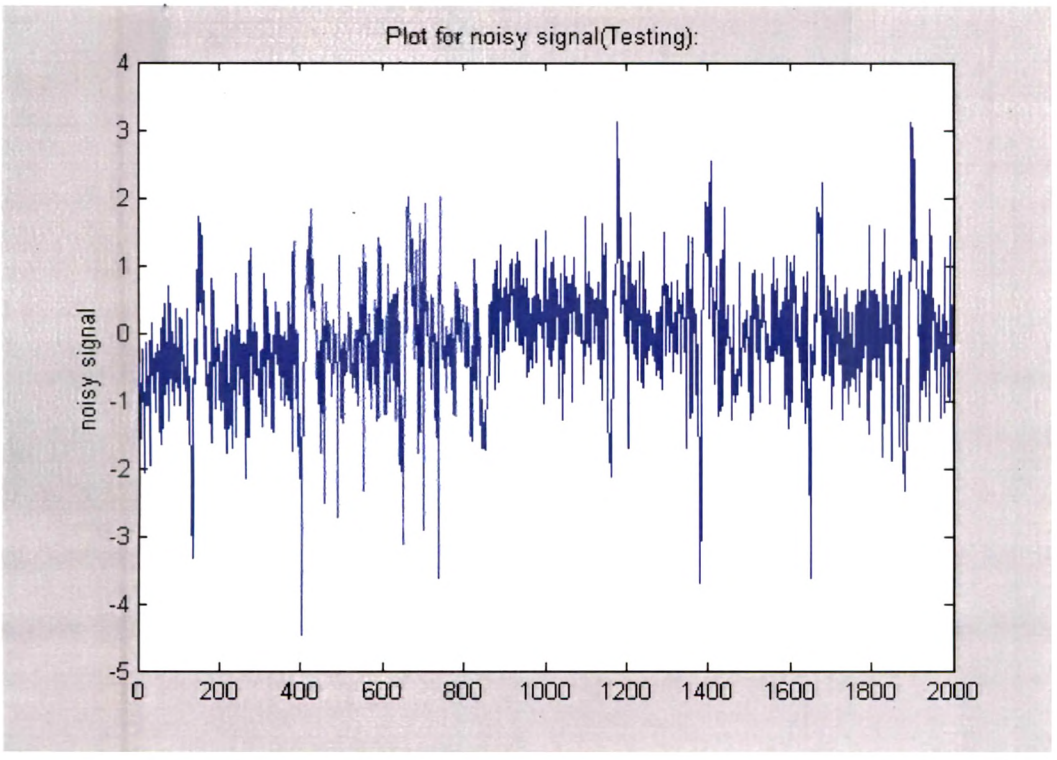

Figure 5.6 (g) Corrupted ECG signal (Actual input): Testing phase

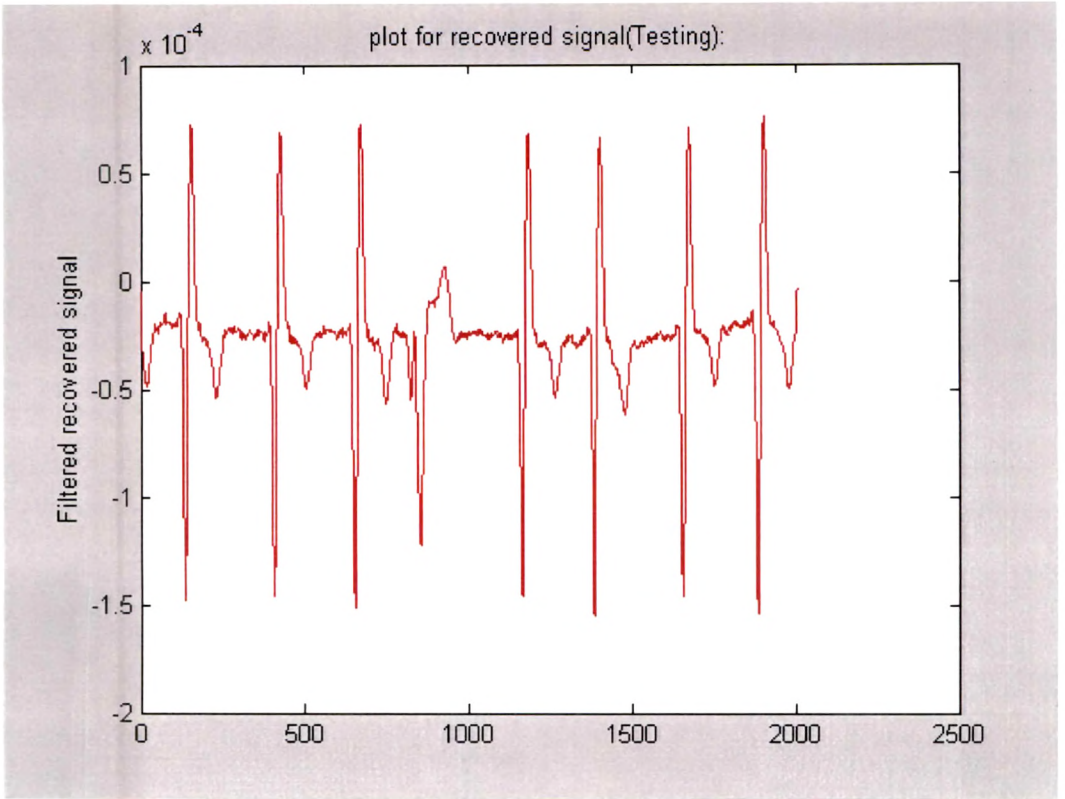

Figure 5.6 (h) Filtered ECG (output): Testing phase

Figure 5.6 Training and testing waveforms for ECG signal filtering using Hopfield NN

5.3.2 Hopfield Neural Network based filtering (Recursive LS Algorithm)

The recursive least-squares (RLS) algorithm is used to minimize **recursively** the time-indexed sum of squares. This means that the available samples in the RLS algorithm are **time-increasing.** The LS solution of (5.13) **at time n** is:

$$W_{LS}(n) = X^{+}{}_{n}d_{n} = \lim_{\alpha \to 0}(X_n^T X_n + \alpha I)^{-1} X_n^T d_n \quad \text{.....(5.22)}$$

Where

$$Xn = \begin{pmatrix} x_1(1) & x_2(1) & \ldots\ldots\ldots & x_N(1) \\ x_1(2) & x_2(2) & \ldots\ldots\ldots & x_N(2) \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ x_1(n) & x_2(n) & \ldots\ldots\ldots & x_N(n) \end{pmatrix} \qquad dn = \begin{pmatrix} d(1) \\ d(2) \\ . \\ . \\ d(n) \end{pmatrix}$$

At time n+1, the LS solution of (5.13) is

$$W_{LS}(n+1) = X_{n+1}^{+} d_{n+1} = \lim_{\alpha \to 0} (X_{n+1}^{T} X_{n+1} + \alpha I)^{-1} X_{n-1}^{T} d_{n+1} \quad \text{.....(5.23)}$$

And the corresponding sample matrix and vector become

$$X_{n+1} = \begin{pmatrix} x_1(1) & x_2(1) & \ldots\ldots\ldots & x_N(1) \\ x_1(2) & x_2(2) & \ldots\ldots\ldots & x_N(2) \\ . & . & & . \\ . & . & & . \\ x_1(n) & x_2(n) & \ldots\ldots\ldots & x_N(n) \\ x_1(n+1) & x_2(n+1) & \ldots\ldots\ldots & x_N(n+1) \end{pmatrix} \quad \text{.....(5.24)}$$

$$d_n = \begin{pmatrix} d(1) \\ d(2) \\ . \\ . \\ d(n) \\ d(n+1) \end{pmatrix} \quad \text{.....(5.25)}$$

One can write (5.24) and (5.25) in the following form

$$X_{n+1} = \begin{bmatrix} X_n \\ z_{n+1} \end{bmatrix} d_{n+1} = \begin{bmatrix} d_n \\ d(n+1) \end{bmatrix} \quad \text{.....(5.26)}$$

Where $Z_{n+1} = [x_1(n+1), x_2(n+1), \ldots, x_N(n+1)]$.

