Artificial neural networks for loudspeaker modelling and fault detection

Caroline Fox

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ....[illegible]............................(candidate)
Date ....30...1...96..................................

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ....[illegible]............................(candidate)
Date ....30...1..96..................................

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ....[illegible]............................(candidate)
Date ....30...1..96..................................

SUMMARY

This thesis is the result of a collaborative project between Cardiff University and Harman/Becker Automotive Systems. It investigates the application of Artificial Neural Networks to loudspeaker fault detection and modelling of the loudspeaker transfer function.

The aim was to utilise the ability of artificial neural networks to model high order nonlinear systems to generate a model of the loudspeaker transfer function which could be used in a linearisation scheme to reduce distortion in loudspeaker output. This thesis investigates a practical approach to transfer function modelling through the use of musical excitation signals. This would allow data to be collected during normal operation of the loudspeaker and, as the transfer function changes over time due to time dependent nonlinearities, would facilitate regular updating of the neural network model to incorporate these nonlinearities. It was determined that although very accurate models could be produced over long training periods, a significant compromise in ANN training set size and number of training epochs were required to reduce the ANN training duration to the required time period, which ultimately resulted in a decline in performance.

The aim in the case of fault detection was to improve on current end of production line testing for loudspeaker distortion. Neural networks were trained with harmonic distortion data in order to emulate the end of line test result. Excellent classification accuracy was achieved when neural network classification results were compared to the end of line test results.

An investigation was also conducted to determine if neural networks could be trained to recognise specific loudspeaker faults. In a development of the end of line test, a system of neural networks were trained to produce an output vector that described which of five frequency regions the loudspeaker distortion levels were above the limits, thus giving an indication of the possible fault.

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Prof. John Watton and Dr. Yiqin Xue, for their guidance during my Ph.D. I am also grateful to many people at Harman/Becker Automotive Systems, in particular Stel Anthony, for his patience while teaching a complete novice the theory of loudspeakers, and for his continued support throughout the project. Also, Toby Newman and Owain Bryant, who collected the data for the loudspeaker modelling.

I would like to thank my Mum and Dad for their unwavering support throughout my academic career; without them I would never have achieved what I have. I must also thank Mitch for his love and support, and for reminding me that the real world still existed.

The research was conducted with the financial support of the EPSRC and Harman/Becker Automotive Systems.

CONTENTS

1 INTRODUCTION

This thesis is the result of a collaborative project between Cardiff University and Harman/Becker Automotive Systems. Harman/Becker produce loudspeakers for a range of automotive manufacturers and were interested in the application of neural networks in their field. Consultation with Harman/Becker established that two appropriate areas to conduct research were the application of The first aim of this project was to determine if Artificial Neural Networks (ANNs) to loudspeaker modelling and fault detection.

Loudspeakers behave nonlinearly and extremely inefficiently. The current approach employed by many loudspeaker manufacturers is to improve loudspeaker performance through signal processing. The first aim of this project was therefore to determine if ANNs could produce a model of the loudspeaker transfer function of adequate performance, over very short training periods, with only loudspeaker input/output data generated from music signals to be employed in a linearisation scheme to eliminate nonlinear distortion in the loudspeaker output. The training duration was prerequisite so as to allow regular adaptation of the model to the time dependent parameters of the loudspeaker transfer function, as was the musical excitation signal, in order to facilitate online model training and therefore, the practical application of the system.

Various stationary models exist, including mathematical representations (Locanthi, (1952), Small, (1972b), Small, (1973), Kaizer, (1987), Gao, Snelgrove, (1991), Klippel, (1992), Frank, et al, (1992)), and ANN models (Low, Hawksford, (1993), Chang, et al, (1994)), with many used in preprocessing schemes to reduce nonlinear distortion. However, as discussed in section 2.4.2, there are several model parameters that are time dependent, the most significant being ageing effects and temperature effects, which a stationary model cannot incorporate. Therefore, during the course of this project the most efficient methods of incorporating time dependent parameters into an ANN model of the loudspeaker transfer function were investigated.

The second aim was to use ANNs for loudspeaker fault detection in order to improve the sophistication of the end of line test, which only provides an indication of whether the loudspeaker should be accepted or rejected. The specific loudspeaker fault is currently determined by expert listeners. A neural network that gave an indication of the specific fault immediately would reduce the time between the faulty loudspeaker being produced and the diagnosis of the fault, which may prevent the production of further loudspeakers with the same fault and would also reduce the need for expert listeners. Neural networks have been used in a wide range of applications, including fault detection systems (Kalayci and Özdamar, (1995), Foo et al, (2002)), however, they have not previously been applied to the problem of loudspeaker fault detection.

After the completion of loudspeaker fabrication it is tested to ensure satisfactory performance. The aim was to establish if an ANN could be trained to determine if a loudspeaker should be accepted and distributed to the customer or rejected and withdrawn from further distribution. The aim was also to investigate if the ANN could be trained to recognise specific faults. In a development of the end of line test, a system of neural networks were trained to produce an output vector that described which of five frequency regions the loudspeaker distortion levels were above the limits, thus giving an indication of the possible fault.

Chapter 2 discusses loudspeaker theory and the development of loudspeaker modelling.

Chapter 3 explains basic neural network theory, and includes a derivation of the algorithms used in the multi layer perceptron and modified Elman networks. It also briefly discusses several applications of neural networks.

Chapter 4 presents the strategy employed during the modelling of the loudspeaker transfer function. Issues such as the perception of distortion in a loudspeaker, the accumulation of ANN training data and preprocessing are discussed.

Chapter 5 contains details of the extensive investigation conducted to determine an optimum configuration for the two ANN structures considered for the modelling of the loudspeaker transfer function; the Multi Layer Perceptron feedforward network and the modified Elman recurrent network. Each of the network parameters, along with algorithm modifications, is considered in order to determine the configurations where correlation between model output and actual loudspeaker output is optimised. Frequency analysis is performed in order to determine the frequency response and harmonic distortion curves generated by the neural network models.

Chapter 6 discusses the various developments to the neural network algorithms used in the modelling of the loudspeaker transfer function with the aim of improving model performance or achieving the target model update rate of five minutes.

Chapter 7 presents the results of the investigation into the application of neural networks to loudspeaker fault detection and includes a description of the current end of production line test and common loudspeaker faults.

Chapter 8 summarises the thesis and discusses the conclusions reached during the project.

Chapter 9 presents the main conclusions of the research.

2 LOUDSPEAKER THEORY

2.1 LOUDSPEAKER DEVELOPMENT

The direct radiator loudspeaker was first proposed by Rice and Kellogg (1925) (they referred to it as 'hornless') and the vented box enclosure by Thuras, (1930). Previously the enclosures employed were either baffles or closed boxes, which prevented the sound wave produced by the back of the driver cone interfering with that produced at the front (the back wave is 180° out of phase with the front wave and would therefore cause destructive superposition). Closed box systems, also referred to as air suspension systems, consist of a loudspeaker with very high compliance mounted in an air tight box. The air pressure in the enclosure acts as a restoring force on the driver assembly. Closed box systems generally operate with relatively large driver excursions, which require the magnetic field to extend over a larger area or the voice coil to overhang the magnet, however, this reduces loudspeaker efficiency (Rossing, 1989). The vented box enclosure, also referred to a bass-reflex enclosure, incorporates a port that can take the form of a tunnel or duct extending into the interior of the enclosure or of a simple aperture, the principle being to allow air to move in and out of the enclosure in response to the pressure variations within the enclosure, thus the air acts as an inertial mass (Small, 1973b), under a similar principle as a Helmholtz vibrator. At frequencies below resonance, the air in the vent moves in phase with the back of the driver, hence the sound radiated will be out of phase with the sound radiated from the front of the driver, which at low frequencies results in destructive superposition, however in the frequency region adjacent to but still below the resonance frequency the sound radiated from the vent exceeds that from the driver and the total radiation is actually increased. In the region adjacent to but just above the resonance frequency, the air in the vent moves out of phase with the back of the driver, thus is in phase with the sound radiated from the front of the driver, hence positive superposition occurs and the total radiation is increased, however, at frequencies significantly above the resonance frequency the output from the vent is considerably reduced (Rossing, 1989). The resultant frequency response curve is shown in figure 2.1.

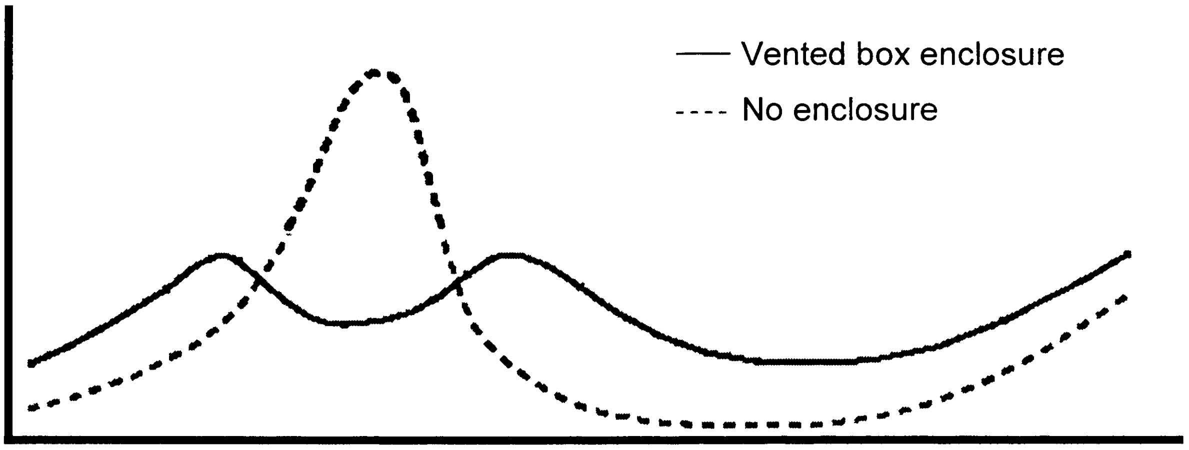


Figure 2.1 – Frequency response of a loudspeaker in a vented box enclosure tuned to the driver resonance frequency (Rossing 1989)

In addition to the increase in efficiency that results from the increased sound output of the loudspeaker and the extension of the loudspeaker response to lower frequencies, the vented enclosure also reduces the cone excursion at frequencies close to the resonance frequency which therefore reduces distortion and increases the power handling capability (Rossing, 1989). This is because the larger the cone excursion the greater the distortion, which is caused by a reduction in the magnetic field strength and the compliance of the cone suspension, and the maximum power output is limited by the maximum excursion possible without causing damage to the driver assembly or significant distortion in the output. The use of a duct rather than a simple aperture reduces the required enclosure volume to achieve a low resonance frequency (Rossing 1989). The duct can be designed so as to reverse the phase of the back wave, thus bringing it in phase with the front wave and hence the back wave may be used to reinforce the front wave over a larger range of frequencies, resulting in an increased power output of the loudspeaker (Caulton et al, 1936). The vent may also incorporate a passive radiator, or drone cone (a loudspeaker cone without a voice coil or magnet, that is driven by the back wave of the driver cone). This further reduces the required enclosure volume as the mass of the passive radiator contributes substantially to the impedance of the vent and also reduces the air velocity at high power levels, which results in significantly less turbulence than a vent which requires larger air velocities (Rossing, 1989).

2.2 ANALOGOUS CIRCUIT DEVELOPMENT

Roder (1936) presented a basic analogous electrical circuit for a loudspeaker driver (assuming an infinite baffle enclosure), modelled as a mass-spring-damper system (inductance-capacitance-resistance in the electrical impedance type analogy, see section 2.3) and with transducers linking the electrical, mechanical and acoustical domains.

Caulton et al (1936) developed a vented enclosure that employed 5 tubes proportioned so as to reverse the phase of the back wave and included in their discussion electrical analogous circuit diagrams. The closed box enclosure was modelled, again with the impedance type analogy, with the addition of a capacitance in series with the capacitance-inductance-resistance representation of the loudspeaker driver, and the vented box enclosure with further parallel inductance and resistance terms. The transformation between domains was represented as the electrical impedance reflected into the electrical domain by the transducer.

Locanthi (1952) developed an electrical circuit that incorporated the external coupling of the cone and the vent. The inclusion of this mutual impedance resulted in an improvement in the agreement between measured impedance of the loudspeaker unit and that derived from the electrical circuit analogy. The circuits could therefore be used for the performance analysis of vented box loudspeaker systems.

In his book 'Acoustics', which presents a complete account of loudspeaker concepts such as acoustics and analogous circuits, Beranek (1954) derived polynomial expressions for the response of loudspeakers in closed and vented boxes. The expression for the vented box enclosure was a significantly simplified version of the equation originally proposed by the inventor of the vented box concept, Thuras, (1930), as Beranek assumed operation in the very low frequency region where the radiation from the vent and the driver is nondirectional (Beranek, 1954), hence the interaction between the cone and the vent was negligible (Small, 1973b). Berenek's expression took into account system losses and the variation of radiation load resistance with frequency.

The use of electrical analogous circuits in the design process was first attempted by van Leeuwen in 1956, however, the paper was published in Dutch therefore did not receive significant attention. Van Leeuwen examined the diaphragm-vent interaction neglected by Beranek and derived polynomial expressions for the frequency response. He determined accurate methods of calculating the driver and system parameters, including their nonlinearities, from the voice coil impedance measurement. He proposed a system design process and illustrated how the analogous electrical circuit could be used to determine the voice coil impedance and the steady-state and transient response of the system (Small, 1973b).

In 1959, de Boer made the connection between the behaviour of a loudspeaker in a vented box enclosure and that of a high-pass filter. This was critical for the analysis of loudspeaker behaviour and for loudspeaker design as well established filter theory could be applied (Small, 1973b).

Novak (1959) further simplified Beranek's transfer function by neglecting the vent radiation resistance as well as the interaction between the diaphragm and the vent. This was justified as the diaphragm radiation resistance could be 20 to 40 times greater than that from the vent. He discounted previous opinion that a large air stiffness in a small closed box increased driver damping; he established that damping was actually a function of the resistances in the system, and the factor exerting the greatest influence on the damping was the product of the magnetic field strength and voice coil conductor length. He determined the optimum range of driver damping for a flat response with a vented enclosure and presented methods for determining the driver and system parameters from voice coil impedance measurements.

He achieved good correlation between calculated and measured loudspeaker response, despite the exclusion of mutual coupling effects.

Thiele, (1961), is widely regarded as the first comprehensive quantitative treatment of loudspeakers in vented box enclosures. He presented a further simplification of the transfer function; as loudspeaker efficiency is extremely low (between 0.4 and 4 per cent) both the diaphragm and vent radiation resistances could be neglected, despite being the acoustic output of the system. Radiation resistance varies with the square of frequency, hence the transfer function was significantly simplified. He also neglected the acoustic resistance of the enclosure and of the air in the vent and lumped together firstly the acoustic mass of the diaphragm and voice coil and the acoustic mass of the air load on the front and the rear of the diaphragm, and secondly the acoustic mass of the air load on the vent and the acoustic mass of air in the vent. He substituted the expression for the transfer function into 4th order high pass filter functions in order to determine design parameters for various response curve shapes. The system parameters for any desired response characteristics could then be determined using filter theory. Previously, only empirical design methods were possible. However, Thiele developed a precise, quantitative method for loudspeaker design based on the knowledge of four measurable loudspeaker parameters:

- resonance frequency of moving system of driver specified either for the driver in air or a specific baffle
- acoustic compliance of driver, expressed as an equivalent volume of air
- Q value due to electrical resistance
- Q value due to voice coil dc resistance

Benson, (1968), presented a detailed derivation of a generalised loudspeaker system transfer function, which he showed could be manipulated to produce the transfer functions of specific types of enclosure through the suitable choice of parameters.

Thiele's 1961 paper did not receive significant acclamation until a decade later when Thiele's PhD student published the series of papers (Small, 1972a, Small, 1972b, Small, 1973a, Small, 1973b, Small, 1973c, Small, 1973d) which generalised Thiele's derivation of the transfer function in a similar way to Benson (1968), to include any type of enclosure, and also presented analysis and synthesis methods for each enclosure type. He extended Thiele's work to include treatment of efficiency-response relationships and large signal behaviour, evaluation of diaphragm-vent interaction, and the assessment of the magnitude and effects of normal enclosure losses.

In Small, (1972a), the observation of de Boer, Beranek and others that the acoustic power radiated by the system is directly related to the volume velocity compressing and expanding air within bass-reflex (vented) enclosures was expanded to include all direct-radiator system enclosures. The equations for efficiency were clearly stated and electrical analogous circuits were used to derive a general transfer function which incorporates terms relating to the type of enclosure and whether a passive radiator is employed. Small identified five fundamental driver parameters which control system small-signal performance:

- dc resistance of the voice coil
- effective projective surface area of the driver diaphragm
- mechanical compliance of the driver suspension
- mechanical mass of the driver diaphragm assembly including voice coil and air load
- mechanical resistance of driver suspension losses.

Each parameter may be set independently and has some effect on the system small-signal performance. These parameters were not straightforward to measure directly, and Small referred to the four basic parameters determined by Thiele which related to the parameters he identified but were easier to measure and manipulate.

Small also discussed large signal parameters in terms of a displacement limit above which distortion becomes unacceptable or damage may be caused to the loudspeaker suspension, assuming linear diaphragm displacement.

In the subsequent papers Small applies the concepts derived in Small (1972a) to loudspeakers in closed box (Small, 1972b, Small, 1973a) and vented box (Small, 1973b, Small, 1973c, Small, 1973d, Small, 1973e) enclosures. He shows the correlation between the closed box loudspeaker system response function and a second order high-pass filter and between the vented box loudspeaker system response function and a fourth order high pass filter. He determined that closed box loudspeaker system efficiency is dependent upon frequency response and enclosure size and that acoustic power capacity is determined from frequency response and the volume of air displaced by the driver diaphragm. For vented box loudspeaker systems efficiency may be determined through knowledge of these parameters plus internal losses.

At the time of the publication of Small's papers there was a trend towards employing closed box enclosures due to their straightforwardness of design,. however, Small suggested that vented box enclosures provided superior efficiency characteristics and power capacity than comparable closed box designs (Small, 1973c), and his simple design methods removed the disadvantage of complex design procedures.

2.3 ELECTRO-MECHANO-ACOUSTICAL CIRCUIT ANALOGY OF THE LOUDSPEAKER

Mechanical and acoustical systems may be analysed through the use of an analogous electrical circuit. This enables the application of well established electrical circuit theory to systems that may contain a combination of electrical, mechanical and acoustical elements, such as the loudspeaker.

There are two commonly applied analogies – the mobility type and impedance type. The mobility type analogy represents quantities that act through elements as analogous, likewise quantities that act across elements, and the impedance type analogy adopts the opposite convention. Beranek (1954) ch.3 discusses the development of electro-mechano-acoustical circuit analogies in detail.

2.3.1 Assumptions

The transfer function derived here is that of a generalised loudspeaker system consisting of a driver unit (the electro-mechano-acoustical transducer), and an enclosure that incorporates apertures for the driver unit, a passive radiator or vent, and also leakage that may occur through the enclosure structure. The system is illustrated schematically in figure 2.2.

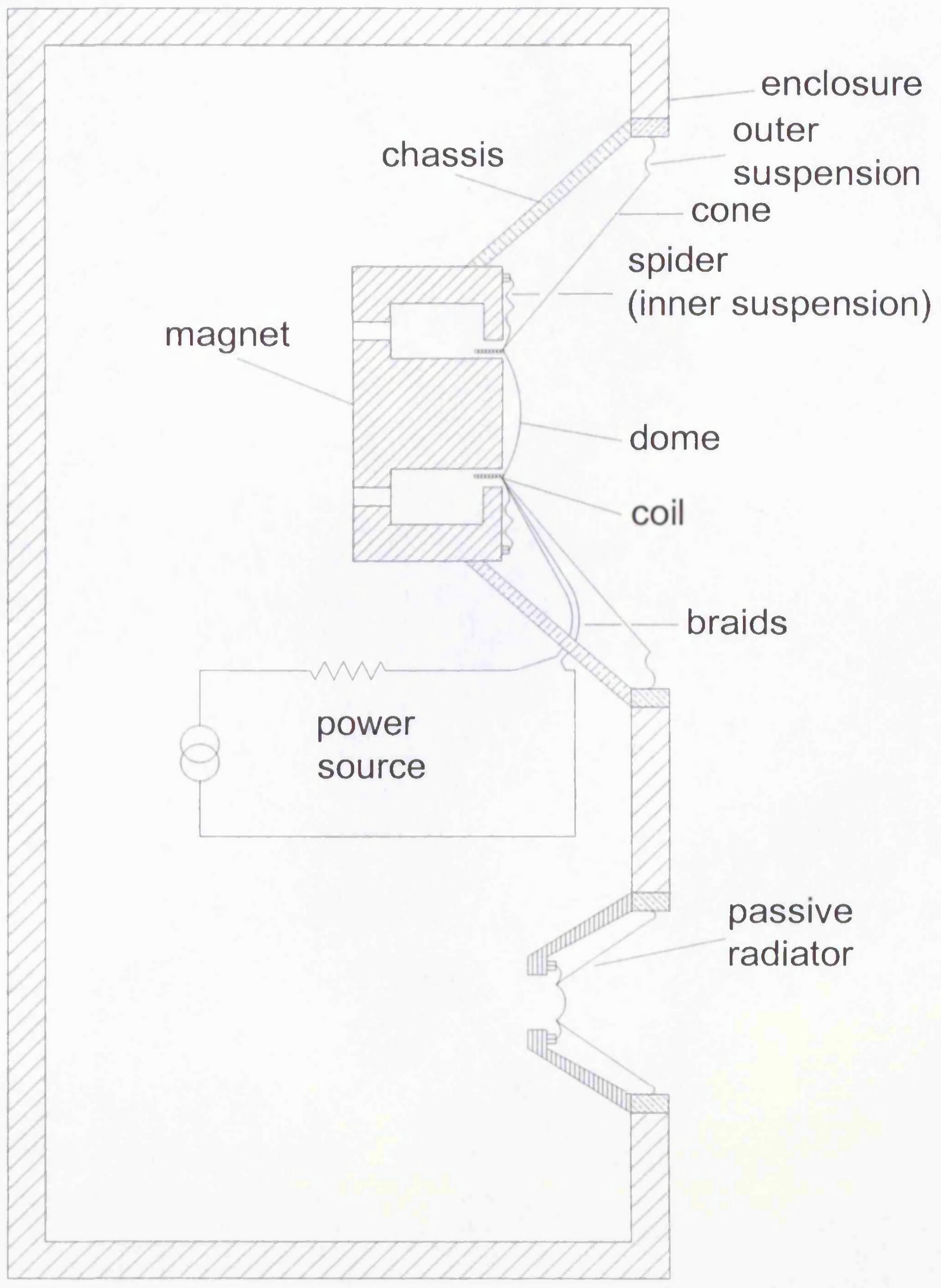


Figure 2.2 – Schematic of a loudspeaker in a vented box with a passive radiator

The following assumptions are made:

- The elements of the loudspeaker system may be considered as lumped i.e. the only independent variable of the system is time.
- The system operates within the piston range of the system driver, where all parts of the diaphragm vibrate in phase.
- The voice coil inductance is negligible.
- The acoustic radiation resistance of the diaphragm and the vent are neglected despite being the acoustic output of the system. This is due to the substantial inefficiency of the loudspeaker system.
- The mutual impedance of the driver diaphragm and passive radiator diaphragm or vent is negligible.
- For the main driver and the passive radiator, the air load on the front and back of the diaphragm are lumped together with the combined mass of the diaphragm and voice coil.

2.3.2 Electrical elements

The electrical elements of a loudspeaker system include the source voltage, e_g, and resistance, R_g, and the voice coil resistance, R_E. The circuit representation of the electrical parameters is shown in figure 2.3.

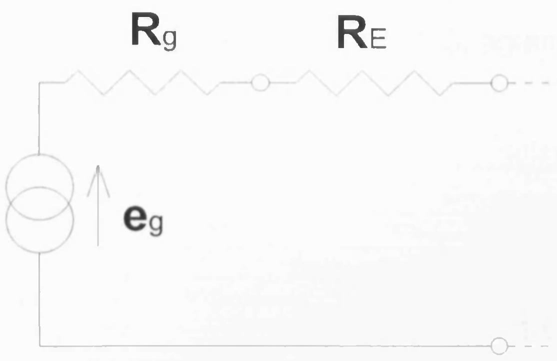


Figure 2.3 – Electrical elements of loudspeaker system

2.3.3 Mechanical elements

The mechanical part of the equivalent circuit consists of elements from the voice coil, the driver diaphragm and suspension and the passive radiator diaphragm and suspension. M_{MS} represents the combined mass of the driver diaphragm and wire on the voice coil, C_{MS} and r_{MS} correspond respectively to the compliance and mechanical responsiveness of the combined centre and edge suspensions of the driver. M_{MP}, C_{Mp} and r_{Mp} are the mass of the passive radiator and the compliance and mechanical responsiveness of the combined centre and edge suspensions of the passive radiator respectively. The equivalent electrical circuit, using the mobility type analogy, is shown in figure 2.4.

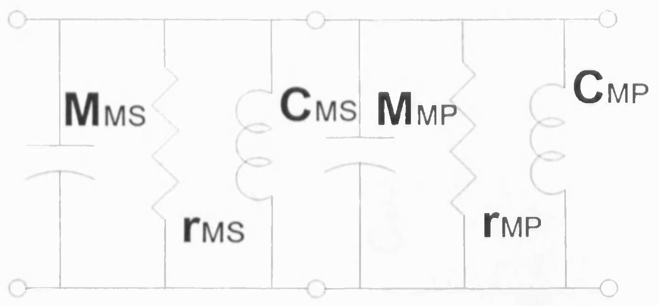


Figure 2.4 – Mechanical elements of loudspeaker system

2.3.4 Acoustical elements

The acoustical elements of the loudspeaker system include the compliance of the air in the enclosure, C_{AB}, the acoustic responsiveness due to internal energy absorption within the enclosure, r_{AB} and that due to losses caused by leakage, r_{AL}, The equivalent electrical circuit of the acoustical elements of the system are illustrated in figure 2.5, again using the mobility type analogy.

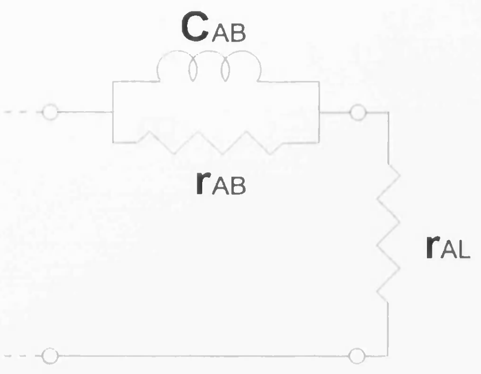


Figure 2.5 – Acoustical elements of loudspeaker system

2.3.5 Complete analogous circuit

These circuits can be combined to create a complete representation of the loudspeaker system as illustrated in figure 2.6.

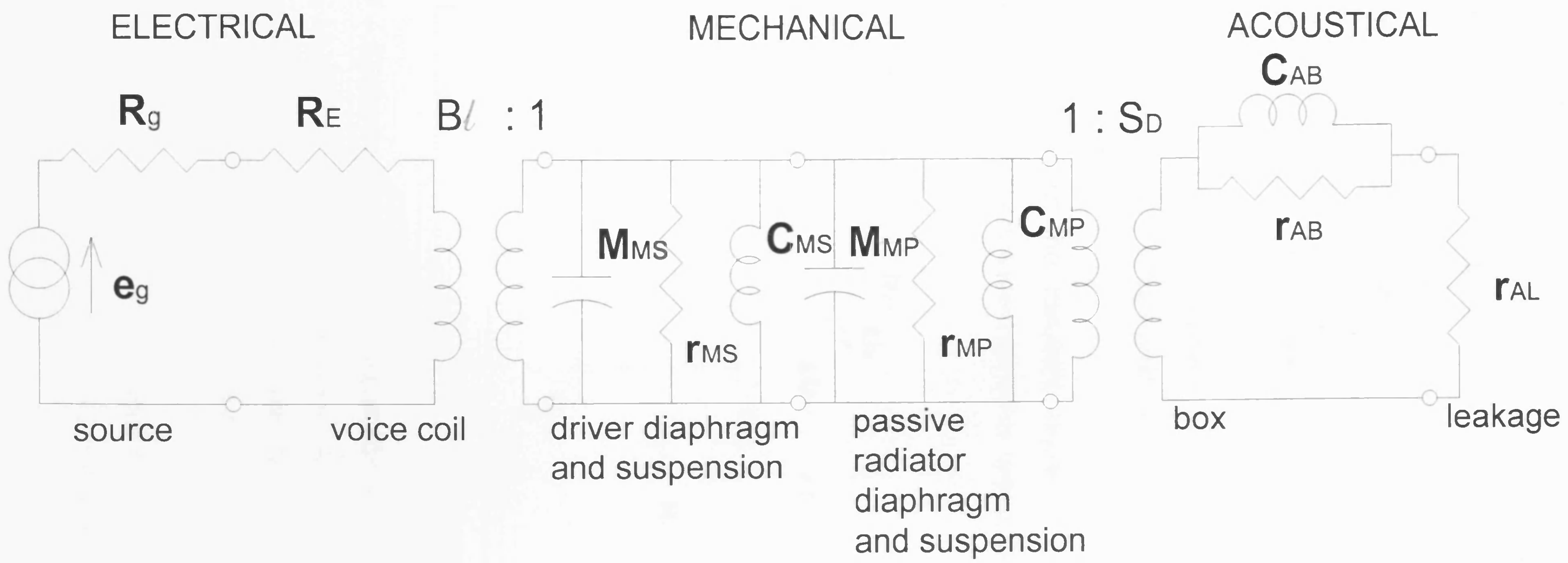


Figure 2.6 – Analogous circuit representation of a vented box loudspeaker system with passive radiator

The acoustical equivalent circuit as described by Thiele (1961) and Small (1972a) can be obtained through the following modifications to the electro-mechano-acoustical circuit of figure 2.6:

1. Refer the electrical elements to the mechanical side of the circuit (i.e. remove electro-mechanical transducer).
2. Refer the mechanical and equivalent electrical elements to the acoustical side of the circuit (i.e. remove mechano-acoustical transducer).
3. Determine the Norton equivalent circuit of the equivalent electrical elements (i.e. convert circuit from constant voltage source to constant current source).
4. Determine dual of the resultant circuit to obtain impedance type analogy acoustical equivalent circuit as shown in figure 2.7.

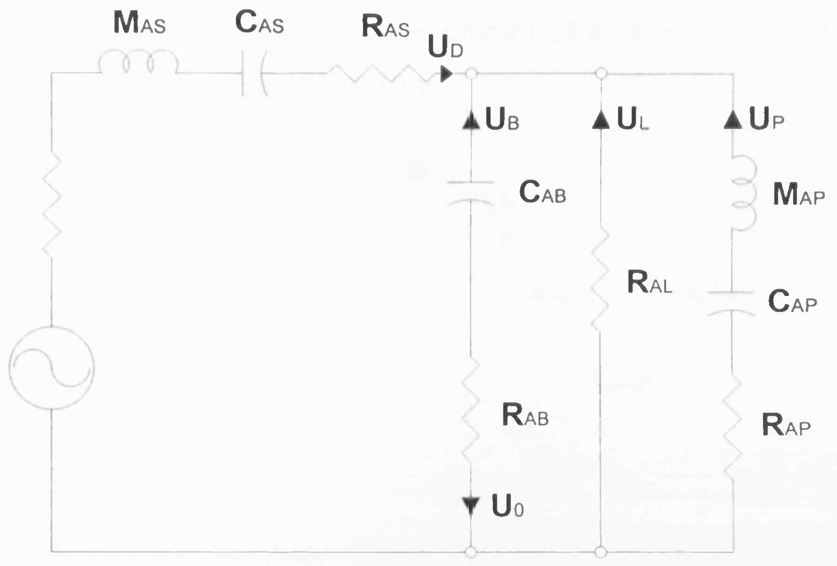


Figure 2.7 – Acoustical analogous circuit (Small, 1972a)

Where U_D, U_P and U_L are the volume velocities of air movement at the driver diaphragm, port and leak respectively, and U_B and U_0 are the total volume velocities entering and leaving the enclosure.

The acoustical analogous circuits of various direct radiator systems can be obtained by removing or short-circuiting appropriate elements in figure 2.7.

Figure 2.8 illustrates a closed box loudspeaker system and figure 2.9 a vented box loudspeaker system.

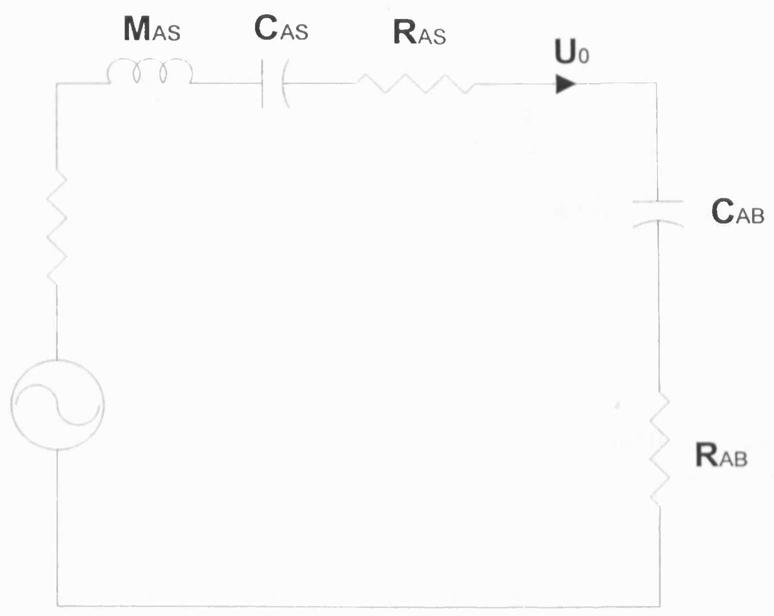


Figure 2.8 – Acoustical analogous circuit of a closed box loudspeaker System (Small, 1972b)

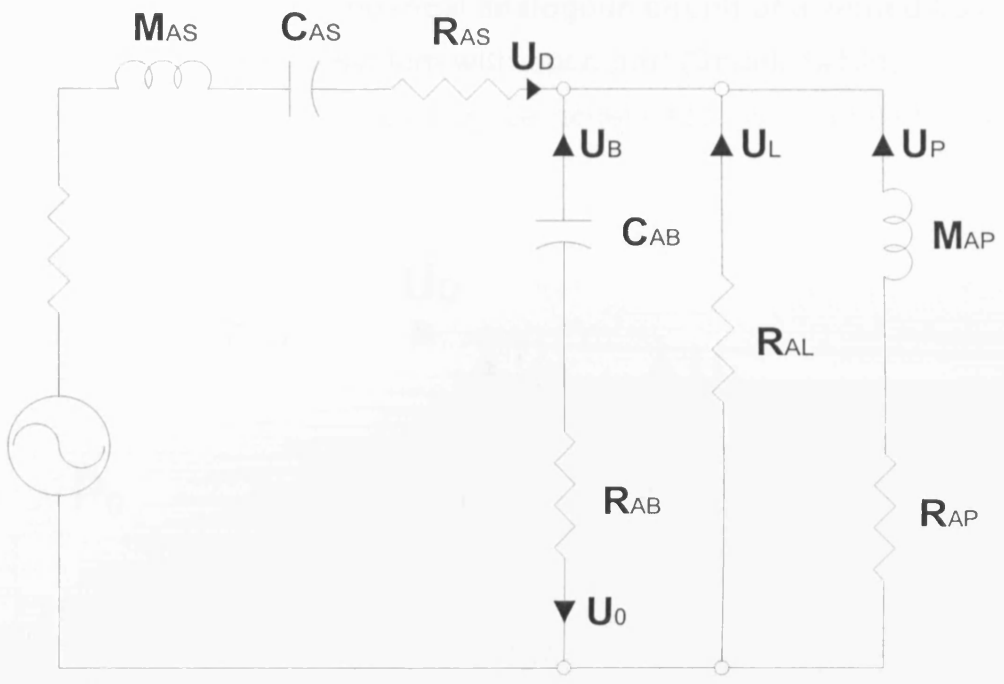


Figure 2.9 – Acoustical analogous circuit of a vented box loudspeaker system with open port (Small, 1973b)

The circuit of figure 2.9 can be further simplified by assuming R_{AL} is the significant loss when compared to R_{AB} and R_{AP}, this is illustrated in figure 2.10.