Based on (5.26) fast and efficient RLS can be proposed. These algorithm can provide the LS solution $W_{LS}(n+1)$ at time n+1 by using only $W_{LS}(n)$, X_{n+1}, and d(n+1). Although these RLS algorithm have greatly reduced the computational complexity of solving equation(5.22), the computation required in these RLS algorithm is still intensive, and hence it is difficult to provide $W_{LS}(n+1)$ **in real time.**

Let there be continuos time Hopfield neural network in **figure 5.7**, having neurons with a linear input-output relationship function $g(u) = K_1 u$

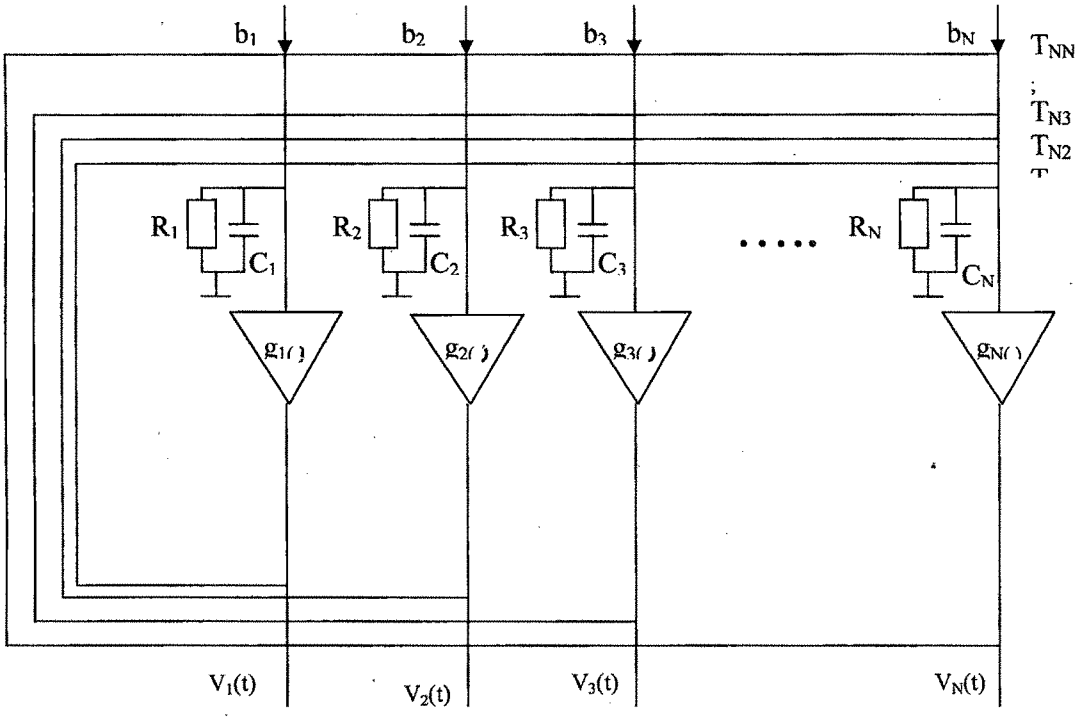

Figure 5.7 Hopfield Model for recursive least square algorithms

At time n, the connection strengths and bias currents of the network are computed from network are computed from the expression

$T_n = X_n^T X_n$ and

$B_n = -X_n^T d_n$ .....(5.27)

Lyapunov stability theorem can be used to prove the stability of this neural network. The output vector $V_{n,f}$ in the stationary state is computed by letting dU(t)/dt = 0, that is,

$$V_{n,f} = \lim_{t \to \infty} V(t) = \left(\frac{1}{RK_1} I + X_n^T X_n\right)^{-1} X_n^T d_n, \quad \text{.....(5.28)}$$

Which is an approximation of (5.22). Consequently, at time n+1, the connection strengths and bias currents of the network are computed by

$T_{n+1} = X_{n+1}^T X_{n+1}$ and
$B_{n+1} = -X_{n+1}^T d_{n+1}$, ...(5.29)

And the network provides the solution:

$$V_{n+1,f} = \lim_{t \to \infty} V(t) = \left(\frac{1}{RK_1} I + X_{n+1}^T X_{n+1}\right)^{-1} X_{n+1}^T d_{n+1}, \quad \text{.....(5.30)}$$

Which approximates the exact solution (5.23).

Equation (5.30) is an approximation of the RLS solution **W**. As the implementation of Algorithm is in real time, with no other difference, we will have same code and waveform quality for filtered ECG. Hence algorithm remains same as for LS Solution.

5.3.3 Multilayer Perceptron (MPLNN) based Model for filtering

5.3.3.1 Introduction

In order to successfully remove noise from biomedical signals, adaptive filter should be implemented to track the noise signal characteristics. Most adaptive noise removal techniques are based on the least mean squares (LMS) principal or on the recursive least square (RLS) principal.

Electromyogram (EMG) is one type of artifact, which contaminate ECG signal resulting into considerable amount of reduction in sharp Q, R and S ECG wave amplitudes. Even with the high accuracy of the instrument, EMG is a very common signal produced from disable persons having to keep on maintaining a position in body posture or in children or in shivering patients. From the **Table 5.1** it can be seen that there is a considerable overlapping of frequency spectra of ECG and EMG. Because of considerable overlap of the frequency spectra of ECG and EMG artifacts, it is difficult to remove EMG artifacts from ECG signal using solely traditional linear adaptive filters. So it is required to remove the artifacts by other nonlinear filtering method.

Table 5.1: Comparison: Amplitude and Frequency Range of ECG and EMG

Signal	Amplitude range	Frequency range
Electrocardiogram (ECG)	10 μV – 5mV	0.05 – 500 Hz
Electromyogram (EMG)	25 - 5000μV	5 – 2000 Hz

Filtering of EMG can be carried out by constructing adaptive filtering by means of a Multi Layer Perceptron (MLP) Artificial Neural Network (ANN). Such filter is constructed here with varying number of input nodes depending on the phase difference between separately recorded EMG artifacts and part of EMG artifact mixed with the ECG signal. This is done based on measurement of cross correlation between noisy ECG and EMG. This makes this Algorithm work for providing very robust solution.

It is utilizing present sample and number of past samples to be given to input layer of MLPANN to generate present time estimate of the artifact value present in corrupted ECG signal. This way MLPANN works as **temporal processing unit.**

Noise removal is complicated by the fact that the characteristics like local spectrum of all biomedical signals vary with time. For ECGs, it varies quasi periodically, with each period corresponding to one heartbeat. Shape of ECG varies slowly or abruptly in time. For successful noise removal for biomedical signals, procedure must be adaptive to track the changing signal characteristics. Due to considerable overlap of the frequency spectra of ECG and EMG artifacts, it is difficult to remove EMG artifacts from ECG signal using solely traditional linear adaptive filters.

In the frequency range of biomedical signals such as ECGs, the body can be treated as a volume conductor that is, purely resistive. Therefore, an EMG signal originating in the arm, for example, is instantaneously superimposed on a signal originating in the heart. However, the difference of phase can occur because of transmission through different conductor paths for ECG and EMG signals. So such linear filter is only partially successful because of phase difference between separately recorded EMG and part of EMG mixed with ECG. It is required to remove the artifacts by other nonlinear, supervised adaptive filtering method.

5.3.3.2 Filter Design

A nonlinear adaptive filter (**Figure 5.8**) can be constructed considering noisy ECG as primary input to the filter and separately recorded EMG channel as reference input to the filter and the difference between primary signal and estimate of noise signal derived from reference signal is made minimum. The resultant output is desired pure ECG signal.

Figure 5.9 shows architecture of MPLNN used for implementing a nonlinear filter.

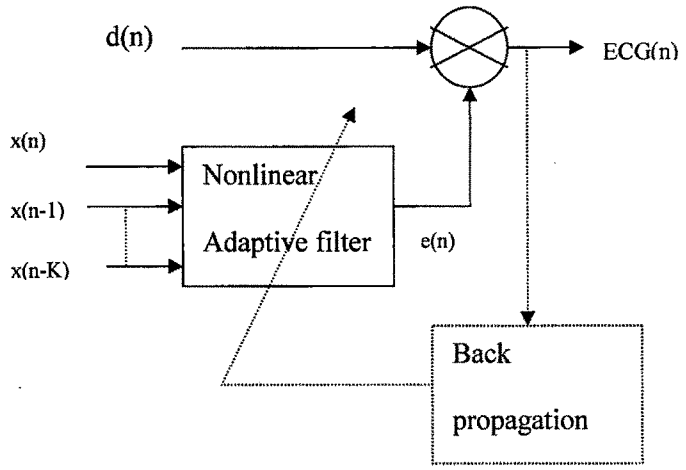

Figure 5.8 Block diagram for MPLNN Based Adaptive filter (Dotted portion shows training phase using Back propagation algorithm)