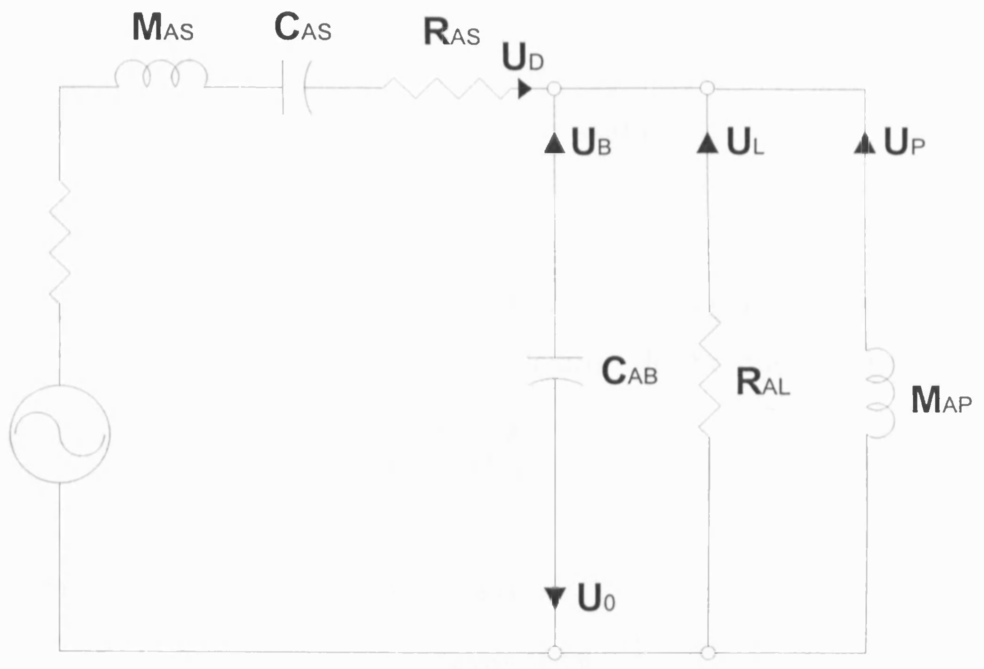


Figure 2.10 – Simplified acoustical analogous circuit of a vented box loudspeaker system with open port (Small, 1973b)

The acoustical analogous circuit may be generalised as illustrated in figure 2.11.

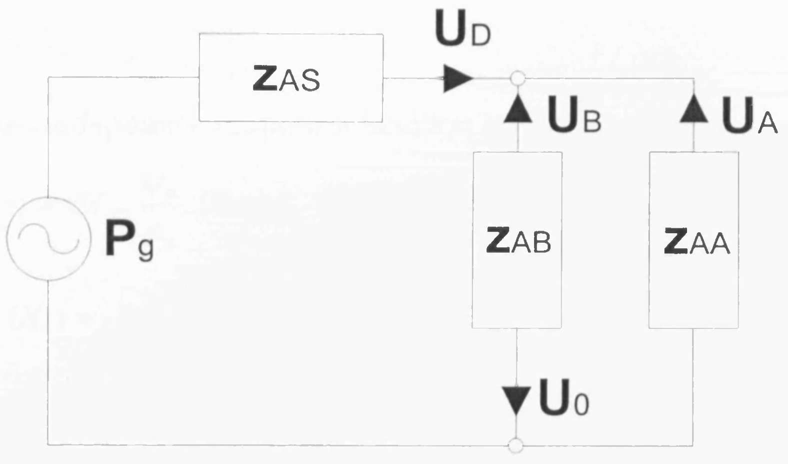


Figure 2.11 – Generalised acoustical analogous circuit (Small, 1972a)

Where p_g is the acoustic driving pressure, $p_g = \dfrac{e_g Bl}{(R_g + R_E)S_D}$ (2.1)

$Z_{AS}(s)$ is the impedance of the driver branch,

$$Z_{AS}(s) = R_{AS} + \frac{B^2 l^2}{(R_g + R_E){S_D}^2} + sM_{AS} + \frac{1}{sC_{AS}} \tag{2.2}$$

$Z_{AB}(s)$ is the impedance of the branch representing the enclosure interior,

$$Z_{AB}(s) = R_{AB} + \frac{1}{sC_{AB}} \tag{2.3}$$

Z_{AA} is the impedance of any enclosure apertures excluding that of the driver and is dependent upon the type of enclosure. In the case of the vented box enclosure of figure 2.9, $Z_{AA} = \dfrac{R_{AL} sM_{AP}}{R_{AL} + sM_{AP}}$ (2.4)

2.3.6 Derivation of transfer function

Applying Kirchoff's second law to figure 2.11:

$$p_g - U_D Z_{AS} = U_0 Z_{AB} = U_D Z_{AT} \tag{2.5}$$

where $Z_{AT} = \dfrac{Z_{AA} Z_{AB}}{Z_{AA} + Z_{AB}}$ (2.6)

Therefore $U_0 = U_D \dfrac{Z_{AT}}{Z_{AB}}$ (2.7)

and $U_D = \dfrac{p_g}{Z_{AS} + Z_{AT}}$ (2.8)

The loudspeaker response function is:

$G(s) = sM_{AS} \dfrac{U_0}{p_g}$ (Small, 1972a) (2.9)

$$\Rightarrow G(s) = \frac{sM_{AS} Z_{AT}}{Z_{AB}(Z_{AS} + Z_{AT})} \tag{2.10}$$

$$\Rightarrow G(s) = \frac{sM_{AS}}{Z_{AB} + Z_{AS} + {Z_{AB} Z_{AS}}/{Z_{AA}}} \tag{2.11}$$

Substitution of parameters results in second and fourth order transfer functions for closed box and vented box enclosures respectively.

2.4 NONLINEARITIES IN LOUDSPEAKER TRANSFER FUNCTIONS

Nonlinear distortion involves the introduction of frequency components, harmonic and inharmonic, inharmonic being the more discernable, that were not present in the input signal. The various mechanisms by which this form of distortion may be introduced are discussed in sections 2.4.1 and 2.4.2.

2.4.1 Constant nonlinear parameters

A principal nonlinearity is that caused by the displacement dependent force factor (Greiner and Sims, 1983, Kaizer, 1987, Klippel, 1992a, Birt, 1991). The magnetic flux acting on the voice coil is significantly reduced when the coil is at an extreme of its excursion (Villchur, 1956). This is illustrated in figure 2.12.

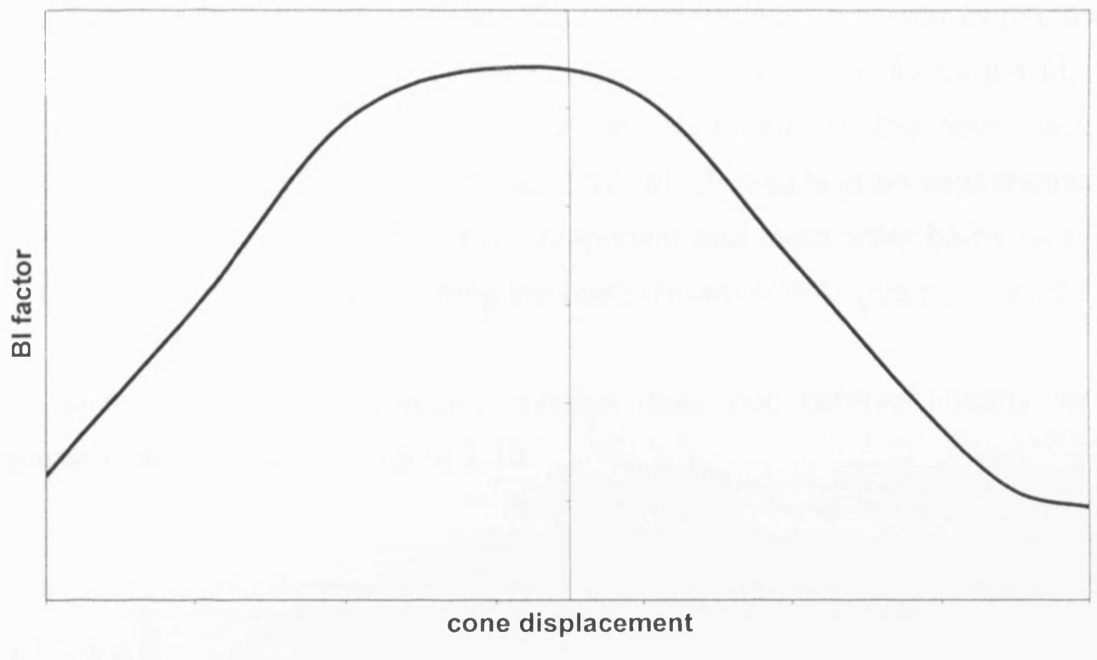


Figure 2.12 – Plot of Bl factor variation with cone displacement (Low, Hawksford, 1993)

Inside the gap there is a position of maximum magnetic flux, to either side of which the flux strength decreases, this therefore results in a continuously decreasing electrodynamic driving force (force factor) as the voice coil traverses towards its maximum excursion. The consequences are a parametric excitation and a nonlinear damping of the mechanical system

(Klippel, 1992a), the observable effect of which is a relatively smaller increase of the fundamental's excursion amplitude at lower frequencies, which require an excursion further away from the position of maximum magnetic flux (cone excursion is inversely proportional to ω^2 for constant power output (Chernof, 1957)), than that at higher frequencies. This is commonly referred to as compression. If the voice coil reaches an excursion at which the magnetic field is no longer strong enough to influence it, the voice coil may enter free vibration, which results in a phase shift between the input signal and voice coil displacement when the voice coil re-enters the influence of the magnetic field. This combination of phase shifting and displacement dependent force factor results in an excursion limit, where an increase in the electric input signal is offset by a proportional decrease in the force factor. (Klippel, 1992a). Mechanical damping is caused by the interaction between the electrical resistance of the source and the voice coil and the displacement dependent force factor in the case of a voltage driven loudspeaker, however in practice this effect is masked by the nonlinear damping caused directly by the force factor (Klippel, 1992a). There is also an asymmetry in the force factor characteristic, as can be seen in figure 2.12, which results in an asymmetrical voice coil excursion and hence a dc component and even order harmonics in the output, the dc component being the predominant effect. (Klippel, 1992a).

The stiffness of the suspension system does not behave linearly with excursion, as illustrated in figure 2.13.

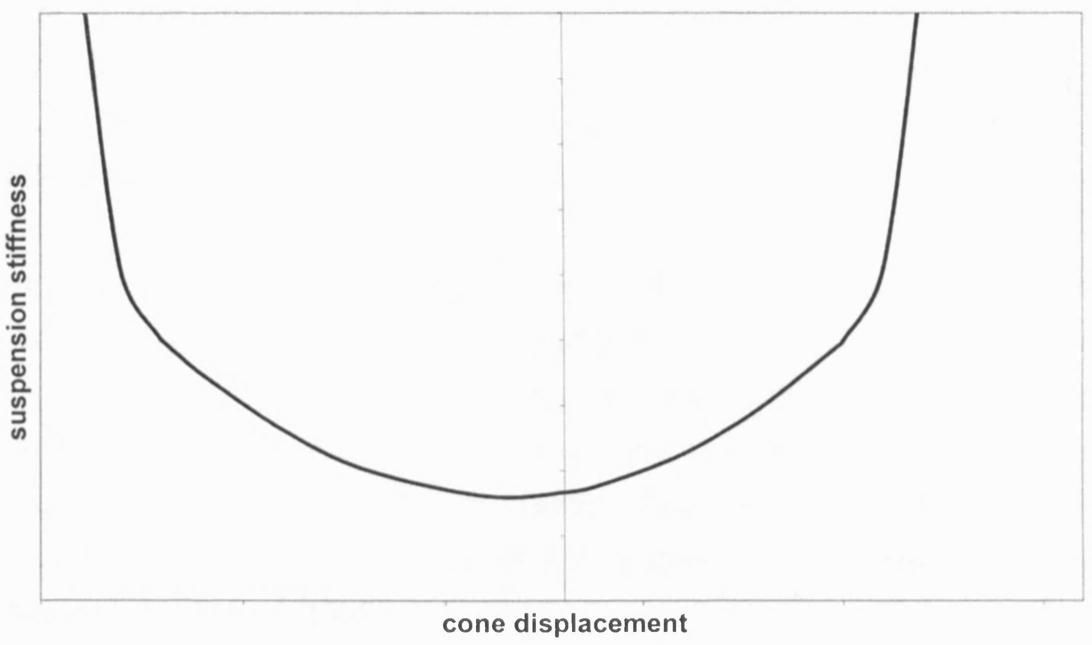


Figure 2.13 – Plot of suspension stiffness variation with cone displacement (Low, Hawksford, 1993, Klippel, 1992a)

A nonlinear suspension system stiffness characteristic results in a virtual alteration of the resonance frequency of the loudspeaker, as it is dependent upon the point at which the force effects of mass and spring cancel, and therefore is dependent upon driver amplitude (Klippel, 1992a). This resonance shifting also results in a jump phenomenon in the response characteristic at extreme amplitudes (Olson, 1944, Klippel, 1992a). The resonance curve is skewed to the point that its elements overlap, resulting in two possible stable vibration states, which the loudspeaker will jump between when the excitation frequency reaches the point of coincidence. This is illustrated in figure 2.14 (Klippel, 1992a). The frequency at which the amplitude jump occurs is dependent upon whether frequency is increasing or decreasing.

There is also nonlinearity in the mechanical damping caused by the compression and expansion of air in the loudspeaker enclosure (Olson, 1960), although this is often concealed by the nonlinear damping caused by the force factor (Klippel, 1992a).

Distortion may also be caused by a voice coil that is not centred, front to back under zero-signal conditions (Greiner and Sims, 1983), or symmetrically in the air gap, which may result in rubbing (Villchur, 1956). Also, voice coil inductance may vary nonlinearly, however this is more significant at higher frequencies (Mills and Hawksford, 1989). Also more significant at higher frequencies are eddy current losses and hysteresis in the magnetic circuit (Mills and Hawksford, 1989).

The problem of nonlinear distortion is further compounded as the distortion components occur simultaneously and interact with the fundamental and with each other to cause further nonlinear effects (Klippel, 1992a).

2.4.2 Time dependant nonlinear parameters

The transfer function of a loudspeaker is also time dependent. Voice coil resistance can produce significant temperature increases in the voice coil assembly, which results in an augmentation of the coil resistance and consequently a loss of sensitivity, reduction in damping and cross-over misalignment (Mills, Hawksford. 1989). As temperature increases, the suspension may also become softer and eddy currents due to temperature gradients may also form, resulting in an altered response. In the longer term, processes such as leaching of plasticisers, work hardening and fatigue (Birt, 1991) will alter the performance of the suspension, hence the transfer function will also vary over longer periods of time.

2.5 MODELLING OF NONLINEAR RESPONSE

The models discussed thus far are limited to the frequency region in the piston range of the loudspeaker (above which the cone ceases to vibrate as a stiff piston), where the loudspeaker response approaches that of a linear system and is independent of frequency and thus the models make no consideration of nonlinearities.

Olson, (1944), discussed the application of differential equations with variable coefficients to the modelling of a loudspeaker with a nonlinear cone suspension. He successfully modelled the phenomenon where a jump in the frequency response is experienced (see section 2.4.1) and the production of harmonics.

Olson modelled the suspension force factor as $F_m = f(x) = \alpha x + \beta x^3$ (2.12)

The compliance of the suspension system therefore becomes:

$$C_M = \frac{x}{f_m} = \frac{1}{\alpha + \beta x^2} \tag{2.13}$$

Substituting this into the equation of motion:

$$m\ddot{x} + r_M \dot{x} + \frac{x}{C_M} = F\cos\omega t$$
$$\Rightarrow m\ddot{x} + r_M \dot{x} + \alpha x + \beta x^3 = F\cos\omega t \tag{2.14}$$

Neglecting the mechanical resistance:

$$\Rightarrow m\ddot{x} + \alpha x + \beta x^3 = F\cos\omega t \tag{2.15}$$

If β is assumed to be small:

$$\omega^2 = \frac{\alpha}{m} + \frac{\frac{3}{4}\beta A^2}{m} - \frac{F}{Am} \tag{2.16}$$

An approximate solution for unit mass is therefore:

$$x = A\cos\omega t + \frac{\beta A^3}{32(\alpha + \frac{3}{4}\beta A^2 - \frac{F}{A})}\cos 3\omega t \tag{2.17}$$

This determined that the nonlinear suspension results in a third order harmonic, a result well substantiated by experimentation. The frequency – amplitude characteristic obtained from this analysis is shown in figure 2.14. It can be seen that at frequencies adjacent to resonance there are two theoretically possible amplitudes, and it is this that results in the jumping phenomenon.

Chernof, (1957), asserted that the most significant nonlinearities are the nonlinear suspension characteristics and the non-uniform distribution of magnetic flux in the air gap, both of which are most apparent at low frequencies. He modelled the force characteristic in the same way as Olson, and suggested a solution to the problem of non-uniform flux would be to design the axial gap length to considerably exceed the length of the voice coil. He also suggested several methods to improve low frequency performance (extending the region of linear frequency response to lower frequencies by lowering the resonance frequency and damping the peak in acoustic output at resonance), such as decreasing suspension stiffness, increasing the mass of the diaphragm or minimising the motional resistance of the voice coil by increasing the magnetic field strength. He also referred to negative voltage feedback combined with positive current feedback through the amplifier to lower the output impedance and compensate for the voice coil resistance, thus suppressing the effects of resonance. He proposed that motional feedback may be employed to counteract the nonlinearities of the cone suspension and magnetic field, and described methods of obtaining the loudspeaker output voltage by placing an additional winding over the voice coil or by inserting the voice coil as one arm of a bridge circuit, balanced with an equal inductance and resistance, thus enabling the back emf to be separated from the driving voltage.

Kaizer, (1987), discussed the modelling of low frequency nonlinear distortion of the transfer function of a loudspeaker in a closed box enclosure with various mathematical methods. He took the governing differential equations (2.18) and (2.19) and approximated the nonlinear force factor, suspension stiffness and voice coil inductance (neglecting frequency dependence) with a truncated power series (2.20-2.22).

$$Bli = m\ddot{x} + R_M\dot{x} + kx \tag{2.18}$$

$$e_g = R_E i + \frac{d(L_e i)}{dt} + Bl\dot{x} \tag{2.19}$$

$$Bl = Bl_0 + b_1 x + b_2 x^2 \tag{2.20}$$

$$k = k_0 + k_1 x + k_2 x^2 \tag{2.21}$$

$$L_E = L_{E0} + l_1 x + l_2 x^2 \tag{2.22}$$

Substitution of the nonlinear terms into the governing equations and neglecting terms with order higher than three results in the following nonlinear differential equation for voice coil excursion:

$$\begin{aligned}&\alpha x + \beta\dot{x} + \gamma\ddot{x} + \delta\dddot{x}\\ &+ aE_g x + bx^2 + cx\dot{x} + dx\ddot{x} + ex\dddot{x}\\ &+ f\dot{x}^2 + g\dot{x}\ddot{x} + AE_g x^2 + Bx^3 + Cx^2\dot{x}\\ &+ Dx^2\ddot{x} + Ex^2\dddot{x} + Fx\dot{x}^2 + Gx\dot{x}\ddot{x} = E_g\end{aligned} \tag{2.23*}$$

*See Kaizer (1987) for definition of terms.

2.5.1 Solution of nonlinear differential equation by series expansion

Kaizer assumed the voltage E to vary sinusoidally according to $E_0 \cos\omega t$, the voice coil excursion therefore satisfies the series expansion:

$$\begin{aligned}x = B_0 &+ A_1 \sin(\omega t) + A_2 \sin(2\omega t) + A_3 \sin(3\omega t) + \cdots\\ &+ B_1 \cos(\omega t) + B_2 \cos(2\omega t) + B_3 \cos(3\omega t) + \cdots\end{aligned} \tag{2.24}$$

Truncated at the k^th term and substituted into the nonlinear differential equation this yields an equation that must be satisfied for any t in order for $\sin(n\omega t)$ and $\cos(n\omega t)$ terms to disappear. The result is a set of 2k nonlinear equations with 2k unknowns and a dependent equation that determines B_0, which can be solved numerically.

2.5.2 Solution of nonlinear differential equation by direct integration

The nonlinear differential equation may also be solved by direct integration. This can be achieved by writing the differential equation as a set of first order differential equations and integrating these numerically.

2.5.3 Volterra series expansion

Kaizer describes these methods as cumbersome; with series expansion a whole new set of nonlinear equations must be written to incorporate an additional nonlinear factor, and with direct integration harmonic analysis is required to determine higher order harmonics. Kaizer therefore proposed the use of a Volterra series expansion for the modelling of loudspeaker nonlinearities.

A Volterra series is a functional series that allows the expression of the relationship between input and output of a nonlinear system. If the system is time invariant the Volterra series expansion is as follows:

$$\begin{aligned} y(t) = & \int_{-\infty}^{\infty} h_1(\tau_1)x(t-\tau_1)d\tau_1 + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 \\ & + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_3(\tau_1,\tau_2,\tau_3)x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)d\tau_1 d\tau_2 d\tau_3 \\ & \cdots + \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h_n(\tau_1,\tau_2,\ldots,\tau_n)x(t-\tau_1)x(t-\tau_2)\cdots x(t-\tau_n)d\tau_1\cdots d\tau_n + \cdots \end{aligned} \tag{2.25}$$

where for n=1,2,...,

$$h_n(\tau_1,\ldots,\tau_n) = 0 \qquad \text{for any } \tau_j < 0, \qquad j = 1,2,\ldots,n$$

An alternative form is:

$$y(t) = H_1[x(t)] + H_2[x(t)] + H_3[x(t)] + \cdots + H_n[x(t)] + \cdots \tag{2.26}$$

where $H_n[x(t)] = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} h_n(\tau_1,\ldots,\tau_n)x(t-\tau_1)\cdots x(t-\tau_n)d\tau_1\cdots d\tau_n$ (2.27)

H_n is known as the nth order Volterra operator.

Isolating the first term of (2.25), the first order Volterra operator, it may be observed that it is of the same form as the convolution integral, which implies that the system response for any given input can be obtained by convolving the given input with the system unit impulse response, thus the latter uniquely characterises the system. In (2.25) therefore, $h_1(\tau_1)$ may be interpreted as the first order system impulse response and the system characteristic in one dimension. The 2nd term of (2.25) is of the same form as the two-dimensional convolution integral and hence $h_2(\tau_1,\tau_2)$ may be considered representative of a two-dimensional impulse response, and in the nth term, $h_n(\tau_1,\ldots,\tau_n)$, a n-dimensional impulse response. An nth order nonlinear system can therefore be uniquely characterised by the sum of n Volterra operators of order 1 to n.

The Volterra operators being convolutions also means the series has memory.

The attribute of memory allows the Volterra series to be used to form a representation of the transfer function of the loudspeaker. Ordinary power series would not be suitable, due to the dispersive nature of the loudspeaker which requires previous input values to be considered (Kaizer 1987). However, Kaizer did not extend the transfer function beyond the 3rd order Volterra operator, as the increased computational burden became unjustifiable in comparison to the improved accuracy obtained.

Kaizer measured the force factor, voice coil inductance and suspension stiffness (including enclosure air load) as a function of voice coil excursion using an accelerometer mounted on the voice coil former, and applied a least-squares curve fitting method to evaluate the 2nd and 3rd order Volterra series coefficients, corresponding to the 2nd and 3rd order distortion components of the voice coil acceleration (as it is approximately proportional to sound pressure, the perceptible response of a loudspeaker (Frank et al, 1992)). The derived 2nd and 3rd order harmonic distortion curves showed some qualitative correlation to the measured distortion data, however the quantitative correlation was poor. The coefficients of the excursion dependent terms in the nonlinear characteristics were therefore modified to fit the actual measured distortion responses (previously they were measured directly, independent of their effect on the response). The resultant distortion model shows reasonable quantitative agreement with the measured distortion at frequencies below 200 Hz. The modified coefficients were used to determine 2nd and 3rd order intermodulation distortion, which also resulted in only a reasonable correlation with measured data.

Although it is possible to apply the Volterra series expansion to time variant systems, the resultant expression is theoretically and experimentally involved (Schetzen, 1989). Hence the transfer function determined by Kaizer was a time invariant representation and therefore did not incorporate the time varying parameters. Kaizer's model also excluded hyteresis effects and the generation of subharmonics at high drive levels due to the Volterra series' restraint on input amplitude.

Further limitations of the Volterra series include the measurement of the Volterra operators, which may only be determined if each operator can be separated from the total system response. This is only possible for systems of a finite order, hence approximations must be made for infinite order systems. Also, the Volterra series representation may only converge over a limited range of the system input amplitude (Schetzen, 1989) and if the nonlinearities are weak (kaizer, 1987).

Kaizer alludes to the use of the model in a distortion reduction circuit, using the inverse of the 2nd order Volterra operator to filter out the 2nd order nonlinear terms (the realisation of the 3rd order filter was considered too complex) for a voltage drive loudspeaker and both 2nd and 3rd order operators for current drive, however no results are presented.

Frank, Reger and Appel (1992) applied a Volterra series, also truncated at the 3rd order operator and with the assumptions that the loudspeaker was a weakly nonlinear, time invariant system, to a linearisation scheme. The transfer function related input voltage to diaphragm acceleration, with the acceleration measured using a laser vibrometer. The linearisation scheme preserved the linear part of the transfer function and removed the 2nd and 3rd order nonlinear components, as described in Kaizer's paper. Frank et al acknowledged that this method generated new higher order nonlinearities, however the Sound Pressure Level (SPL) of these distortions were considered low enough not to deteriorate the linearisation. The 3rd order operator was approximated in order to reduce the computational load and hence enable real time processing. A modification of the Weiner G-operators (a generalisation of the Volterra operator) was proposed in order to obviate the need for a white noise signal during the identification of the operators, however, stationary, zero-mean Gaussian noise was still required. It was also suggested that the LMS algorithm be used to determine the coefficients of the G-operators. The LMS algorithm would utilise the difference between the model output and actual loudspeaker output as an error signal to adapt the coefficients to acceptable values. This eliminated the requirement of gaussianity of the input signal, however a Gaussian input would increase the speed of adaptation. A pseudo noise signal was employed during coefficient identification, and the resultant Volterra filter was found to reduce harmonic distortion at all frequencies, most significantly in the region of 100Hz.

In the same year, Klippel, (1992a), also achieved good agreement between measured distortions and those calculated using a 3rd order Volterra series model for small voice coil excursions, however, significant discrepancies were observed in the large signal range. The use of a higher order Volterra series was discounted due to its complexity and an alternative method using an adjusted Volterra model was proposed. This approach incorporated coefficients calculated using the harmonic balance method which involves the expression of the loudspeaker voice coil displacement as a Fourier series, where the first terms of the truncated Fourier series may be assumed to approximate the fundamental and dc component of the voice coil displacement. These can be substituted into the characteristic nonlinear differential equation, and by comparing the coefficients determined for each harmonic, a further set of differential equations may be derived which can be solved using numerical methods. Substituting functions representing the first order response and the dominant nonlinearities derived using the harmonic balance method for the corresponding constants in the Volterra series resulted in a model with improved correlation to the measured loudspeaker response in the large signal domain when compared to that of the original 3rd order Volterra series and resulted in a lower computational load than employing a higher order Volterra series.

The linear parameters of the model can be measured simply, however the nonlinear parameters require either a static measurement and subsequent least-square curve fitting to determine the Volterra coefficients, or a two-tone-intermodulation measurement to determine the coefficients dynamically. Both the aforementioned methods utilise a microphone to measure SPL in the small signal domain, however, large signal responses require specialised techniques such as laser interferometry.

2.6 PERCEPTION OF NONLINEAR DISTORTION

There is debate as to how the perception of nonlinear distortion may be measured, as it is impossible to determine if any two people perceive a sound in the same way. However, it is generally agreed that certain sounds may be classified as distortion, and are commonly described as harshness or roughness, or in terms of sounds that were not present in the original signal such as crackles or clicks. The most commonly applied measures of distortion are total harmonic distortion and intermodulation distortion. Fielder and Benjamin, (1987), proposed that the level of audibility of nonlinear distortion in subwoofers was above 3 per cent of the fundamental for the 2nd harmonic, above 1 per cent of the fundamental for the 3rd harmonic and above 0.1-0.3 per cent of the fundamental for higher harmonics. However, the perception of distortion does not correlate highly with total harmonic and intermodulation distortion measurements. This is due to distortion perception's dependency upon factors such as the frequency separation and relative phase of the distortion products and input signal components, which is not considered in the total harmonic and intermodulation distortion measurements, and also that the measurements are usually obtained using sinusoidal test signals, which may not stimulate distortion scenarios that may occur with more realistic input signals (Tan, Moore, Zacharov, 2003). The most accurate model would therefore be based upon a measure of the listener's perception of the distortion. Methods of quantifying the perception of nonlinear distortion have been developed (Tan, Moore, Zacharov, 2003), along with more qualitative methods (Mason et al, 2001), however they are reliant upon subjective measurements, which is the predicament of all sound perception measurements.

3 ARTIFICIAL NEURAL NETWORK THEORY

3.1 ARTIFICIAL NEURAL NETWORKS

An Artificial Neural Network (ANN) is designed to model the structure and the processes of a biological neural network. Biological neural networks are present in the nervous systems of biological organisms and are the mechanisms through which pattern recognition, perception, motor control and all other life support functions are achieved. The biological neural network is made up of a network of neurons, which consists of a soma connected to a long axon, which ends in several dendrites, which in turn are connected to other neurons in the network. This is illustrated in figure 3.1.

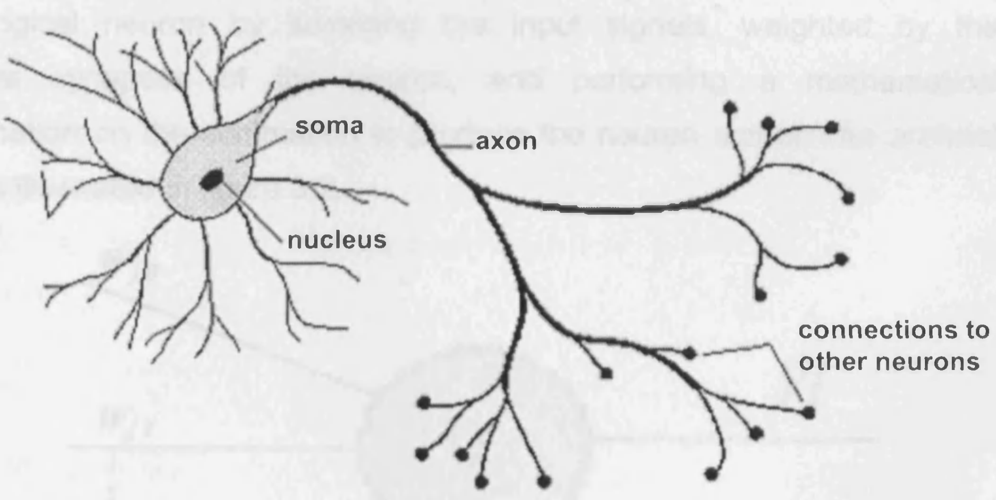


Figure 3.1 – Schematic of a biological neuron (Fraser, 1998)

The soma is the cell body of the neuron where the electrical input signal is received and processed and an electrical output generated, the axon is the transmitter of the pre-synaptic electrical signal and the dendrites are the receivers of the post-synaptic electrical signal. The terminal of the axon does not make direct contact with the dendrites; there is a small gap, known as a synapse. At the terminal of the axon the electrical signal triggers the release of chemical messengers (neurotransmitters) which diffuse across the synaptic gap into receptors on the surface of the dendrites, which generates the post-synaptic electrical signal. Some synapses create stronger post-synaptic electrical signals than others, as a result of the pre-synaptic terminals of the axon releasing a greater quantity of neurotransmitters or the post-synaptic dendrites having more receptors. The strength or 'weight' of the synapse determines the influence of each neural connection, which may be modified by the presence of chemicals known as neuro-modulators. This facilitates learning in the biological neural network, and is the fundamental process that is simulated during the training of artificial neural networks.

An artificial neural network consists of several layers of artificial neurons, which are computational elements designed to emulate a biological neuron. Each neuron consists of a set of 'synaptic' links to other neurons in the network, which are weighted in order to simulate the synaptic gap of a biological neuron. A summation and activation function simulate the soma of the biological neuron by summing the input signals, weighted by the respective synapses of the neuron, and performing a mathematical transformation on the summation to produce the neuron output. An artificial neuron is illustrated in figure 3.2.

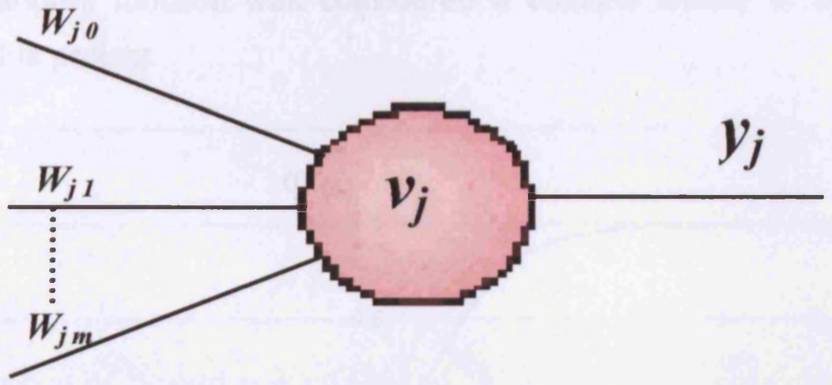


Figure 3.2 – Schematic of an artificial neuron *j*

The sum of the weighted neuron inputs plus a bias may be referred to as the induced local field $v_j(n)$.