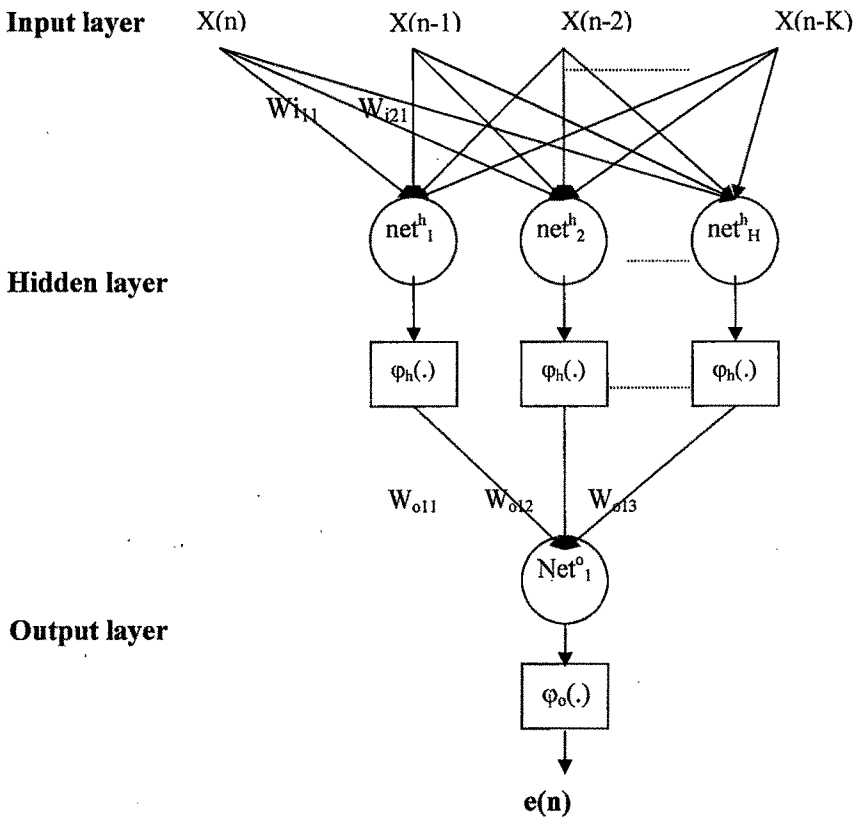

Figure 5.9 MLPANN to estimate noise value at time n, with K+1 number of input nodes, H number of hidden nodes and one output node

The units at the top in **Figure 5.9** are input units. Input layer is given total K+1 number of samples from recorded EMG. Out of which K number of past samples at sampling time n-1, n-2... n-K and a present sample at time instant n are taken, to predict the part of EMG artifact that is present in the ECG signal at present instant n of time. Intermediate level is hidden layer and units at the bottom in **figure 5.9** is output units. All the units at the lower layer are connected to each unit at the upper layer by the connection strength called weights. All the weights are initialized to small random values at the beginning. MLPANN becomes a temporal processing unit, with its taps connected to the synapses of a neuron. Outputs of the neuron i in hidden layer at time instant n can be described as:

$$y_i(n) = \varphi_h(v_i(n)) \quad \text{.....(5.31)}$$

$$v_i(n) = \sum_{l=0}^{K} w_{il}(l)x(n-l) + b_i) \quad \text{.....(5.32)}$$

Similarly, output of the j^{th} output node is,

$$e_j(n) = = \varphi_o(y_j(n)) \quad \text{.....(5.33)}$$

Where

$$y_j(n) = \sum_{l=1}^{H} W_{jl}y_l + b_j \quad \text{.....(5.34)}$$

Where $\varphi_h(.)$ and $\varphi_o(.)$ are the activation functions of neuron in hidden and output layer respectively. The activation function for the hidden as well as output layer is hyperbolic tangent which is one type if sigmoid nonlinear function, utilized to introduce non-linearity. The range of the function is (-1,1) so that it can represent output signal of both polarity. b_i and b_j are called bias input at i^{th} and j^{th} nodes respectively.

The expression for the activation function for the neuron i in hidden layer is:

$$\varphi_h(\mathbf{W_i}, \mathbf{X_i}, b_i) =$$

$$\frac{1 - \exp(-\mathbf{WiXi} - b_i)}{1 + \exp(-\mathbf{WiXi} - b_i)} \quad \text{.....(5.35)}$$

Where

$\mathbf{X_i}$: Input vector of size **IXM**

I: Number of input layer neurons

M: Number of test patterns for training of MLPANN

$\mathbf{W_i}$: Matrix for Synaptic weights for input layer size: **HXI**

H: Number of hidden layer neurons.

$\mathbf{b_i}$ is the bias term for i^{th} hidden node

Similarly, the expression for the activation function for output layer neuron is:

$\varphi_o(\mathbf{Wo}, \mathbf{Xo}, b_o) =$

$$\frac{1 - \exp(-\mathbf{WoXo} - b_o)}{1 + \exp(-\mathbf{WoXo} - b_o)} \quad \text{.....(5.36)}$$

Where

Xo: Output vector of size **HXM**

Wo: Matrix for Synaptic weights for hidden layer size: **OXH**

O: Number of output layer neurons (Here 1).

$\mathbf{b_o}$: bias term for o^{th} output node.

5.3.3.3 Back propagation algorithm

is used as training algorithm for the MLPANN with I number of input layer nodes, H number of hidden nodes and one output node. It is a gradient descent algorithm designed to minimize the mean square error between the actual computed output and the desired output. Training by the back propagation algorithm consists of two passes:

(1) During propagation pass, the input for output is known is presented to the network and multiplied by the weights. All these weighted inputs to each unit of the next layer (hidden layer) are summed and produce output according to the activation function. These outputs of hidden layer become input to the output layer. They are multiplied by weights and weighted inputs are summed and produce output according to activation function.

(2) Outputs at the output layer are compared to desired output resulting error signals. During backward pass (As shown dashed in the **figure 5.8**, the error signals are propagated backward through each unit of the network in output as well as hidden layer, and appropriate weight changes are made according to gradient descent technique, as described by following equations:

Weight Update for jth output node:

Weight update equation at time n+1 is:

$$W_{ji}(n+1) = W_{ji}(n)+\eta\delta_j(n)y_i(n) +\alpha\Delta W_{ji}(n-1) \quad \text{.....(5.37)}$$

$$\text{Where } \delta_j(n) = \phi_j'(v_j(n))e_j(n) \quad \text{.....(5.38)}$$

Where

$\phi_j'(v_j(n))$: is the derivative of the activation function at the input of the j^{th} node $v_j(n)$.

$e_j(n)$: is the difference between actual output and expected output at the j^{th} node, at instant n.

$y_i(n)$: is the input from i^{th} hidden neuron to the neuron j at output layer at time instant n.

α : Momentum factor

η : Learning rate

Similarly, Weight Update for i^{th} hidden node:

Weight update equation at time n is:

$$w_{ik}(n+1) = W_{ik}(n) + \eta\delta_i(n)x_k(n) +\alpha\Delta W_{ji}(n-1) \quad \text{.....(5.39)}$$

Where

$$\delta_i(n) =\phi_i'(v_i(n))\sum_{j=1}^{O}\delta_j(n)W_{ji}(n) \quad \text{.....(5.40)}$$

Where

$\phi_j'(v_j(n))$: is the derivative of the activation function at the input of the j^{th} node $v_j(n)$.

$\delta_j(n)$: is value of δ calculated at j^{th} output node .

$W_{ji}(n)$: is the value of synaptic weight from i^{th} hidden node to j^{th} output node.

x_k : input from the k^{th} input node.

5.3.3.4 Training set

Input to the MLPANN is K+1 number of noise (EMG) signal samples. The output is compared with present sample of noisy ECG. The difference is taken as error signal, for computation of weight update equation. Number of past samples to consider (K) and hence number of input layer nodes (I = K + 1) is variable and is automatically

decided by the MATLAB program depending upon the relative phase difference between EMG and part of EMG mixed in desired ECG signal.

Number of patterns (M) for training and number of time epochs (T) are selected as different values and results were noted down.