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n) \qquad (3.1)$$

where $w_{ji}(n)$ is the synaptic weight connecting the output of neuron i to the input of neuron j, $y_i(n)$ is the signal at the output of neuron i, m is the total number of inputs (excluding the bias) applied to neuron j and n refers to the nth training data pattern presented to the ANN. The synaptic weight w_{j0} equals the bias applied to neuron j.

After the application of the activation function the output of neuron j is $y_j(n)$.

$$y_j(n) = \varphi_j(v_j(n)) \tag{3.2}$$

A commonly used activation function is the sigmoid function, as it is a good approximation of the mean firing rates of a biological neuron (Hutt, 2002) and is easily differentiable (as required by the back propagation algorithm, see section 3.2). Equation 3.3 and figure 3.3 illustrate the hyperbolic tangent function, the form of the sigmoid function used during this project.

$$\varphi(s) = \tanh(s) \tag{3.3}$$

Highly nonlinear systems have been successfully identified by ANNs using sigmoidal nonlinear processing elements (Pham, Liu, 1995), hence the hyperbolic tangent function was considered a suitable choice of activation function for this project.

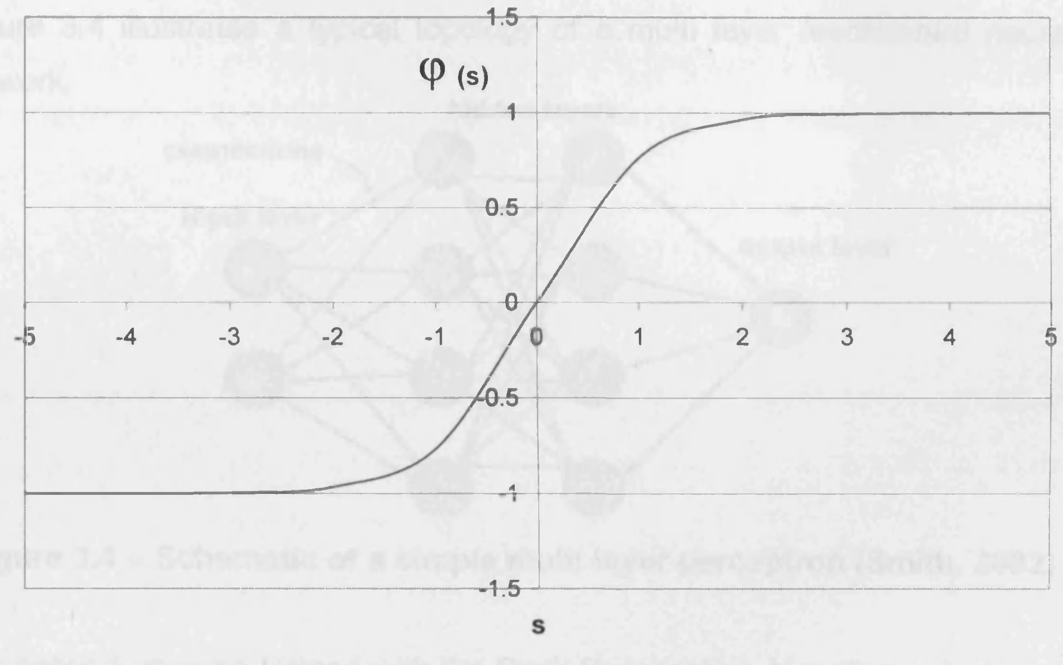


Figure 3.3 – Hyperbolic tangent function

There are three main neural network architectures; single layer feedforward networks, multi layer feedforward networks and recurrent networks. A single layer network has a single layer of computational neurons (the output layer) plus an input layer. When the network is executed, the input variable values are placed in the input neurons, and passed to the output layer neurons via the synaptic weights. The neurons in the output layer calculate their output, which is the output of the network. A multi layer network has an input layer, one or more hidden layers containing computational neurons and an output layer, also containing computational neurons. When the network is executed, the input variable values are placed in the input neurons, as in the single layer network, and the hidden layers and output layer are progressively executed. Each neuron calculates its output value which is then passed on to the neurons in the proceeding layer, via the synaptic weights. When the entire network has been executed, the outputs of the output layer act as the output of the entire network. Recurrent networks include internal feedback that allows the network's hidden units to be influenced the previous network output, and thus have the added attribute of memory.

3.2 FEEDFORWARD NETWORKS – THE MULTI LAYER PERCEPTRON

Figure 3.4 illustrates a typical topology of a multi layer feedforward neural network.

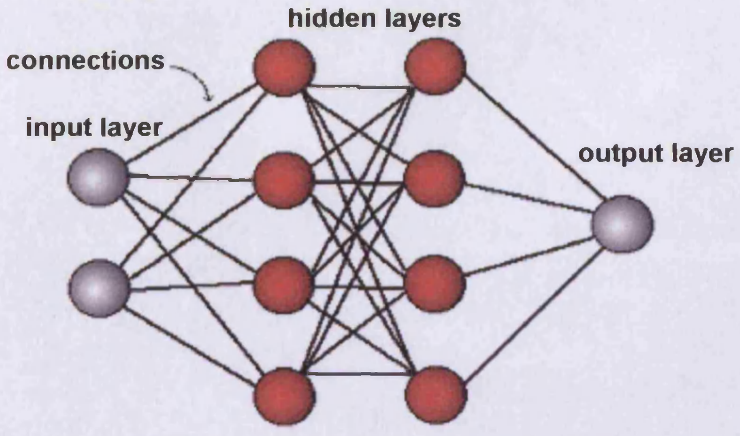


Figure 3.4 – Schematic of a simple multi layer perceptron (Smith, 2002)

The network may be trained with the Back Propagation Algorithm, which was introduced as the Generalised Delta Rule by Rumelhart, et al (1986). In the training process inputs whose desired responses are known are applied to the network. For example, when training an ANN to emulate the Harman/Becker end of line test (see section 7) an approved loudspeaker has an ideal desired response value of +1 and a rejected loudspeaker a desired response of -1. The actual output value generated by each output neuron is compared with the desired output value and used to calculate an error value, $e_j(n)$.

$$e_j(n) = d_j(n) - y_j(n) \tag{3.4}$$

where $d_j(n)$ is the desired output of neuron j for the nth data pattern in the training set, and $y_j(n)$ is the output calculated by the network.

A measure of the network's performance may be gained from the mean square error or sum of squared errors over the whole training sample, defined as a function of the free parameters of the network (i.e. the weights). This may be visualised as a multidimensional error surface with the free parameters as coordinates (Haykin, 1999). During the training process the gradient of this error surface is calculated in order to determine a suitable alteration to the weights that will move the operating point of the network to a lower position on the error surface.

Each neuron in the network contributes to the position of the network on the error surface. This position may be represented by the average error energy (sum of error squares) of the network.

$$\varepsilon_{av} = \frac{1}{N}\sum_{n=1}^{N}\varepsilon(n) \tag{3.5}$$

where ε_{av} is the average error energy, calculated by summing the instantaneous total error energy ($\varepsilon(n)$) over all the samples, N, in the training set. $\varepsilon(n)$ is the sum of the error signals squared of all neurons in the output layer for the instantaneous data pattern, n.

$$\varepsilon(n) = \tfrac{1}{2}\sum_{j\in C} e_j^2(n) \tag{3.6}$$

where set C includes all the neurons in the output layer.

The aim of the back propagation algorithm is to minimise the average error energy function with respect to the free parameters of the network (the synaptic weights and bias values). This can be achieved using an optimisation technique based upon the method of steepest descent.

The method of steepest descent states the condition for optimality as:

$$\nabla \varepsilon_{av}(w^*) = 0 \tag{3.7}$$

where w^* is the weight vector of an optimal solution and ∇ is the gradient operator:

$$\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, \ldots, \frac{\partial}{\partial w_m} \right]^T \tag{3.8}$$

and $\nabla \varepsilon_{av}(w)$ is the gradient vector of the average error energy function:

$$\nabla \varepsilon_{av}(w) = \left[\frac{\partial \varepsilon_{av}}{\partial w_1}, \frac{\partial \varepsilon_{av}}{\partial w_2}, \ldots, \frac{\partial \varepsilon_{av}}{\partial w_m} \right]^T \tag{3.9}$$

The weight vector is initiated randomly and adjustments made in the direction of steepest descent, which is the opposite direction to the gradient vector:

$$w(n+1) = w(n) - \eta \nabla \varepsilon_{av}(w)$$ (3.10),Haykin (1999)

where η is the learning rate.

However, the summation of total error energy over the training set to obtain the average error energy is not possible with sequential learning (where the ANN weights are updated after the presentation of each training sample, the alternative is batch learning where the weights are updated after the presentation of all of the training samples in the data set). Therefore an estimate of the method of steepest descent that utilises the error calculated for the individual data pattern is required.

The gradient vector for data pattern n, evaluated at the point $w_{ji}(n)$ is:

$$\nabla \varepsilon(n) = \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} \tag{3.11}$$

This may be written as:

$$\nabla \varepsilon(n) = \frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = \frac{\partial \varepsilon(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \tag{3.12}$$

Differentiating equation 3.6 with respect to $e_j(n)$ gives:

$$\frac{\partial \varepsilon(n)}{\partial e_j(n)} = e_j(n) \tag{3.13}$$

Differentiating equation (3.4) with respect to $y_j(n)$ gives:

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \tag{3.14}$$

Differentiating equation (3.2) with respect to $v_j(n)$ gives:

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n)) \tag{3.15}$$

Differentiating equation (3.1) with respect to $w_{ji}(n)$ gives:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n) \tag{3.16}$$

Substituting equations (3.13) to (3.16) into equation (3.12) results in the following approximation of the gradient vector at point $w_{ji}(n)$:

$$\frac{\partial \varepsilon(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j'(v_j(n))y_i(n) \tag{3.17}$$

Substituting equation (3.17) into equation (3.10) gives the approximation of the gradient descent method used by the back propagation algorithm:

$$w(n+1) = w(n) + \eta e_j(n)\varphi_j'(v_j(n))y_i(n) \tag{3.18}$$

Equation (3.18) may also be written as:

$$w(n+1) = w(n) + \eta \delta_j(n) y_i(n) \tag{3.19}$$

where $\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$ (3.20)

and is known as the local gradient:

$$\delta_j(n) = -\frac{\partial \varepsilon(n)}{\partial e_j(n)}\frac{\partial e_j(n)}{\partial y_j(n)}\frac{\partial y_j(n)}{\partial v_j(n)} \tag{3.21}$$

$$= -\frac{\partial \varepsilon(n)}{\partial v_j(n)} \tag{3.22}$$

The weight alteration is therefore dependent upon the error signal $e_j(n)$ at the output of neuron j. When neuron j is an output neuron it is a simple case of comparing the neuron output to the desired output as in equation (3.4). When neuron j is a hidden neuron, a direct comparison between desired and actual output is not possible. This problem is solved by the back propagation algorithm in the following way:

The local gradient, equation (3.21), may be written as:

$$\delta_j(n) = -\frac{\delta\varepsilon(n)}{\delta y_j(n)}\frac{\delta y_j(n)}{\delta v_j(n)} \tag{3.23}$$

Substituting equation 3.15 into equation (3.23) :

$$\Rightarrow \delta_j(n) = -\frac{\delta\varepsilon(n)}{\delta y_j(n)}\varphi_j'(v_j(n)) \tag{3.24}$$

Substituting the subscript j for k into equation (3.6) to indicate the neuron is in the output layer gives:

$$\varepsilon(n) = \frac{1}{2}\sum_{k\in C} e_k^2(n) \tag{3.25}$$

Differentiating with respect to $y_j(n)$ gives:

$$\frac{\delta\varepsilon(n)}{\delta y_j(n)} = \sum_k e_k \frac{\delta e_k(n)}{\delta y_j(n)} \tag{3.26}$$

and applying the chain rule:

$$\Rightarrow \frac{\delta\varepsilon(n)}{\delta y_j(n)} = \sum_k e_k(n)\frac{\delta e_k(n)}{\delta v_k(n)}\frac{\delta v_k(n)}{\delta y_j(n)} \tag{3.27}$$

Equations (3.2) and (3.4) may be combined to give:

$$e_k(n) = d_k(n) - \varphi_k(v_k(n)) \tag{3.28}$$

Differentiating equation (3.28) with respect to $v_k(n)$ therefore gives:

$$\frac{\delta e_k(n)}{\delta v_k(n)} = -\varphi_k'(v_k(n)) \tag{3.29}$$

Equation (3.1) may be written as:

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{3.30}$$

for the connection between the hidden layer and output layer. Equation (3.30) may be differentiated with respect to $y_j(n)$ to give:

$$\frac{\delta v_k(n)}{\delta y_j(n)} = w_{kj}(n) \tag{3.31}$$

Substituting equations (3.29) and (3.31) into equation (3.2) gives:

$$\frac{\delta\varepsilon(n)}{\delta y_j(n)} = -\sum_k e_k(n)\varphi_k'(v_k(n))w_{kj}(n) \tag{3.32}$$

and substituting equation (3.20) with subscript j replaced with k gives:

$$\frac{\delta\varepsilon(n)}{\delta y_j(n)} = -\sum_k \delta_k(n) w_{kj}(n) \tag{3.33}$$

Substituting equation (3.33) into equation (3.24) gives:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \tag{3.34}$$

(Haykin, 1999)

All terms in equation (3.34) are derivable for hidden neurons, therefore the weight alteration described in equation (3.19) may be achieved for a connection between a neuron in a hidden layer and a neuron in the output layer, or a connection between 2 neurons in adjacent hidden layers without knowledge of the desired output of the neuron.

3.2.1 Learning rate

The learning rate, η, determines the magnitude of the change in weight values at each iteration. A high learning rate value will result in a relatively large change in weight values and hence a large step along the network error surface. This may result in faster convergence of the network to the optimum solution. However, it may cause instability in the network output that may prevent the convergence of the network. A lower learning rate value would reduce the possibility of divergence of the network away from the optimum solution, however, would result in the requirement for more iterations before the network converges to the optimum solution, and hence a longer training period.

3.2.2 Momentum

A momentum term is often added to (3.19) that allows the weight change from the previous iteration to influence the new weight change. Equation (3.19) becomes:

$$w_{ji}(n+1) = w_{ji}(n) + \eta\delta_j(n) y_i(n) + \mu\Delta w_{ji}(n) \tag{3.35}$$

where $\Delta w_{ji}(n)$ is the weight change from the previous iteration n and μ is the momentum coefficient (Pham, Liu, 1995). In the event of two consecutive weight changes in the same direction along the error surface, this will accelerate the training process. In the event of two consecutive weight changes resulting in movements along the error surface in opposite directions, the momentum term will reduce the influence of the inconsistent weight change on the value of the weights and thus have a stabilising effect upon the training process, as it prevents the network diverging significantly from the optimum solution.

The training process is repeated until the error between the actual and desired response reaches a level acceptable to the application. The network should now be configured so as to produce an output for previously unseen data patterns consistent with the input/output relationship determined from the training data.

3.2.3 ANN performance validation

Throughout the research, the ANN performance was evaluated with a validation data set comprised of previously unseen data. The measure of performance was the root mean square error, e_{rms}, calculated over the whole validation set:

$$e_{rms} = \sqrt{\frac{\sum_{n=1}^{N}\left(d_j(n) - y_j(n)\right)^2}{\sum_{n=1}^{N} d_j(n)^2}} \qquad (3.36)$$

where $d_j(n)$ is the desired output of neuron j for the nth data pattern in the validation set, $y_j(n)$ is the output calculated by the network for neuron j for the nth data pattern, and N is the total number of data patterns in the validation set.

3.2.4 Tapped delay line

In order to model dynamic systems, the neural network requires knowledge of previous states of the system. The most frequently used method of supplying this information to a multi layer perceptron is to employ a tapped delay line

(Waibel et al, 1989). The tapped delay line stores previous inputs and outputs of the system so that they may be used simultaneously with the current system input as the network input. The neural network therefore maps an input vector consisting of the current system input plus previous system inputs and outputs to the current system output.

This method has several disadvantages, such as a significantly increased training time in comparison to a network trained with a single input. It also requires a similar input vector when the network is engaged in its intended application, unless a significant degree of network performance is sacrificed.

3.3 RECURRENT NETWORKS – THE ELMAN NETWORK

In addition to feedforward connections, a recurrent neural network has feedback connections, which enable the network to learn temporal sequences of events. Recurrent network architectures vary in their degree of connectivity. Neurons in fully connected (fully recurrent) networks have feedforward and feedback connections with all other neurons in the network, all of which are trainable. The structure of partially connected (partially recurrent) networks is similar to that of a feedforward network, with an additional set of neurons (known as context units) that receive feedback from selected neurons in the network. Therefore, feedback information about the network's state at iteration k is fed back and incorporated into the network at iteration $k+1$, thus the internal representations that develop in the neurons during training have a temporal perspective i.e memory (Elman, 1990). This allows the network to identify dynamic systems, without the use of a tapped delay line and large input vectors. Only the feedforward connections of a partially recurrent network are trainable.

Simple partially recurrent ANNs were introduced in the late 1980's by several researchers including Rumelhart, Hinton and Williams to learn strings of characters (Rumelhart et al, 1986, Medsker and Jain, 2000). The recurrent networks employed in this project were the Elman network (Elman, 1990) and modified Elman network (Liu, 1993, Pham and Liu, 1993). Elman also developed this network to learn strings of characters. The structure of the Elman network is extremely similar to that of a multi layer perceptron with a single hidden layer, with the addition of a set of context units in the input layer, as illustrated in figure 3.5. The context units receive feedback from the hidden layer and thus in the proceeding iteration supply the hidden units with information regarding their activation from the previous iteration. Hence the network maps a combination of the current input and the previous internal state of the network to the current desired output.

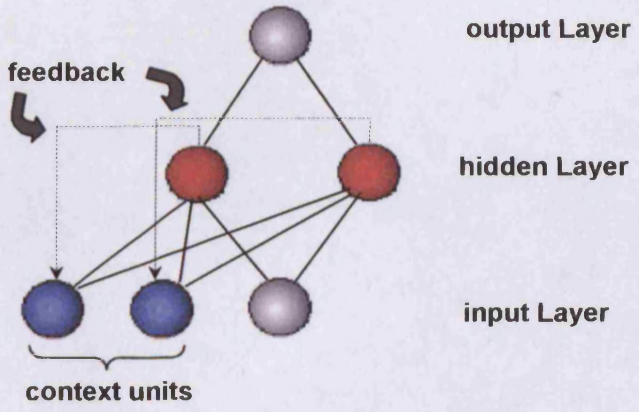


Figure 3.5 – Schematic of an Elman network

All neuron activations are the same as the multi layer perceptron described in section 3.2, except those in the hidden layer, which becomes:

$$y_j(k) = \varphi_j\left(\sum_{i=0}^{m} w_{ji}(k)y_i(k) + \sum_{c=0}^{p} w_{jc}(k)y_j(k-1)\right) \tag{3.3}$$

where p is the number of context units (which corresponds with the number of neurons in the hidden layer) plus 1 (for the neuron bias), and k is the current time step, $w_{ji}(n)$ is the synaptic weight connecting the output of neuron i to the input of neuron j, and $w_{jc}(n)$ is the synaptic weight connecting the output of context unit c to neuron j.

Only the feedforward connections between neurons and context units are modifiable, the weights of the context layer connections with the hidden layer are fixed. Therefore the network may be trained with the back propagation algorithm, also described in section 3.2.

In his PhD thesis (Liu, 1993), Liu suggested modifying the Elman network to include self-connections in the context units, as illustrated in figure 3.6. The modification to the Elman network involves the incorporation of past context unit values, and hence hidden layer activations from further in the past than just one time step. The previous context unit values are fed back through self-connections in the context units. The activations of the context units in the modified Elman network therefore become:

$$y_c(k) = \alpha y_c(k-1) + y_j(k-1) \tag{3.38}$$

where $y_c(k)$ is the activation of the context unit and α is the feedback gain of the self-connections. The value of α is universal throughout all self-connections and is not modified by the training algorithm (Liu, 1993). It is set between 0 and 1, with a higher value resulting in an increased influence of previous context unit (and therefore hidden layer) activations on the current activation.

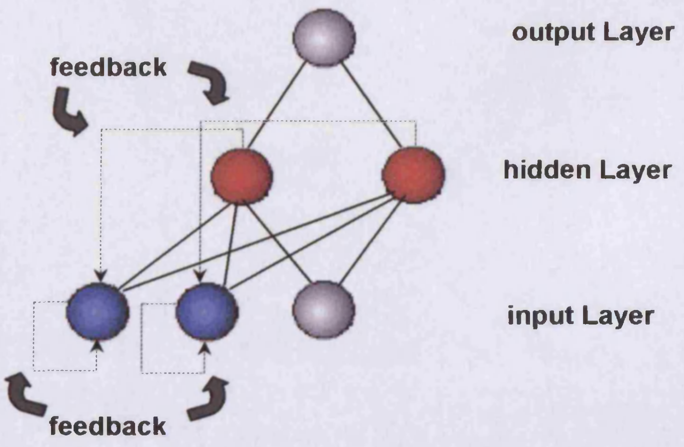


Figure 3.6 – Schematic of a modified Elman network

3.4 ANN APPLICATIONS

Artificial Neural Networks have been applied to identification problems in many fields. Their ability to model a relationship between system input / output patterns without knowledge of the system makes them extremely useful tools. System models can be generated relatively quickly and with little expert knowledge of the system. Below several applications are discussed in detail.

3.4.1 Recurrent networks for dynamic system identification

Neural networks have been employed extensively in the modelling of dynamic systems (Lippmann, 1987). In Pham and Liu (1993) the basic and modified Elman networks discussed in section 3.3 are employed in the identification of dynamic systems. Recurrent networks are well suited to system identification as they do not require spatial representation of the system inputs and outputs (such as that generated by a tapped delay line) in order to extract a temporal relationship from the data. Pham and Liu (1993) simulated various linear and non linear systems in order to determine the modelling performance of the two networks. The basic Elman network performed well with first order linear systems, and was in fact superior to the modified Elman network, however its performance deteriorated with higher order linear systems and the modified network's performance improved. It was determined that optimum α parameter values were dependent upon the order of the modelled system. The modified network's performance was also superior when modelling nonlinear systems. Optimum α parameter values were also found to be dependent upon degree of non linearity of the system to be modelled.

3.4.2 Feedforward ANNs for online control

ANNs may be employed as system controllers. The ANN is trained to generate an inverse model of the system, and can therefore be used to determine the required system inputs for any desired system output. However, this method has several disadvantages; the ANN must be trained off-line and therefore cannot control the system during this time and, once trained, the ANN cannot take into account any changes in the system's behaviour until the next training session. Psaltis, Sideris and Yamamura (1988) proposed an alternative learning algorithm called 'specialized learning' which allows the ANN controller to learn in an on-line and autonomous way. The error value used to adjust the weights is calculated differently in specialized learning; instead of the difference between the actual system input and that generated by the ANN model (see figure 3.7), the difference between the desired and actual system outputs is employed (see figure 3.8).

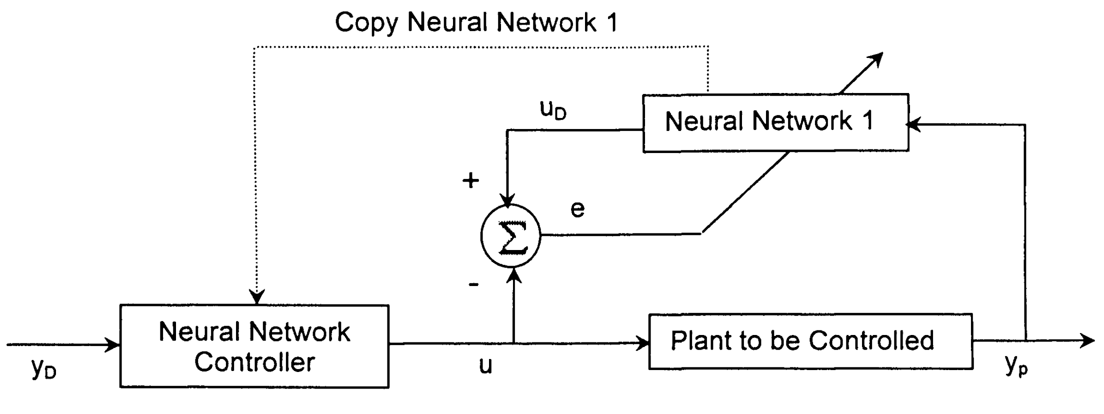


Figure 3.7– Adaptive inverse neuro-control with general learning (Wang, Bao, 2000)

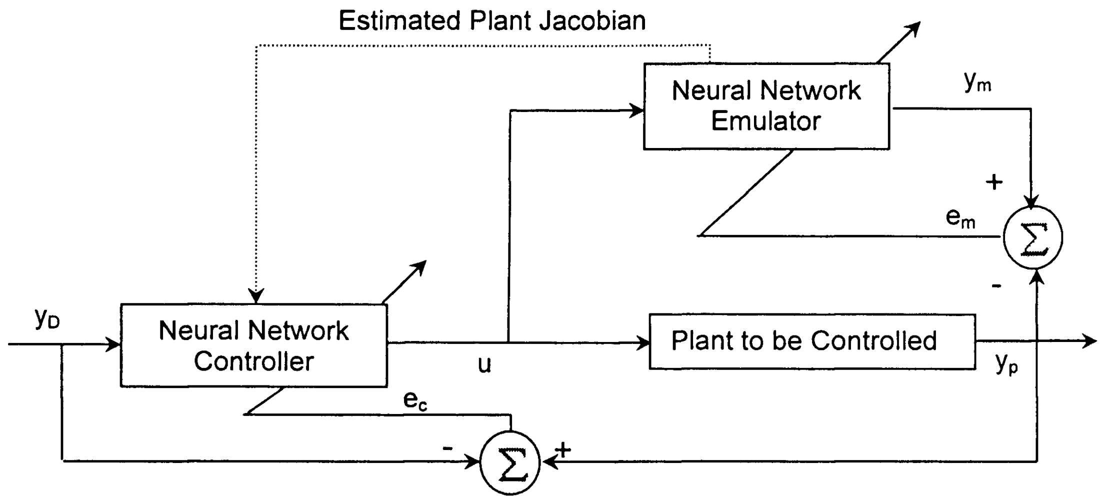


Figure 3.8 – Adaptive inverse neuro-control with specialized learning (Wang, Bao, 2000)

In figure 3.7, u is the control input, y_D and y_P are the desired and actual plant outputs respectively, u_D is the input calculated by the neural network which is compared to the actual plant input in order to generate an error signal, e with which to train the ANN. In figure 3.8, u, y_D and y_p are as figure 3.7 and y_m is the output calculated by the neural network emulator which is compared to the actual plant output in order to generate the error signal e_m which is used to train the neural network emulator, and e_c is the error generated by the comparison of the desired plant output and actual plant output, which is used to train the neural controller.

However, the unknown system lies between the controller and the output error, therefore the standard back propagation algorithm cannot be applied to adjust the ANN's weights (Wang, Bao, 2000). Various approaches have been proposed to overcome this, including:

- Considering the plant as an additional, unmodifiable, layer of the ANN controller in conjunction with a modification to the back propagation algorithm in order to incorporate the additional layer (Psaltis, Sideris and Yamamura, 1988). This method requires an estimate of some parameters required by the ANN controller, which may be achieved by utilising basic qualitative knowledge of the plant (Saerens, Soquet, 1991).
- Utilising a second ANN to emulate the plant (Nguyen, Widrow, 1990).

Although theoretically these algorithms should improve the accuracy of an on-line ANN controller, the degree of estimation required is still high, and thus the performance of such controllers is often unsatisfactory.

3.4.3 Feedforward ANNs for Fault Detection

3.4.3.i Visual Inspection

ANNs have found many applications involving pattern recognition. An example of such an application is in the automated visual inspection of components, post production. Pham and Bayro-Corrochano (1994) proposed the use of a neural network to recognise surface defects and to classify the shape of the inner perimeter of a valve stem seal for use in a car engine. Images of the valve were captured using CCD (Charge Coupled Device) cameras and preprocessed to extract feature vectors that were used as input to the neural networks. One network was trained for the surface defect recognition task and one for the classification of the shape of the seal inner perimeter. Both networks achieved good classification accuracy, with the former correctly classifying 93 per cent and the latter 83 per cent of previously unseen feature vectors. This could be considered a considerable improvement upon existing methods of visual inspection, which involved the manual inspection of the valve on a sampling basis. The results of this study highlighted several advantages of employing neural networks in such a task; a greater number of components could be assessed due to the speed of the operation, and therefore increased confidence in the product's quality could be achieved, the system was simple to implement and gave consistent results (Pham, Bayro-Corrochano, 1994).

3.4.3.ii Condition monitoring

Many manufacturers experience significant costs due to loss of production as a result of failure of machinery. This has given rise to the development of condition monitoring techniques in order to detect faults in key components and to provide an indication of the severity of the fault condition. This information can be used to make an assessment of the maintenance requirements of the component in order to minimise costs to the manufacturer.

Bailey and Watton (2002) employed a neural network to monitor leakage fault conditions in an electrohydraulic pressure control system installed in a steel rolling mill. The ANN was trained with pressure and flow rate data measured from the plant under no-fault and fault conditions and successfully diagnosed the existence of leakage in the system, although it could not determine the location of the fault.

Jack and Nandi (2000) used a Genetic Algorithm to dramatically reduce the number of inputs required by an ANN in order to classify bearing faults from frequency domain data. The Genetic Algorithm identified the most significant features in the input data and these alone were used to train the neural network. This method not only reduced the size of the ANN required, but also significantly improved classification accuracy.

Lurette and Lecoeuche (2003) incorporated unsupervised learning and the ability to adapt to new fault conditions into an ANN for detecting faults in hydraulic systems. A network resembling an Radial basis-function (see Haykin, (1999) for a description) was adapted to include unsupervised on-line learning rules that allowed the modification of the size of the hidden and output layers as well as the modification of the weight values when input vectors were significantly different from previously encountered input vectors, thus identifying new fault conditions that evolved over time. The learning rules also allowed the merging of neurons when both their outputs identified the same fault condition, thus keeping the network architecture compact.

3.4.4 Time delay ANNs for speech recognition

Time Delay Neural Networks (TDNNs) may be employed as speech recognition systems (Waibel et al, 1989). The input to the network is a spectrogram generated from speech signals with different time delays. This input vector is used to train the network to extract temporal relationships between acoustic-phonetic events and thus recognise certain phonemes (Bodenhausen, Waibel, 1991), in Waibel et al (1989), these were 'B', 'D', and 'G'. Significant problems occur as a result of the heterogeneous nature of speech signals, even from an individual speaker, the features of the phoneme may vary substantially at each utterance. Waibel et al (1989) overcame this by showing the network a group of spectograms of the same acoustic event, each shifted one time step and applying the regular back propagation forward and backward pass to each one as if they were separate events. This yields different error derivatives for corresponding (time shifted) connections. However, the weights are not updated according to each separate error derivative, but by the average of all corresponding time-delayed weight changes. The network can therefore extract useful acoustic-phonetic features in the input, regardless of when in time they actually occurred (Waibel et al, 1989). The resultant ANN was able to correctly recognise 98.5 per cent of approximately 2000 previously unseen spectograms of 'B', 'D', and 'G' phonemes recorded by 3 different speakers.

3.4.5 Feedforward ANNs for EEG/EKG diagnosis

Electroencephalograms (EEGs) and electrocardiograms (EKGs) are recordings of the electrical activity of the brain and the heart respectively. Kalayci and Özdamar (1995) proposed a neural network approach to diagnose neurological conditions, such as epilepsy, from EEG patterns. The EEG data required significant preprocessing including removal of background noise and manual spike identification by experts (a spike is a transient waveform, clearly defined from background activity). The highest classification accuracy achieved was 91 per cent.

Foo, et al (2002) proposed a similar scheme to diagnose heart conditions from EKG data. A TDNN was successfully trained to recognise the difference between a normal heartbeat and a premature ventricular contraction (one form of abnormal heartbeat) with a classification accuracy of 92 per cent. However, the process was still not totally automated, as part of the preprocessing of the EKG data involved an algorithm to detect the heart beat spikes (spikes are recorded during a heart beat) that required manual adjustment for each EKG pattern in order for the algorithm to successfully extract the spikes. A training algorithm known as Levenberg-Marquardt was used that proved to be significantly faster to converge than the standard propagation algorithm often employed to train a multi-layer perceptron.

3.4.6 ANNs for modelling loudspeakers

Low and Hawksford, (1993) proposed a linearisation scheme for current driven loudspeakers employing a neural network model (current drive eliminates certain nonlinearities experienced in voltage driven loudspeakers (Klippel, 1992b)). The scheme was a refinement of Klippel's mirror filter (see section 4.3). The ANN weights were adjusted using an error signal derived from the difference between actual (nonlinear) loudspeaker cone displacement and a desired response generated from a linear filter. The resultant ANN provided a corrected input current for the loudspeaker which reduced the nonlinearity in its output. Cone displacement was derived from the loudspeaker back emf signal through numerical integration which requires the knowledge of several system parameters which must be measured before the commencement of ANN training. The training sets were generated from a number of sine sweep excitation signals.

Chang et al (1994) used a multi layer perceptron in conjunction with a tapped delay line to model the combined transfer function of loudspeaker and room acoustics, which in theory would also compensate for nonlinearities resulting from room acoustics, however the measurement of such effects would be ambiguous due to the significant variation in the acoustics throughout the listening space.

4 MODELLING STRATEGY

4.1 INTRODUCTION

This chapter presents the strategy employed during the modelling of the loudspeaker transfer function. Issues such as the perception of distortion in a loudspeaker, the accumulation of ANN training data and preprocessing are discussed.

4.2 PERCEPTION OF LOUDSPEAKER PERFORMANCE IMPROVEMENT

The extent to which the reduction of nonlinearities can be perceived by the listener will determine the degree to which it is economic to remove the distortion. The amount of irritation caused to the listener, and the level at which distortion becomes intolerable will vary with individual listeners (Klipsch, 1968). The perception of distortion is also dependent upon listener training; a trained ear will decipher significantly more distortion than that of an untrained ear. Therefore, assuming that the listener with a trained ear is more likely to be purchasing higher end loudspeaker products, it would be economic to remove a greater degree of distortion components from the loudspeaker output in this case. The lower distortion may also distinguish the product from that of competitors, hence providing a lead in the market. In the case of the lower end of the market, it would not be economic to remove as much of the distortion. This assumes that the amount of distortion reduction attainable from the proposed linearisation scheme is proportional to implementation cost. It is likely that this will be the case, as the more accurate model required to reduce distortion to lower levels will require an ANN with a larger topology, smaller learning rate, a greater number of training epochs etc (see section 5.3), which would necessitate a more powerful processor than that required for a less accurate model.

There is also evidence (Griesinger, 2004) that the listener becomes attuned to distortion in a very short period of time, and that to decipher distortion most efficiently two short tracks, containing different levels of distortion, must be played to the listener in quick succession, therefore in the case of a loudspeaker with relatively constant, low distortion levels, the listener will quickly become unable to perceive it.