5.3.3.5 MATLAB Code

```
%-------------------------------------------------------------------------------
%TRAINING PHASE:
%-------------------------------------------------------------------------------
close all;
clear all;
N=10;                %reading data to decide correlation
M=1000;              %Training sets
 H=2;                %hidden nodes
 O=1;                %output nodes
 TST=3000;           %Testing sets
 Time=50;            %No. of Epochs
 eta= 0.1;           %Learning Rate
fid = fopen('remg.dat','r'); %Reading recgm and remg to decide phase difference
in terms of %number of samples
[X1,count]= fscanf(fid, '%f',N);
fclose(fid);
fid = fopen('recgn.dat','r');
[X2,count]= fscanf(fid, '%f',N);
fclose(fid);
C2=xcorr(X1,X2);
figure
plot(C2);grid
tmp = 0;
i2=0;
indx=0;
for n = 1: 2*N-1
  i2=i2+1;
  if tmp< C2(i2)
    tmp = C2(i2);
    indx2=i2;
  end
end
```

```
indx2=abs(indx2-N);
I=indx2+1;              % Number of input nodes = phase difference +1
 xbi=ones(I,M);
 xbo=ones(H,M);
 e1=1;
 e2=1;
 %-------------------------------------------------------------------------------------
rand('seed',0);
Wi=2.0*rand(H,I)-1.0;
Wo=2.0*rand(O,H)-1.0;
%Wi=0.5*ones(H,I);%Wi=eye(H,I);
%Wo=0.5*ones(O,H);%Wo=eye(O,H);
Wbi=zeros(H,I);
Wbo=zeros(O,H);              %For biasing
%-------------------------------------------------------------------------------------
fid = fopen('remg.dat','r');              %reading  training data
[X,count]=  fscanf(fid, '%f',M+I);
for j =1 :M
  for i = j:I+j-1
    n=i-(j-1);
    Up(n,j)= X(i,1);
  end
  end              %Up is training input to the MLP
   fclose(fid);
 fid = fopen('recgn.dat','r');
[X1,count]=  fscanf(fid, '%f',M+I);
 for j =1:M
   Yp(1,j)= X1(j+I-1,1);%+1 for next sample
   end
fclose(fid);
%-------------------------------------------------------------------------------------
% Reading one more data set for testing:Up1:
%-------------------------------------------------------------------------------------
fid = fopen('c1emg.dat','r');
[X2,count]=  fscanf(fid, '%f',TST+I);
for j =1 :TST
  for i = j:I+j-1
    n=i-(j-1);
```

```
    Up1(n,j)= X2(i,1);
     end
end
 fclose(fid);
fid = fopen('c1ecgn.dat','r');
[X3,count]=  fscanf(fid, '%f',TST+I);
for j =1:TST
Yp1(1,j)= X3(j+I-1,1);%+1 for next sample
end
fclose(fid);
%-------------------------------------------------------------------------------------------------
 MSE=zeros(Time,1);
 momo(O,H)=0;
 momi(H,I)=0;
alfa=0;
for T=1:Time
  T
  for k = 1:M  %no. of cols of datae
    xbii=xbi(:,k);
    xboo=xbo(:,k);
    di=Wi;
    do=Wo;
    x=Up(:,k);
     x1=(1.0-exp(-e1*Wi*x-e2*Wbi*xbii))./(1.0+exp(-e1*Wi*x-e2*Wbi*xbii));
    y1=(1.0-exp(-e1*Wo*x1-e2*Wbo*xboo))./(1.0+exp(-e1*Wo*x1-
e2*Wbo*xboo));
     erro=Yp(:,k)-y1;
     tmp = sum(erro.*erro);
     MSE(T,1)= MSE(T,1)+tmp;
     diffo=(1.0-y1.*y1);
    do =  0.5* erro.*diffo; %
    diffi=(1.0-x1.*x1);
   d11=Wo'*do;
   d1=d11.*diffi;
    for k1=1:H
      Wo(:,k1)=Wo(:,k1)+eta*do.*x1(k1,1)+ momo(:,k1);
      %Wbo(:,k1)=Wbo(:,k1)+eta*do.*x1(k1,1);
    end
```

```
   for k1=1:I
      Wi(:,k1)=Wi(:,k1)+eta*d1*x(k1,1)+momi(:,k1);
     %Wbi(:,k1)=Wbi(:,k1)+eta*d1*x(k1,1);
    end
    momo=alfa*(Wo-do);
    momi=alfa*(Wi-di);
  end
 %MSE = MSE/M;
end
%-------------------------------------------------------------------------------------------
%Test the MLP WITH SAME INPUT PATTERNS Up:
%-------------------------------------------------------------------------------------------
fid = fopen('recgn.dat','r');
[X1,count]= fscanf(fid, '%f',M+I);
 for j =1:M
   Yp(1,j)= X1(j+I-1,1);%+1 for next sample
   end
fclose(fid);
for k = 1:M
  u = Up(:,k);
  xbii=xbi(:,k);
   xboo=xbo(:,k);
  x1=(1.0-exp(-e1*Wi*u-e2*Wbi*xbii))./(1.0+exp(-e1*Wi*u-e2*Wbi*xbii));
  y1=(1.0-exp(-e1*Wo*x1-e2*Wbo*xboo))./(1.0+exp(-e1*Wo*x1-e2*Wbo*xboo));
  Yn(:,k)=y1;
end
 fid = fopen('recg.dat','r');
[X4,count]= fscanf(fid, '%f',M+I);
 for j =1:M
    ecgip(1,j)= X4(j+I-1,1);  %for next sample
   end
   fclose(fid);
k=0;
for i=1:M
  for j=1:O
    k=k+1;
    x1(k)=Up(j,i);              %noise:EMG input
    d11(k)=Yp(1,i);
```

```
    dexp(k)=Yn(j,i);   %Expectation of noise in ECG available at ANN output
   act_ecg(k)=ecgip(1,i);
 end
end
filterop=d11'-dexp;
x11 = [0 0 0 0 0 filterop];
x22 = [0 0  0  0 filterop 0];
x33 = [0 0 0 filterop 0 0];
x44 = [0 0 filterop 0 0 0];
x55 = [0 filterop 0 0 0 0 ];
x66=  [filterop 0 0 0 0 0 ];
y = (x11 + x22 + x33 +x44 +x55 + x66)/6;
y=y(1:M);
l=1:k;
%perr=(act_ecg-y)./act_ecg*100;
perr=(act_ecg-filterop)/act_ecg*100;
figure(1)
plot(MSE);
ylabel('Mean Square error);
title('MSE during Training'');
figure(2)
title('WAVEFORMS DURING TRAINING');
subplot(2,1,1);plot(l,x1);grid
ylabel('Noise');
subplot(2,1,2);plot(l,perr);grid
ylabel('Percent Error');
figure(3)
title('WAVEFORMS DURING TRAINING');
subplot(4,1,1);plot(Yp);
ylabel('Noisy ECG');grid
subplot(4,1,2);plot(l,act_ecg);grid
ylabel('Expected output');
subplot(4,1,3);plot(l,filterop);grid
ylabel('output of the filter');
subplot(4,1,4);plot(l,y);grid
ylabel('averaged & improved');

%-------------------------------------------------------------------------------------------
```

%Test the MLP WITH DIFFERENT INPUT PATTERNS Up1:

```
%-------------------------------------------------------------------------------------------
fid = fopen('c1ecgn.dat','r');
[X3,count]=  fscanf(fid, '%f',TST+I);
 for j =1:TST
 Yp1(1,j)= X3(j+I-1,1);%for next sample
end
fclose(fid);
xbi=ones(I,TST);
 xbo=ones(H,TST);
for k = 1:TST
 xbii=xbi(:,k);
  xboo=xbo(:,k);
u = Up1(:,k);
  x1=(1.0-exp(-e1*Wi*u-e2*Wbi*xbii))./(1.0+exp(-e1*Wi*u-e2*Wbi*xbii));
  y1=(1.0-exp(-e1*Wo*x1-e2*Wbo*xboo))./(1.0+exp(-e1*Wo*x1-e2*Wbo*xboo));
  Yn1(:,k)=y1;
 end
 fid = fopen('c1ecg.dat','r');
[X5,count]=  fscanf(fid, '%f',TST+I);
  for j =1:TST
  ecgip1(1,j)= X5(j+I-1,1);%for next sample
   end
 fclose(fid);
k=0;
for i=1:TST
  for j=1:O
    k=k+1;
    x2(k)=Up1(j,i);%noise:EMG input
    d22(k)=Yp1(1,i);
    dexp1(k)=Yn1(j,i); %Expectation of noise in ECG available at ANN output
   act_ecg1(k)=ecgip1(1,i);
 end
end
filterop1=d22-dexp1;
x11 = [0 0 0 0 0 filterop1];%
x22 = [0 0  0  0 filterop1 0];
x33 = [0 0 0 filterop1 0 0];
```

```
x44 = [0 0 filterop1 0 0 0];
x55 = [0 filterop1 0 0 0 0 ];
x66= [filterop1 0 0 0 0 0 ];
y1 = (x11 + x22 + x33 +x44 +x55 + x66)/6;
y1=y1(1:TST);
l=1:k;
perr1=(act_ecg1-filterop1)./act_ecg1*100;
figure(4)
title('WAVEFORMS DURING TESTING');
subplot(2,1,1);plot(l,x2);grid
ylabel('Noise');
subplot(2,1,2);plot(l,perr1);grid
ylabel('Percent Error');
figure(5)
title('WAVEFORMS DURING TESTING');
subplot(4,1,1);plot(Yp1);
ylabel('noisy ECG');grid
subplot(4,1,2);plot(l,act_ecg1);grid
ylabel('desired output');
subplot(4,1,3);plot(l,filterop1);grid
ylabel('output of filter');
subplot(4,1,4);plot(l,y1);grid
ylabel('averaged & improved');
```

5.3.3.6 Results:

During training as described earlier, value of error is noted down. Also, during training, mean square error is stored as variable and plotted as a function of time epochs. **Figure 5.10** shows how it decreases as number of iterations increases. After training phase is over, the network is tested for 3000 number of samples and results are presented in **figure 5.11. Figure 5.11(a)** shows noisy ECG signal, corrupted by EMG signal. **Figure 5.11(b)** shows the pure ECG. **Figure 5.11(c)** shows the ECG signal after filtering. Though the signal to noise ratio is improved, the quality of the filtered signal is not acceptable from the medical inspection viewpoint. So averaging of the signal is done and averaged signal is shown in **figure 5.11(d). Figure 5.12** shows the amount of error for all 3000 samples.