4.3 BLACK BOX MODELLING

The use of a music input signal meant that the loudspeaker could not be excited in order to extract behavioural information specific to a certain source of nonlinearity; the effects of all sources of nonlinearity culminated in the measured back emf signal. It was therefore not possible to model each type of nonlinearity individually, however, due to the consecutive occurrence and mutual interaction of nonlinear components, the modelling of the nonlinearities singularly and independently of one another would not be an optimum approach. This is because parameters required to isolate the loudspeaker response due a particular nonlinearity may be difficult to obtain without contamination from another nonlinear effect, and the simple amalgamation of individual nonlinear models may not incorporate the possible interactions. The proposed ANN model should incorporate the cumulative effects of all the nonlinearities present in the frequency and amplitude range, as well as any interactions between the nonlinear elements, which may not be assimilated by individual parameter models, assuming that it has been trained with sufficient data to be able to generalise to this extent.

Klippel's mirror filter linearisation scheme (Klippel, 1992b) models each nonlinearity independently and therefore compensates for each nonlinear element individually. This requires that the signal be passed through a series of filters before reaching the loudspeaker. Linearisation schemes that utilise models derived from a Volterra series expansion (Kaizer, 1987, Gao, Snelgrove, 1991, Frank et al, 1992) also model each nonlinearity independently.

Feedforward models, such as Klippel's mirror filter, as well as the Volterra series models, and that proposed by Marshall Leach, (1989), require parameter measurements to realise the model, which is not required in the proposed method. Low and Hawksford, (1993) proposed a refinement to

Klippel's mirror filter by using an ANN model of the inverse transfer function, however, parameters determined through measurement are still utilised in the integration process to derive cone displacement from the back emf signal. The proposed method is a genuinely black box technique that does not require prior knowledge of any parameters and can therefore be applied universally.

4.4 DETERMINATION OF THE TRAINING SET

The acquisition of training data through excitation with a music signal was necessary in order to facilitate the adaptation of the ANN model to alterations in the loudspeaker transfer function during operation. Using a different form of excitation signal, such as a sine sweep or white noise would interrupt the performance of the loudspeaker, which would be unacceptable to the listener who should be completely unaware of the linearisation process. The input voltage to the loudspeaker was the input signal and the back EMF (as discussed in Klippel, (1992b)), inverted to bring it in phase with the input signal, was used as the output signal. The back EMF represented loudspeaker displacement (Newman, 2004).

4.4.1 INITIAL MODEL

Consideration was made of applying a test signal as the sound system was switched on, such as a sine sweep of short duration that would not significantly interfere with the listening experience. The training set obtained from this sine sweep would therefore contain information pertaining to the full frequency range of the loudspeaker, which should result in a superiorly performing ANN model in comparison to an ANN model trained with an incomplete frequency range, which may be the case if the training set is compiled randomly from music signal excitation. Low and Hawksford, (1993), used a number of sine sweep excitation signals to obtain training sets for an ANN model of a loudspeaker, which produced an efficiently performing ANN model.

This method of establishing a robust initial model would result in some disturbance to the listener, also, it does not take into consideration the period between when the user turns on the system and the end of the first training period. A default model is therefore required for this initial period. One method would be to employ the last active model from the previous system operation. This model would integrate the nonlinear effects of system ageing, and those arising from other parameters such as the suspension and force factor. However, it may also incorporate significant nonlinear elements resulting from temperature effects, which would not be present during the early stages of operation, when the loudspeaker is still relatively cold.

An alternative would be to perform this initial training of the ANN model in the factory, using the sine sweep excitation signal that should theoretically result in a more robust model than using data derived from a music signal excitation. The ANN model could then be initialised to this default model every time the system is switched on, and the alteration in long term time dependant parameters incorporated into the first subsequent model update.

4.4.2 SUBSEQUENT MODEL TRAINING

The aim was to update the model at regular intervals during operation in order to incorporate developments in the transfer function that occur over time. The options considered for subsequent model training methods are discussed in the following sections.

4.4.2.i New training set generation for each training session

Collecting a new set of data in the period before training begins, either as small samples taken intermittently throughout the time interval, which could conceivably result in a good range of amplitudes recorded, or one large sample taken just before the beginning of the new training cycle, which may severely limit the range of data recorded, however, it would produce the most up to date model possible. Even with the latter method, this would result in a model trained with data that could be considered out of date, with respect to a music signal which could change significantly over the training period and also to changes in the loudspeaker response due to temperature increases.

The training set may contain limited amplitude and frequency data as only those captured during data sampling will be present. If during the data collection period the loudspeaker does not perform particularly nonlinearly, the ANN trained with that data set will return a linear transfer function and for the period over which the model is active in the linearisation scheme it will be ineffective at removing any nonlinearities. This may be acceptable in that the genre of music that the listener is playing results in the linear operation of the loudspeaker over a significant period. However, if the genre of the music and/or the dominant frequencies in the signal change significantly between the data collection period, the training period and the period during which the model is active in the linearisation scheme, significant distortion may be experienced. This therefore brings into question the generalisation ability of the ANN. The use of online training may overcome this difficulty by immediately incorporating a significant change in loudspeaker behaviour, if the learning rate is sufficiently high.

4.4.2.ii Accumulation of most nonlinear data for training set

The loudspeaker output data could be continuously assessed to determine the degree of nonlinearity (Methods of determining nonlinear content of a signal are discussed in Kantz, Screiber, (2004).) and included in the training set if it lies in the top percentile, with respect to nonlinearity, of the data under consideration. The data could be considered over just one training cycle or over the whole period of operation. Utilising only the most nonlinear data would be beneficial as there would be a significantly improved chance that the ANN would learn the nonlinear response of the loudspeaker. It was ascertained through analysis of the data that the loudspeaker used during this investigation behaved linearly for a significant proportion of the measurement period. Therefore, if a random sample of the data was used for ANN training the possibility existed that the training set would contain a very small proportion of data relating to nonlinear behaviour, and that the resultant model may be a linear one, or close thereto, as the nonlinear behaviour of the loudspeaker has not been incorporated into the model. However, this may be detrimental to the model's generalisation performance as the training set is not representative of the overall behaviour of the loudspeaker, as using only the most nonlinear data in the training set may skew the transfer function towards that which was occurring earlier in its operation, rather than its current (or most recently recorded) behaviour. However, this is unlikely as the transfer function theoretically should become more nonlinear over time due to temperature effects. In order to obtain representative data a random sample is required. There is the possibility of the linearisation scheme adding distortion to the loudspeaker output in both cases - if a randomly obtained training set is used, the resultant model may tend towards a linear transfer function, which will result in distortion being added to the loudspeaker output at instances where it behaves more nonlinearly, or if extracted nonlinear data is used in the training set the model may be overly nonlinear which will lead to instances of additional distortion in the loudspeaker output when the loudspeaker's behaviour is less nonlinear.

4.4.2.iii Accumulation of full frequency and amplitude range training set

A training set could be compiled that contains data pertaining to the full frequency and amplitude range of the loudspeaker. This could take the form of a look up table that is updated with the most recent data obtained for a particular frequency/amplitude. As there is no guarantee that the loudspeaker will perform over its entire frequency or amplitude range in the data collection period, this method would therefore ensure the model encompasses the full scope of the loudspeaker's behaviour, however, a considerably complex algorithm would be required to facilitate the updating of the look up table, as the frequency and amplitude of the data would have to be determined and the storage address then determined. It would also require a sizeable memory space.

4.4.2.iv Continuous ANN training

Continuously updating the ANN weights with errors generated from every sample measured real-time, with each sample used only once, would ensure that training data had a large range of amplitude/frequency content. However, the ANN training process generally performed better over many epochs, where the training set is shown to the ANN repeatedly. There are also logistic problems with direct online learning while the ANN is consecutively filtering the loudspeaker input signal (Narendra, Parthasarathy, 1990). The possibility of using a combination of online training with several training epochs is discussed in section 6.3.

4.4.3 PREPROCESSING

The only preprocessing applied to the input/output signal was phase inversion and a conversion from millivolts to volts, in order to reduce the magnitude of the ANN inputs in line with the magnitude of the initial values of the weights. Without this preprocessing the extremely large initial error values generated during training caused the program to crash. This is an extremely simple preprocessing sequence in comparison to many linearisation schemes, for example (Low, Hawksford, 1993) derive cone displacement from the back emf signal through numerical integration which requires the knowledge of several system parameters. Reducing the required preprocessing simplifies implementation and also reduces processing time, hence reducing delay between input and loudspeaker output.

4.4.4 MEASUREMENT NOISE

Signal measurements were subject to noise, with low amplitude measurements being particularly susceptible. It was observed that below 4.0mV the amplitude was almost indiscernible from the noise in the signal. This data could still be used in the training set under the assumption that the ANN will still be able to extract the transfer function from other elements of the training set. The degree of inaccuracy this may add to the model would be assumed to be low. Experimental results would suggest that this is the case.

A comparison of ANN performance resulting from a training session with a training set that contained a full range of amplitude data and one where the low amplitude, noisy data was removed showed little advantage in removing the noisy data, the ANN appeared to be able to generalise so as not to be affected by the inclusion of the noisy data. However, ANNs trained with a significant proportion of low amplitude, noisy training samples performed inadequately. The ANN could not extract a relationship from the noisy, low amplitude data, whereas it could generalise and achieve acceptable model accuracy from the less noisy data of the full amplitude range sample.

An alternative to the inclusion of the noisy data in the training set would be to employ a high pass filter to allow the low amplitude elements of the input signal to pass through to the loudspeaker without being pre-processed. The low amplitude response of the loudspeaker would be assumed to be linear, hence not require preprocessing. However, experimental results suggest that, assuming the ANN training set contains a relatively small proportion of noisy, low amplitude data, the additional computational load of pre-filtering is unjustified.

Consideration was also made of only employing the linearisation scheme above a threshold level of nonlinearity in the system. This would limit the possibility of the linearisation scheme causing additional nonlinear distortion, which is of greatest probability when the loudspeaker is behaving nearly linearly. In this case the loudspeaker input signal would be filtered, and only inputs that are likely to result in distortion above the limit would be pre-processed by the linearisation scheme. However, this would also result in considerable additional computational load.

4.5 METHODS TO IMPROVE ANN PERFORMANCE

Modelling the full frequency range of the loudspeaker with one ANN may result in over generalisation, an alternative would therefore be to split the frequency range into smaller segments and train an ANN for each frequency sub set. This would result in a more specific, less generalised set of models, however the complexity of the linearisation algorithm would be significantly increased with the various filtering operations required to direct the input signal to the correct ANN for processing and also in collecting data for subsequent model updates.

4.6 CAR INTERIOR ACOUSTICS

Chang et al (1994) used a multi layer perceptron in conjunction with a tapped delay line to model the combined transfer function of loudspeaker and room acoustics, which in theory would also compensate for nonlinearities resulting from room acoustics, however the measurement of such effects would be ambiguous due to the significant variation in the acoustics throughout the listening space. However, the car interior is unique in that the position of the loudspeakers and the listeners is fixed, hence the acoustics of the interior can be determined accurately by the system designer. The ANN model could therefore be trained to optimise the sound image in a limited area of the car interior, where the acoustics are relatively consistent, this is normally the driver's position, with secondary attention paid to the front passenger and finally an overall good image is desirable throughout the whole car.

5 MODEL OPTIMISATION

Two ANN structures were considered over the course of this investigation, the Multi Layer Perceptron feedforward network and the modified Elman recurrent network. Appendices 3 and 5 contain the C++ code used throughout this investigation for the Multi layer perceptron and the modified Elman network respectively. Each of the network parameters, along with algorithm modifications, were investigated in order to determine the configurations where correlation between model output and actual loudspeaker output was optimised.

5.1 MULTI LAYER PERCEPTRON

5.1.1 Default parameter values

In order to examine how a parameter affected the rms error value of the multi layer perceptron, all other parameters were set to default values as outlined in table 5.1 and the parameter under investigation was systematically altered. The rms error value was calculated for the data used in the training set and the set of weights that returned the lowest training rms error was saved, in order to be tested with a set of previously unseen data in order to generate a validation rms error. Appendix 4 contains the C++ code for the validation program.

Table 5.1 – Default parameter values for the multi layer perceptron

Parameter	Default value
Topology	9-10-5-1
Epoch number	1×10^4
Training set size	2.56×10^4
Validation set size	2.56×10^4
Learning rate	1×10^{-5}
Momentum	5×10^{-2}
Input file format	in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2}, out_{t-3}, in_{t-3}, out_{t-4}, in_{t-4}
Sampling frequency	44.1 kHz

5.1.2 Topology

The number of neurons in each hidden layer can affect the performance and modelling capability of the network. Tests were performed with the number of neurons in the first hidden layer varied between 10 and 25 and those in the second hidden layer kept constant at 5. The number of neurons in the first hidden layer was then kept constant at 10 and those in the second hidden layer varied between 2 and 35. Two further networks were trained with 50 and 100 neurons in the first hidden layer and 20 and 50 neurons in the second hidden layer respectively, in order to gain insight into the advantage of using significantly larger network architectures.

The results in tables 5.2, 5.3 and 5.4 show that there is no direct relationship between the number of neurons in the hidden layers and the rms error values. The ANN weights are adjusted according to the error generated during the forward pass, and due to the parallelism of the network the error value will not be directly related to the number of neurons in the hidden layers, hence neither will the final value of the weights and therefore the rms error value. Table 5.4 further illustrates that there is little or no gain in employing significantly larger architectures for this application, especially when the substantially increased training period is considered. The larger network architectures may not significantly improve the model performance as there is an appreciably higher possibility of instability in such networks, due to each neuron's output dependence on a larger number of terms i.e. the outputs of the previous layer's neurons.

The results of this investigation confirm the assertion by Miller, (1999) that the performance of the ANN model is subject to the suitability of the network architecture to model that particular system.

Table 5.2 – Effect of number of neurons in 1st hidden layer on rms error value

Number of neurons in 1st hidden layer	Training rms error	Validation rms error
10	0.0162	0.0564
12	**0.0168**	**0.0357**
14	0.0172	0.0360
16	0.0201	0.0421
18	0.0194	0.0410
20	0.0187	0.0396
25	0.0162	0.0444

Table 5.3 – Effect of number of neurons in 2nd hidden layer on rms error value

Number of neurons in 2nd hidden layer	Training rms error	Validation rms error
2	0.0368	0.0619
3	0.0253	0.0406
4*	0.0683	0.114
5	0.0192	0.0564
6	0.0209	0.0500
7	**0.0221**	**0.0404**
8	0.0249	0.0537
15	0.0276	0.0742
20	0.0445	0.0955
25	0.0261	0.0551
30	0.0231	0.0525
35	0.0309	0.0707

*The training reached a local minimum that could not be overcome, hence the significantly higher rms error values.

Table 5.4 – Effect of employing larger topologies on rms error value

Number of neurons in 1st hidden layer	Number of neurons in 2nd hidden layer	Training rms error	Validation rms error
50	20	0.0162	0.0443
100	**50**	**0.0100**	**0.0363**

In these, and all subsequent results tables, the best performing ANN model is highlighted in bold text. The lowest validation rms error achieved overall, and hence the best performing ANN in terms of generalisation was the ANN with 12 neurons in the first hidden layer and 5 in the second. All ANN models responded well to low amplitude input data, as the loudspeaker transfer function approaches linearity in this region, however with higher amplitude input the loudspeaker response was more nonlinear and thus it was in this region that the models' performances could be discriminated. The validation data set therefore contained only high amplitude data. The response of the best performing ANN model to larger amplitude inputs is illustrated with the actual loudspeaker output in figure 5.1. The high amplitude segments of the signal did not occur concurrently, hence the disjointed appearance of the signal.

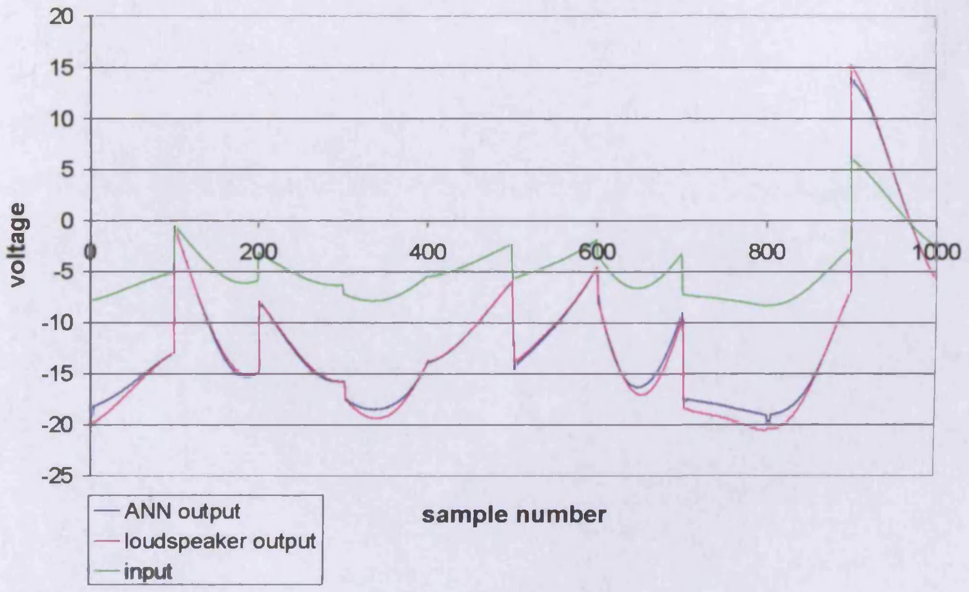


Figure 5.1 – Response of multi layer perceptron model with 12 neurons in the first hidden layer and 5 in the second to large amplitude inputs

5.1.3 Training epochs

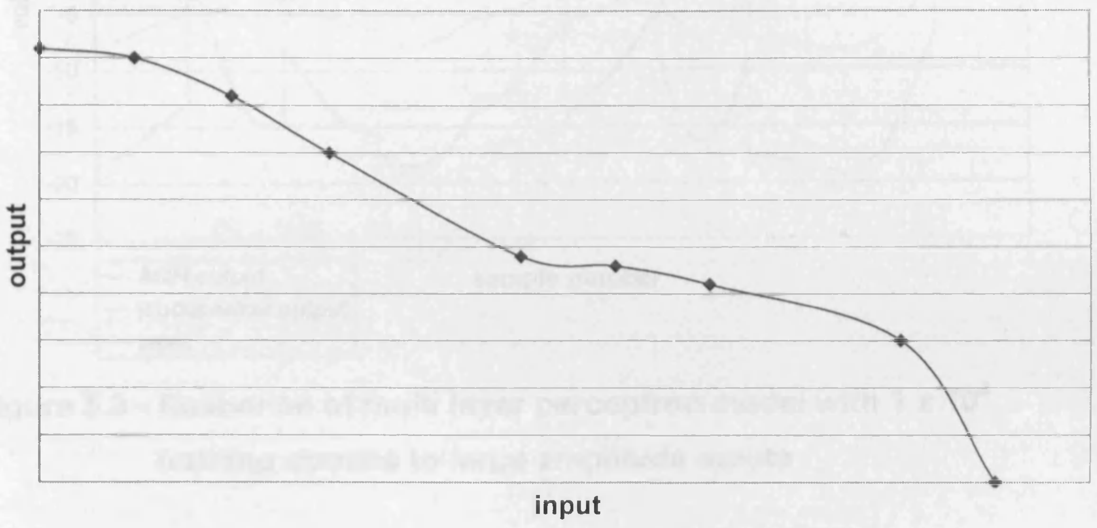
Increasing the number of training epochs involves the use of each data sample an increasing number of times in the training process, thus improving the chance that the ANN will learn the association between the input and output of that sample and that the resultant model will be a good representation of the actual transfer function. Table 5.5 shows that although the rms error value calculated for the training set consistently decreases as epoch number increases, the validation data rms error begins to increase again after 1 x 10^4 epochs. This is due to a phenomenon known as overfitting.