Filtering using MLP ANN is successfully carried out with phase shifted signal as well as with DC offset available in noise signal.

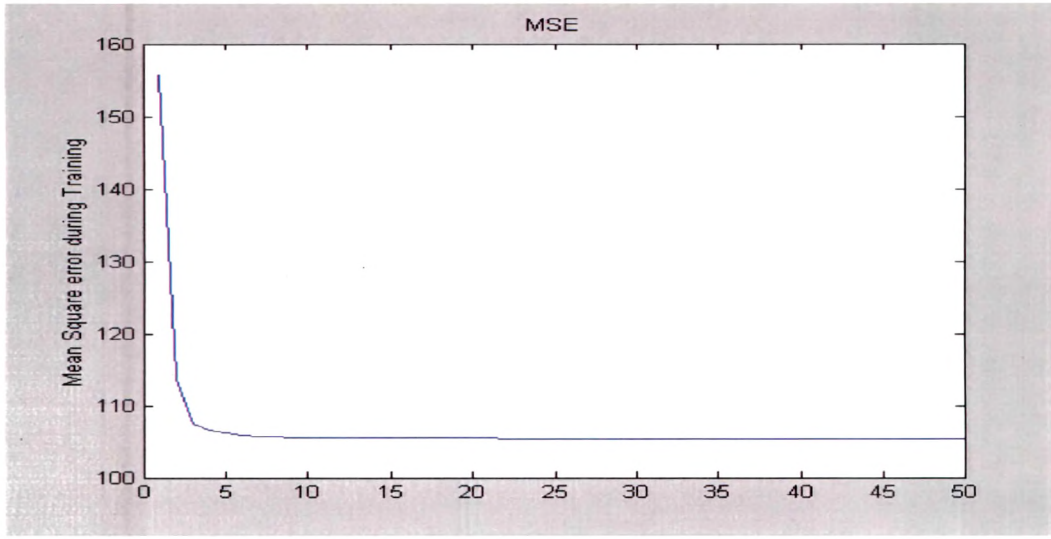

Figure 5.10: Mean square error for time epochs 50 during MLPANN training

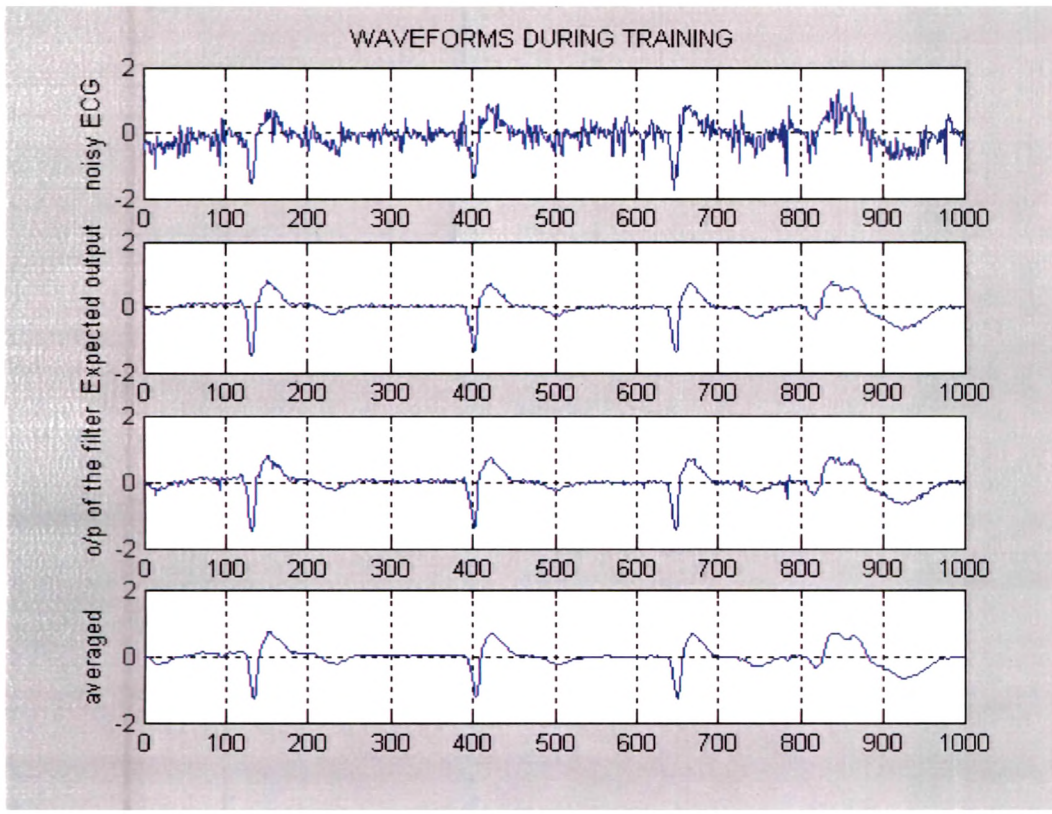

Figure 5.11 Results of the filtering using MLPANN during training (a) noisy ECG samples contaminated by EMG signal (b) Expected ECG (c) Recovered ECG (d) averaged signal by the adaptive filter with number of test inputs M = 1000, number of time epochs, T = 50, I = 3, H = 2, O = 1, eta = 0.1.

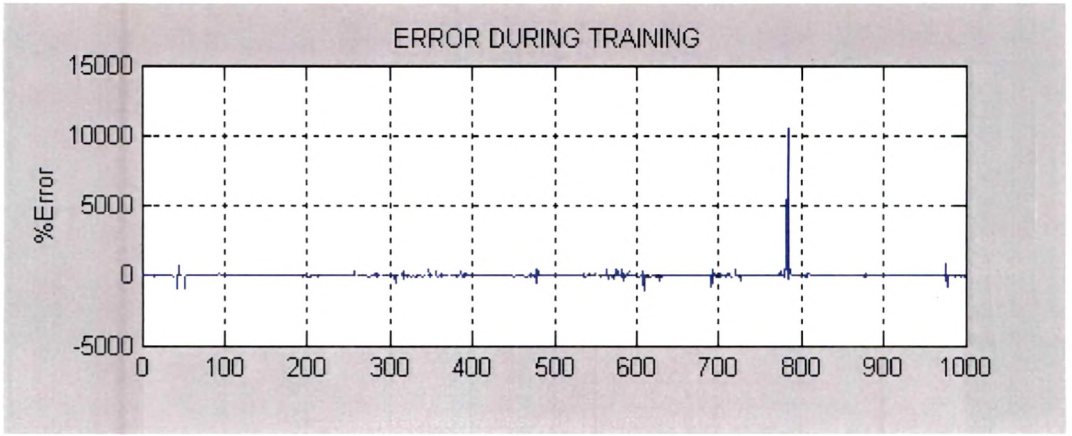

Figure 5.12 Errors associated in filtering using MLPANN during training

Following are the waveforms during testing with a different set of ECGs:

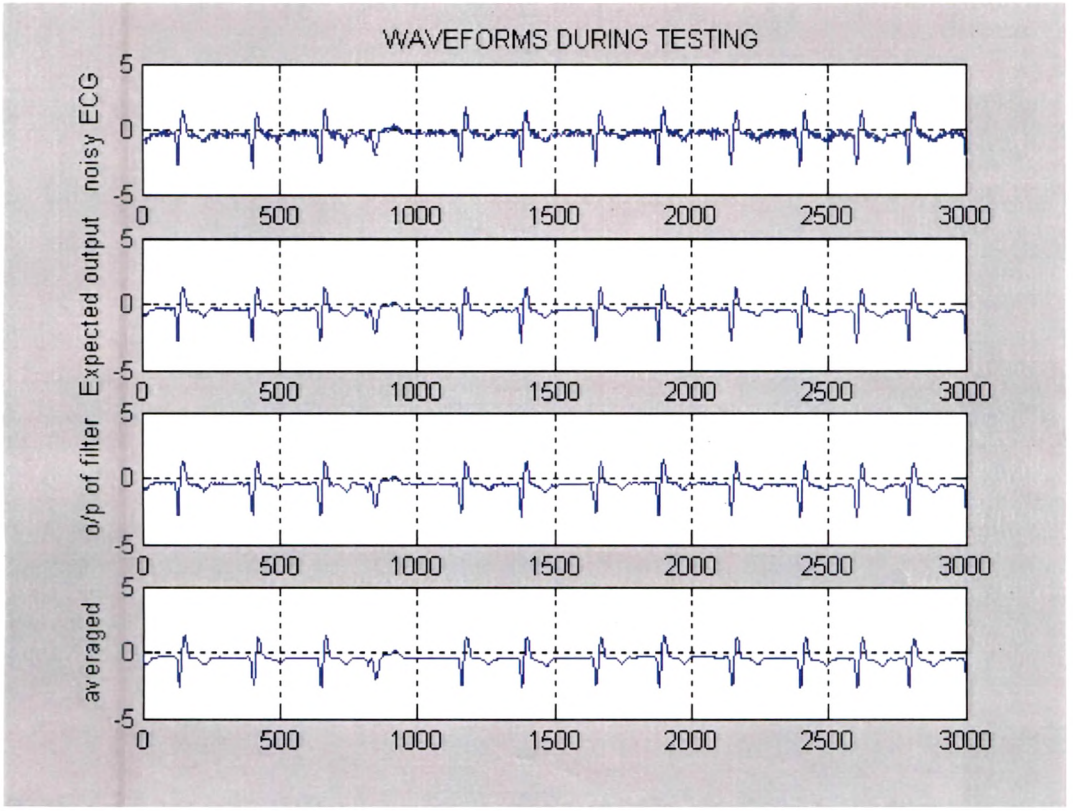

Figure 5.13 Results of the filtering using MLPANN during testing (a) noisy ECG samples contaminated by EMG signal (b) Expected ECG (c) Recovered ECG (d) averaged signal

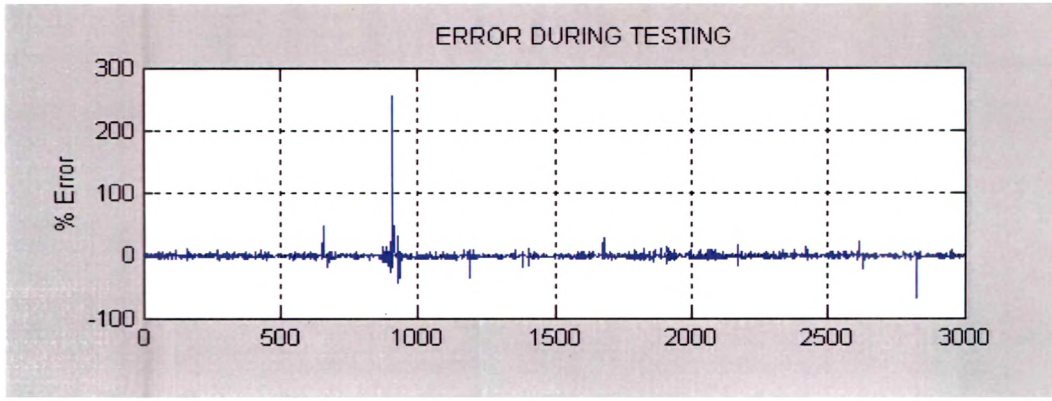

Figure 5.14 Errors in filtering using MLPANN during testing

5.3.4 Radial Basis Function Neural Network for filtering

5.3.4.1 Introduction

RBFNN is designed to perform Input-Output mapping trained by examples ($X^K d^K$), K = 1,2... p. It is based on the concept of locally tuned and overlapping receptive field structure. It is a hybrid network, and uses the hybrid supervised and unsupervised learning scheme. The hidden nodes have Gaussian activation function

$$Z = \varphi_q(x) \quad \frac{R_q(X)}{\sum_K R_q(X)}$$

$$= \frac{\exp [-1 x - m_q I^2 / 2.o_q{}^2]}{\sum_k \exp [-1 x - m_q I^2 / 2.o_q{}^2]} \qquad \text{.....(5.41)}$$

Where x is input vector. The hidden node q gives maximum response to input vectors close to m_q Each hidden nodes q is said to have its own response field $R_q(x)$ in input space, which is centered on m_q with size proportional to o_q., which are mean and variance of q^{th} Gaussian function, which is an example of RBF.

The output of RBFN is the weighted sum of the hidden node output:

$$i = a_i [\Sigma w_{iq}.z_q + \theta_i] \qquad \text{.....(5.42)}$$

Training for RBFNN is carried out by hybrid learning rule. i.e. unsupervised learning in input layer and supervised one in output layer.

5.3.4.2 Filter Design

As an effort to remove this EMG artifact, a nonlinear adaptive filter **Figure 5.15** can be constructed considering noisy ECG as primary input to the filter and separately

recorded EMG channel as reference input to the filter and the difference between primary signal and estimate of noise signal derived from reference signal is made minimum. The resultant output is desired pure ECG signal.

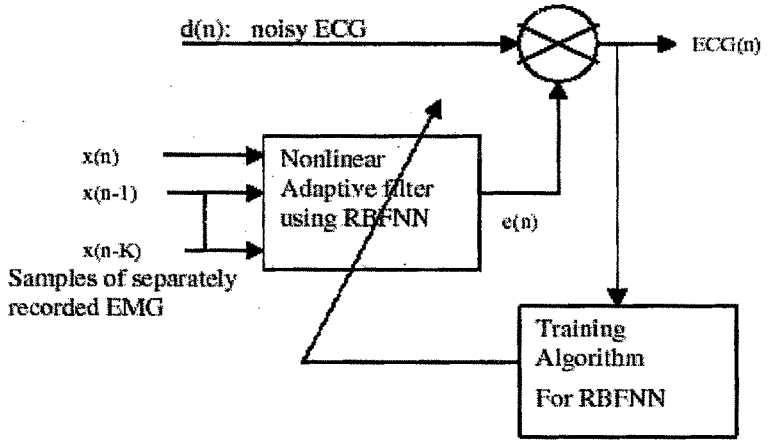

Figure 5.15 Adaptive filter using RBFNN

The RBFNN is used to create temporal unit for nonlinear filter. **Figure 5.16** represents structure used for the application. Input layer is given total (K+1) number of samples from recorded EMG. Out of which K number of past samples at time n-1, n-2... n-K and a present sample at time instant n are taken, to predict the part of EMG artifact that is present in the ECG signal at present instant n of time. Intermediate level is called hidden layer and units at the bottom in **figure 5.16** are called output units, and the layer is called output layer.

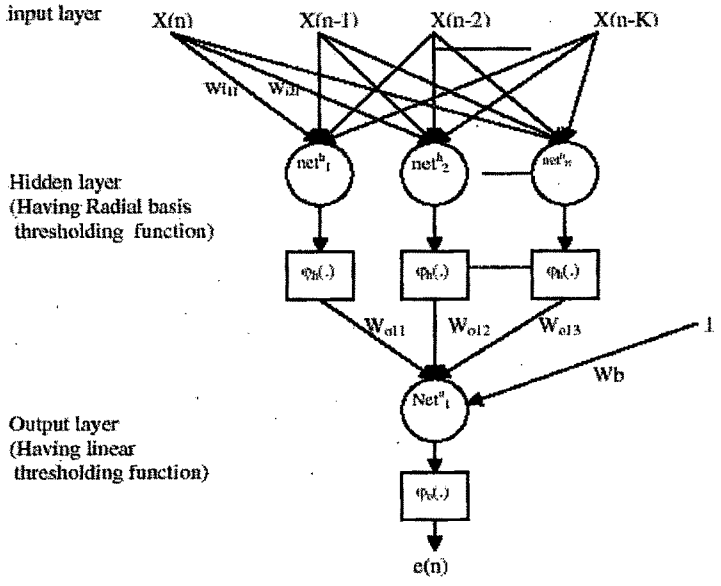

Figure 5.16 RBFNN for filtering

All the units at the lower layer are connected to each unit at the upper layer by the connection strength called weights. All the weights are initialized to small random values at the beginning. This way, this RBFNN becomes a temporal processing unit, with its taps connected to the synapses of a neuron.

5.3.4.3 Training RBFNN

5.3.4.3(A) Training output layer

A generalized delta rule is used for updating weights assuming linear output units.

$$w_{ij} = \eta(d_i - e_i). z_q \quad \text{.....(5.43)}$$

When averaged over p training pairs, it minimizes MSE cost function

$$E(w_{iq}) = \tfrac{1}{2} \sum_{K} \sum_{I} [d_{ik} - \sum_{q} w_{iq}. \varphi_q(x_k)]^2 \quad \text{.....(5.44)}$$

5.3.4.3(B) Training input layer:

Learning involves the determination of centers $[m_q]$, and widths $[w_q]$ for q = 1, using VQ, CPL or SOM approach.

$$m_{closest} = \eta (x - m_{closest}) \quad \text{.....(5.45)}$$

$m_{closest}$ is the respective field closest to input vector. The approach is to adjust all centers with the update value tuned by relative distance via control of effective radius of the receptive fields.

Number of past samples to consider (K) and hence number of input layer nodes (I = K + 1) is variable and is automatically decided by the MATLAB program depending upon the phase difference between EMG and part of EMG mixed in desired ECG signal. "NEWRB" function in Artificial Neural Network toolbox of **MATLAB** is used. NEWRB returns a new radial basis network. This function adds neurons to the hidden layer of a RBFNN basis network until it meets the specified mean squared error goal.

NEWRB creates two layer network. First layer has RADBAS newrons and calculates its weighted inputs with DIST and its net input with NETPROD. Second layer has PURELIN neurons which calculate its weighted inputs with DOTPROD and its net inputs with NETSUM. Both layers have biases.

5.3.4.4 MATLAB Code

```
%-------------------------------------------------------------------
%TRAINING PHASE
%-------------------------------------------------------------------
close all;
clear all;
N=100;
fid = fopen('remg.dat','r');
[X1,count]= fscanf(fid, '%f',N);
fclose(fid);
fid = fopen('recgn.dat','r');
[X2,count]= fscanf(fid, '%f',N);
fclose(fid);
C2=xcorr(X1,X2);
figure
plot(C2);grid
tmp = 0;
i2=0;
indx=0;
for n = 1: 2*N-1
i2=i2+1;
if tmp< C2(i2)
tmp = C2(i2);
indx2=i2;
end
end
indx2=abs(indx2-N);
I=indx2+1;
M=100;          %Training sets
O=1;            %output nodes
tst=3000;       %Testing sets
%-------------------------------------------------------------------
 fid = fopen('remg.dat','r');
[X,count]= fscanf(fid, '%f',M+I);
for j =1 :M
for i = j:I+j-1
 n=i-(j-1);
 Up(n,j)= X(i,1);
```

```
 end
 end          %Up is training input to the MLP
fclose(fid);
 fid = fopen('recgn.dat','r');
[X1,count]= fscanf(fid, '%f',M+I);
 for j =1:M
Yp(1,j)= X1(j+I-1,1);     %+1 for next sample
end
fclose(fid);
%-------------------------------------------------------------------------
% Reading one more data set for testing:Up1:
%-------------------------------------------------------------------------
fid = fopen('lemg.dat','r');
[X2,count]= fscanf(fid, '%f',tst+I);
for j =1 :tst
 for i = j:I+j-1
n=i-(j-1);
 Up1(n,j)= X2(i,1);
 end
 end
 fclose(fid);
 fid = fopen('lecgn.dat','r');
[X3,count]= fscanf(fid, '%f',tst+I);
for j =1:tst
Yp1(1,j)= X3(j+I-1,1);              %+1 for next sample
 end
fclose(fid);
%-------------------------------------------------------------------------
 eg=0.01;
sc=800;
net=newrb(Up,Yp,eg,sc);
%-------------------------------------------------------------------------
%Test the MLP WITH SAME INPUT PATTERNS Up:
%-------------------------------------------------------------------------
fid = fopen('recgn.dat','r');
[X1,count]= fscanf(fid, '%f',M+I);
 for j =1:M
Yp(1,j)= X1(j+I-1,1);              %+1 for next sample
```

```
end
fclose(fid);
for k = 1:M
 u = Up(:,k);
Yn(:,k)=sim(net,u);
 end
 fid = fopen('recg.dat','r');
[X4,count]= fscanf(fid, '%f',M+I);
 for j =1:M
 ecgip(1,j)= X4(j+I-1,1);   %for next sample
 end
 fclose(fid);
 k=0;
for i=1:M
for j=1:O
k=k+1;
x1(k)=Up(j,i);              %noise:EMG input
d11(k)=Yp(1,i);
dexp(k)=Yn(j,i);                    %Expectation of noise in ECG available at ANN output
 act_ecg(k)=ecgip(1,i);
 end
end
filterop=d11-dexp;
x11 = [0 0 0 0 0 filterop];
x22 = [0 0  0  0 filterop 0];
x33 = [0 0 0 filterop 0 0];
x44 = [0 0 filterop 0 0 0];
x55 = [0 filterop 0 0 0 0 ];
x66= [filterop 0 0 0 0 0 ];
y = (x11 + x22 + x33 +x44 +x55 + x66)/6;
y=y(1:M);
I=1:k;
%perr=(act_ecg-y)./act_ecg*100;
perr=(act_ecg-filterop)./act_ecg*100;
err=act_ecg-filterop;
figure(1)
title('Waveforms during training');
subplot(3,1,1);plot(I,x1);grid
```

```
ylabel('Noise');
subplot(3,1,2);plot(l,perr);grid
ylabel('% Error');
subplot(3,1,3);plot(l,err);grid
ylabel('Error');
figure(2)
title('Waveforms during training');
subplot(4,1,1);plot(Yp);
ylabel('noisy ECG');grid
subplot(4,1,2);plot(l,act_ecg);grid
ylabel('desired output');
subplot(4,1,3);plot(l,filterop);grid
ylabel('filter output');
subplot(4,1,4);plot(l,y);grid
ylabel('averaged');

%------------------------------------------------------------------
%Test the MLP WITH DIFFERENT INPUT PATTERNS Up1:
%------------------------------------------------------------------
fid = fopen('lecgn.dat','r');
[X3,count]= fscanf(fid, '%f',tst+I);
 for j =1:tst
 Yp1(1,j)= X3(j+I-1,1);  %for next sample
 end
fclose(fid);
for k = 1:tst
u = Up1(:,k);
Yn1(:,k)=sim(net,u);
 end
 fid = fopen('lecg.dat','r');
[X5,count]= fscanf(fid, '%f',tst+I);
 for j =1:tst
ecgip1(1,j)= X5(j+I-1,1);%for next sample
end
fclose(fid);
k=0;
for i=1:tst
for j=1:O
```

```
k=k+1;
x2(k)=Up1(j,i);  %noise:EMG input
d22(k)=Yp1(1,i);
 dexp1(k)=Yn1(j,i);        %Expectation of noise in ECG available at ANN output
act_ecg1(k)=ecgip1(1,i);
 end
end
filterop1=d22-dexp1;
x11 = [0 0 0 0 0 filterop1];%
x22 = [0 0  0  0 filterop1 0];
x33 = [0 0 0 filterop1 0 0];
x44 = [0 0 filterop1 0 0 0];
x55 = [0 filterop1 0 0 0 0 ];
x66=  [filterop1 0 0 0 0 0 ];
y1 = (x11 + x22 + x33 +x44 +x55 + x66)/6;
y1=y1(1:tst);
l=1:k;
%perr1=(act_ecg1-y1)./act_ecg1*100;
perr1=(act_ecg1-filterop1)./act_ecg1*100;
err1=act_ecg1-filterop1;
figure(3)
title('Waveforms during testing');
subplot(3,1,1);plot(l,x2);grid
ylabel('Noise');
subplot(3,1,2);plot(l,perr1);grid
ylabel('Percent Error');
subplot(3,1,3);plot(l,err1);grid
ylabel('Error');
figure(4)
title('Waveforms during testing');
subplot(4,1,1);plot(Yp1);
ylabel('noisy ECG');grid
subplot(4,1,2);plot(l,act_ecg1);grid
ylabel('desired output');
subplot(4,1,3);plot(l,filterop1);grid
ylabel('filter output');
subplot(4,1,4);plot(l,y1);grid
ylabel('averaged');
```

5.3.4.5 Results

Training is carried out with 100 number of input patterns (**Figure 5.17**). After training phase is over, the network is tested for 3000 number of samples and results are presented in **figure 5.18**

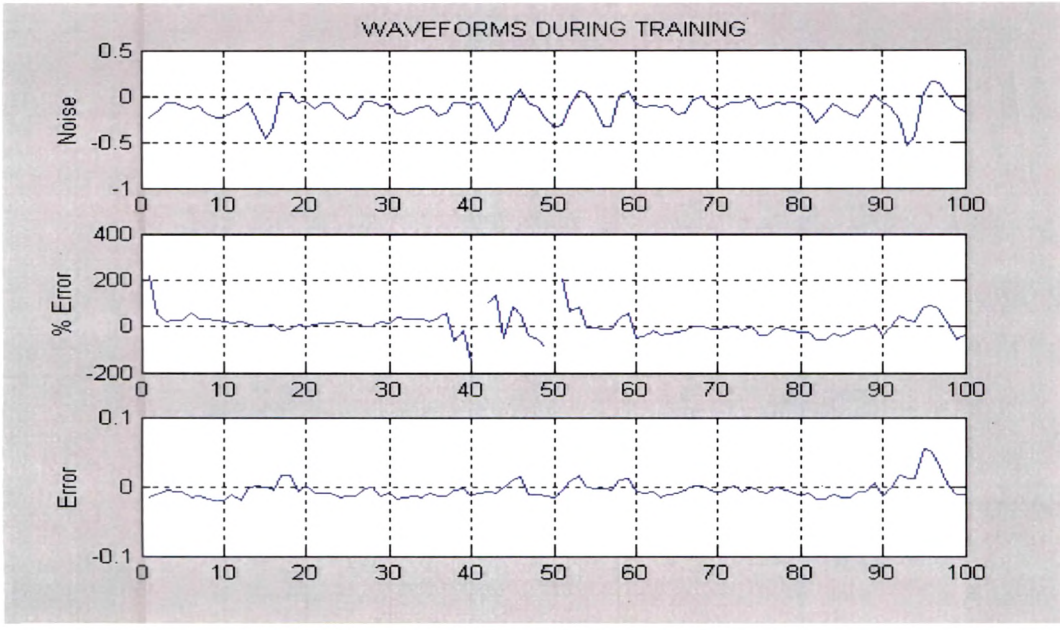

Figure 5.17 (a) Error response for 100 samples of training

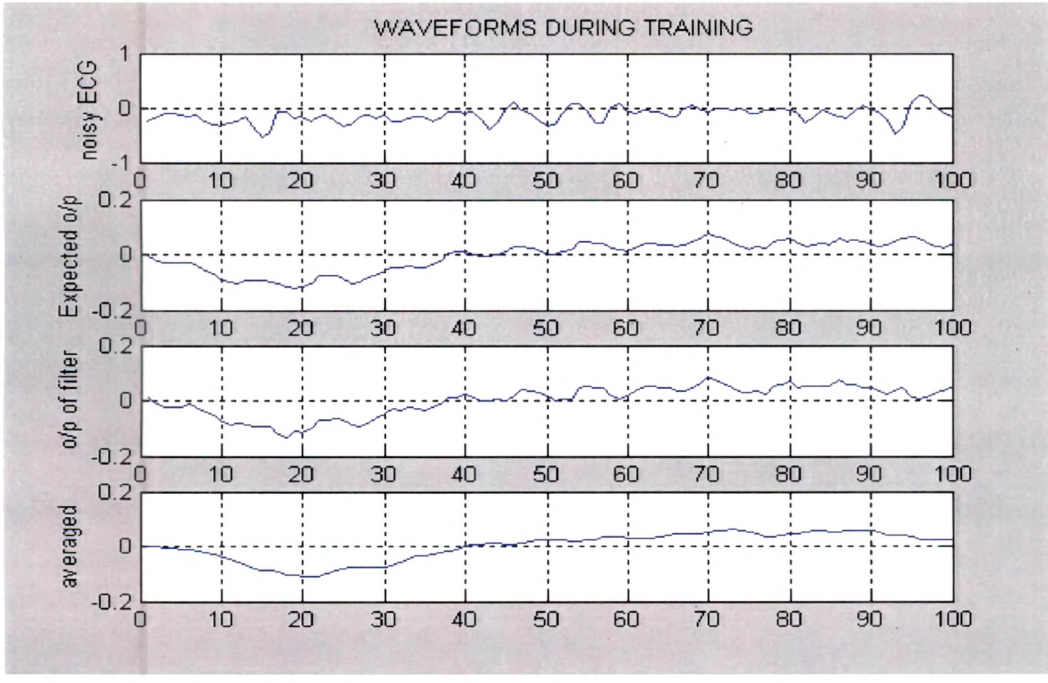

Figure 5.17 (b) Output of filter shown for 100 samples of training.

Figure 5.17 Training phase with 100 input samples (RBFNN)

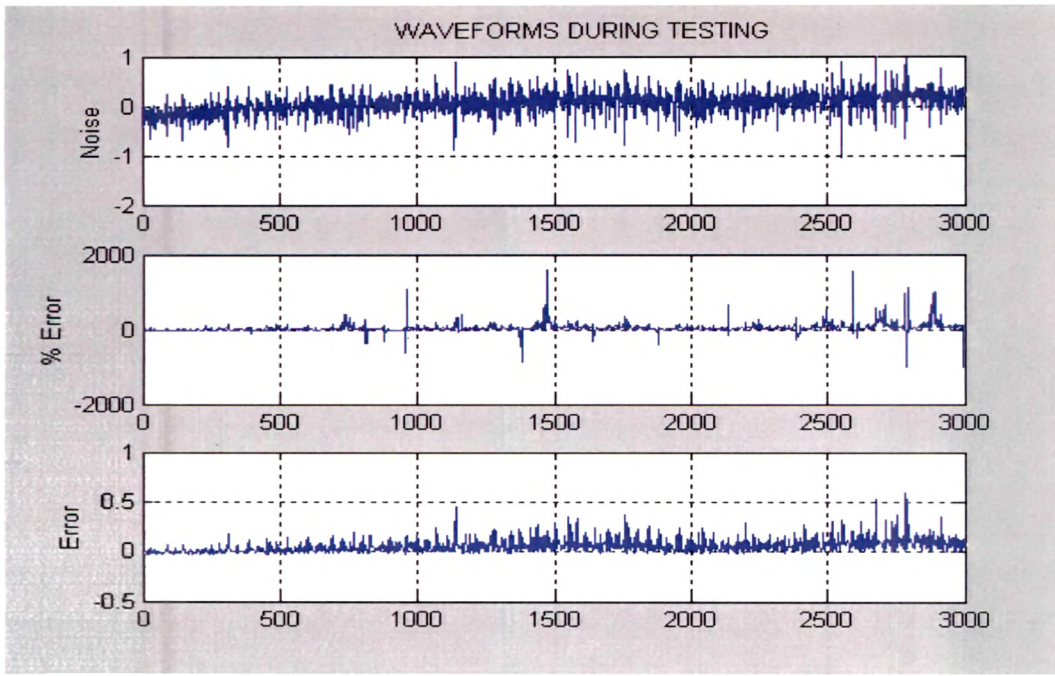

Figure 5.18 (a) Error response for 3000 samples of testing

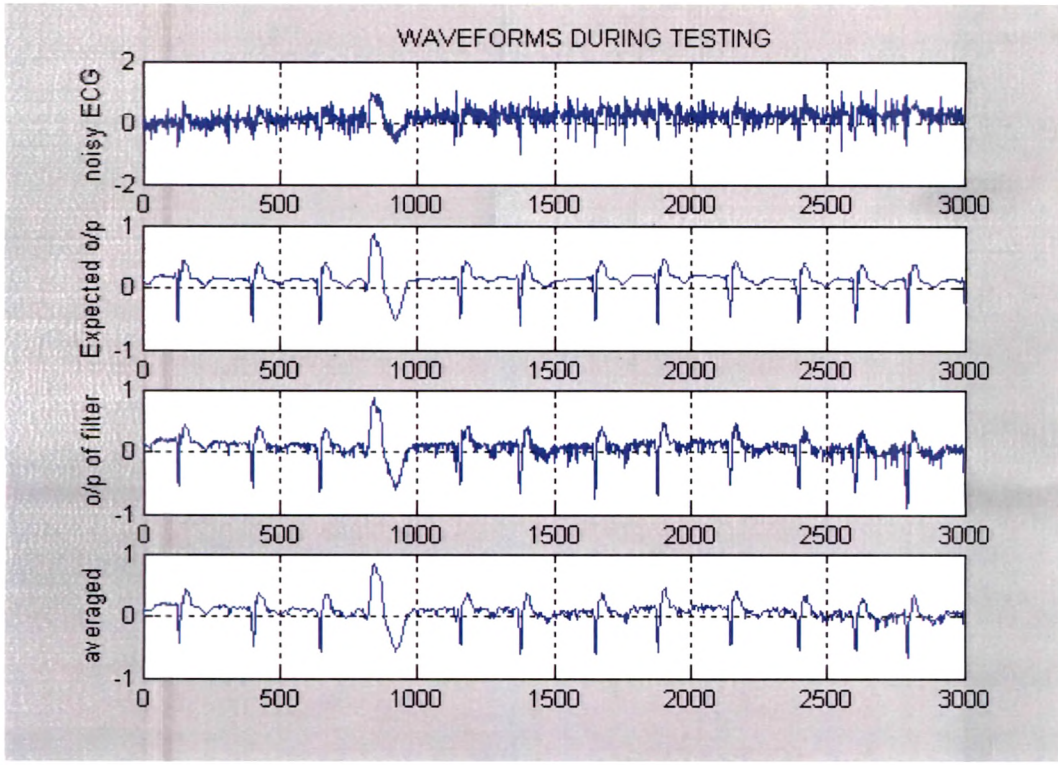

Figure 5.18(b) Output of filter shown for 3000 samples of testing.

Figure 5.18 Testing phase with 3000 input samples (RBFNN)

The comparison of response of ANN filtering models and training algorithm is carried out in Chapter 9.