Table 5.5 – Effect of epoch number on rms error value

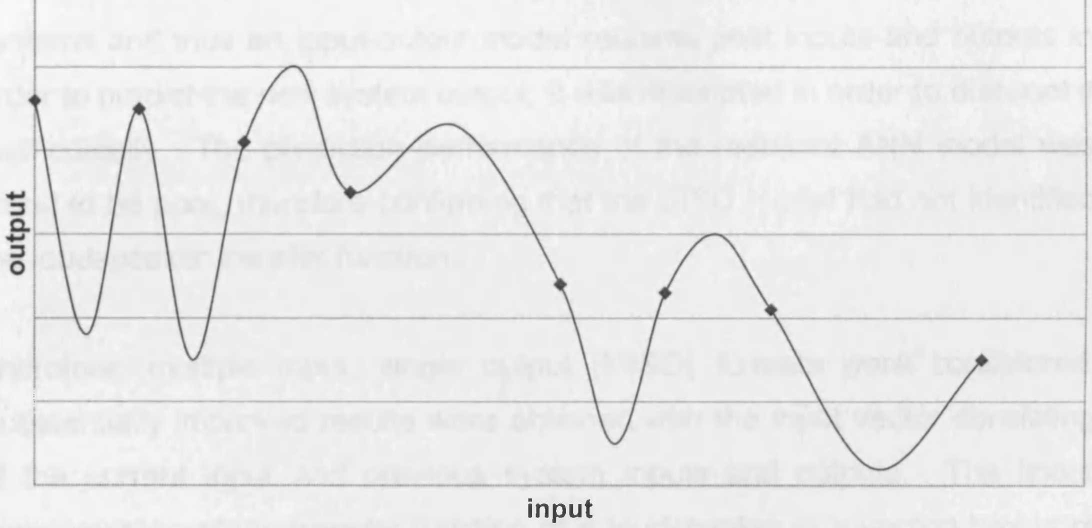
Epoch number	Training rms error	Validation rms error
1 x 10^3	0.0759	0.133
2 x 10^3	0.0450	0.0776
5 x 10^3	0.0284	0.0580
1 x 10^4	**0.0192**	**0.0564**
2 x 10^4	0.0164	0.0597
5 x 10^4	0.0135	0.0617

The objective of the back propagation algorithm is to minimise the error function of the training set, thus facilitating the reproduction of the training data as closely as possible. However, this can actually be at the cost of accurate generalisation, as the ANN weights are adjusted to model just the training data, and not the underlying transfer function, thus overtraining or overfitting the training data, which results in a poor performance from the model when predicting the output of previously unseen data. Overtraining is illustrated in a simplified manner in figure 5.2, where the training samples (red) are the same in both charts, and the underlying functions (blue) would return the correct values for the training data in both cases, however, the ANN subject to overfitting would not return the correct values with previously unseen data samples.

It is therefore unbeneficial to train the ANN further once overfitting has commenced, which in this instance, indicated by an increase in the validation rms error, was between 1 x 10^4 and 2 x 10^4 epochs. Figure 5.3 shows the response of the best performing ANN model, that which had 1 x 10^4 training epochs, to larger amplitude inputs.



Properly fitted data (good generalisation)



Overfitted data (poor generalisation)

Figure 5.2 – Illustration of properly fitted and overfitted data (Haykin, 1999)

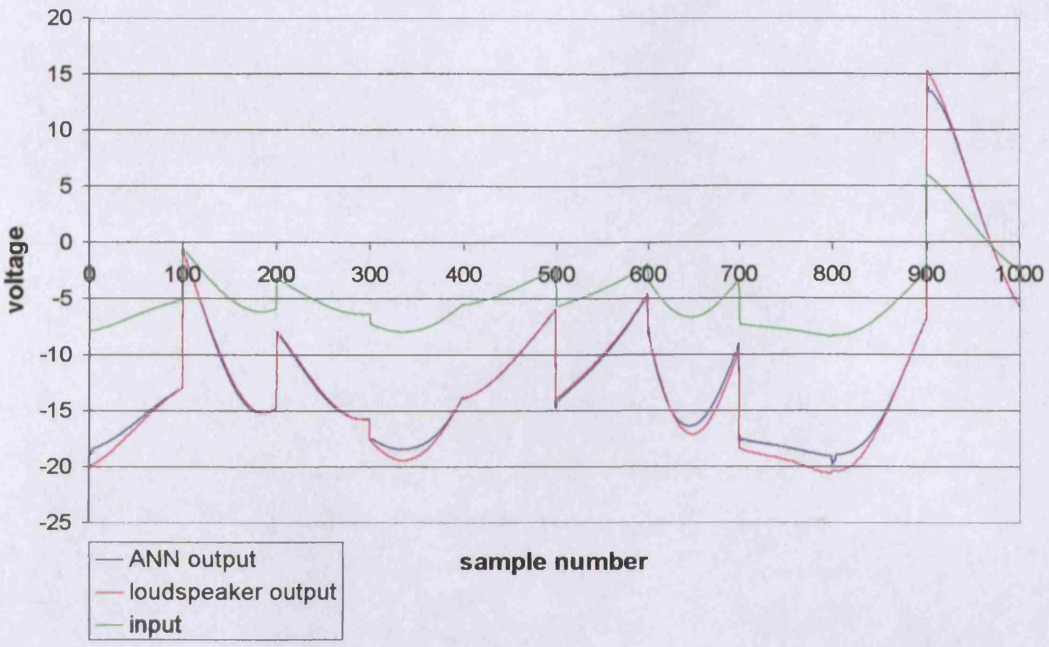


Figure 5.3 - Response of multi layer perceptron model with 1 x 10^4 training epochs to large amplitude inputs

5.1.4 Training data formatting

Although it was anticipated that a single input, single output (SISO) format model would be insufficient for this application, as loudspeakers are dynamic systems and thus an input-output model requires past inputs and outputs in order to predict the new system output, it was attempted in order to discount it methodically. The prediction performance of the resultant ANN model was found to be poor, therefore confirming that the SISO model had not identified the loudspeaker transfer function.

Therefore, multiple input, single output (MISO) formats were considered. Substantially improved results were obtained with the input vector consisting of the current input and previous system inputs and outputs. The linear approximation of the transfer function of a loudspeaker in a vented box is of fourth order, (Thiele,1961, Small,1973), the ANN would therefore theoretically require an input/output history of four previous time steps to model the transfer function (Pham, Liu, 1995). However, the nonlinearities in the system may result in a higher order transfer function, hence an investigation was performed to determine the optimum number of previous time steps in the ANN training set. The results are shown in table 5.6.

Table 5.6 – Effect of number of previous time steps in training data on rms error

Previous time steps	Data format	Training rms error	Validation rms error
0	in_t	0.0458	0.0490
1	in_t, out_{t-1}, in_{t-1}	0.0104	0.0283
2	in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2}	0.0135	0.0351
3	**in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2}, out_{t-3}, in_{t-3}**	**0.0114**	**0.0271**
4	in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2},... out_{t-4}, in_{t-4}	0.0192	0.0564
5*	in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2},... out_{t-5}, in_{t-5}	0.0165	0.0405
6*	in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2},... out_{t-6}, in_{t-6}	0.0168	0.0413
20*	in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2},... out_{t-20}, in_{t-20}	0.0434	0.0819

*The default topology resulted in local minima as it is generally necessary to have a larger number of neurons in the first hidden layer than in the input layer, hence the topology used in these cases had 50 neurons in the first hidden layer and 20 in the second. As discussed in section 5.1.2 this should not significantly affect the resultant rms errors.

The best performing ANN was trained with an input vector containing 3 previous inputs and outputs. This was a smaller vector than was expected and may be due to the low nonlinear content of the training data (see section 5.4)

Alternative formats were also investigated to determine the optimum configuration for inputting the data to the ANN, the results are shown in table 5.7.

The training data format resulting in the best performing ANN model was that of set d. The ANN model output to large amplitude input signals is shown in figure 5.4.

Table 5.7 – Effect of training data format on rms error

Data set	Data format	Training rms error	Validation rms error
a	in_t, out_{t-1}, out_{t-2},... out_{t-4}	0.0253	0.0489
b	in_t, out_{t-1}, in_{t-1}, out_{t-2}, in_{t-2},... out_{t-4}, in_{t-4}	0.0192	0.0564
c	in_t, in_{t-1}, out_{t-1}, in_{t-2}, out_{t-2},... in_{t-4}, out_{t-4},	0.0202	0.0463
d	**in_t, out_{t-1}, out_{t-2},... out_{t-4}, in_{t-1}, in_{t-2},... in_{t-4}**	**0.0198**	**0.0454**
e	in_t, in_{t-1}, in_{t-2},... in_{t-4}, out_{t-1}, out_{t-2},... out_{t-4}	0.0150	0.0520

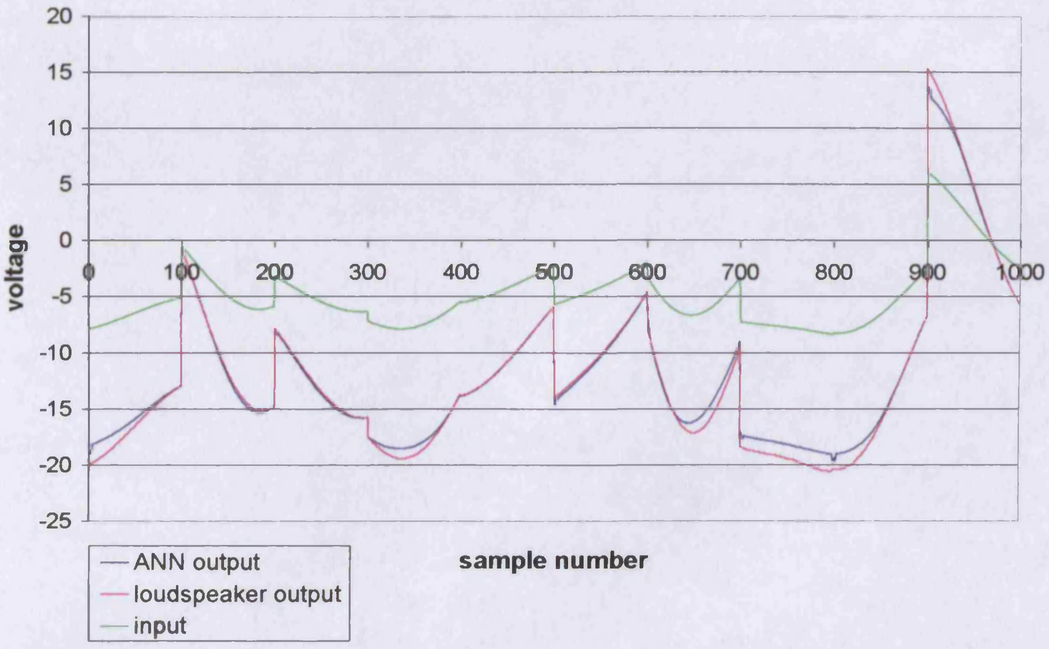


Figure 5.4 - Response of multi layer perceptron model with data format d to large amplitude inputs

5.1.5 Momentum term

The momentum term determines the degree to which the previous iteration influences the change in the weights in the current iteration, and can significantly decrease the convergence time of the back propagation algorithm. However, it may also cause instability in the training process if it is too large. The results in table 5.8 suggest that the momentum term had little effect upon the outcome of the training session. The results also suggest that increasing the momentum value slightly decreases the training rms error, yet actually increases the validation rms error, this could be due to the increased momentum term leading to faster onset of the overfitting phenomenon described in section 5.1.3. A high momentum term may also be detrimental to the training process as it is more suited to a linear system, given that it assumes a direct relationship between the system behaviour in the previous time step and that in the current time step, which is not necessarily the case for nonlinear systems (Miller, 1999). The best performing ANN in the tests conducted had a momentum value of 1 x 10^{-3} and its output to large amplitude input signals is shown in figure 5.5.

Table 5.8 – Effect of momentum value on rms error

Momentum	Training rms error	Validation rms error
0	0.0192	0.0553
5 x 10^{-4}	0.0192	0.0553
1 x 10^{-3}	**0.0192**	**0.0553**
5 x 10^{-3}	0.0192	0.0554
1 x 10^{-2}	0.0192	0.0555
5 x 10^{-2}	0.0192	0.0564
1 x 10^{-1}	0.0191	0.0569
5 x 10^{-1}	0.0170	0.0599

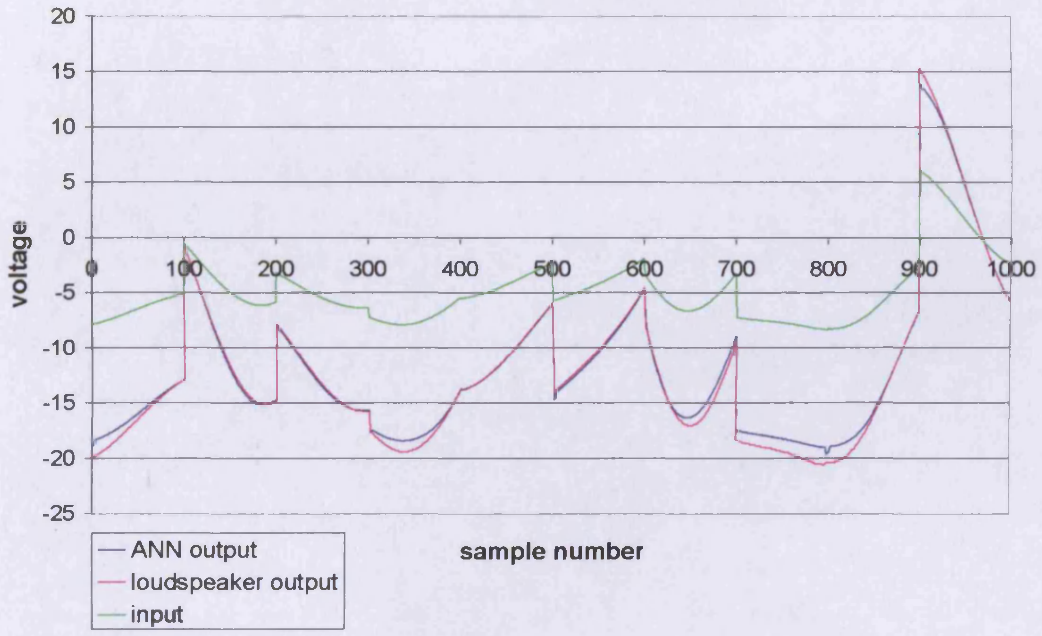


Figure 5.5 – Response of multi layer perceptron model with momentum value 1 x 10^{-3}

5.1.6 Learning rate

The learning rate determines the influence of the error generated during the forward pass on the weight alteration of the backward pass. Tests were conducted where the learning rate was varied between 1 x 10^{-6} and 1 x 10^{-1}, the results are shown in table 5.9.

Table 5.9 – Effect of learning rate on rms error

Learning rate	Training rms error	Validation rms error
1 x 10^{-1}	-*	-*
1 x 10^{-2}	-*	-*
1 x 10^{-3}	**0.0125*^{2}**	**0.0258*^{2}**
1 x 10^{-4}	0.0186*^{2}	0.0648*^{2}
1 x 10^{-5}	0.0192	0.0564
1 x 10^{-6}	0.0189	0.0391
1 x 10^{-7}	0.0827	0.139

*The training process crashed.

*^{2}The training process reached a local minimum.

Theoretically, the lower the learning rate, the longer the convergence time of the ANN. However, too high a learning rate may result in instability, where the ANN weight values oscillates instead of converging or even diverge, as in the first two instances in table 5.9, causing the training process to crash. This was due to the incompatibility of the learning rate and the initial values of the weights – the high learning rate caused considerable changes in the weight values, which subsequently generated a sufficiently high error value to crash the program. Despite achieving the lowest rms error of the investigation, the ANN with a learning rate of 1 x 10^{-3} reached a local minimum that it could not escape within the training period (the algorithm incorporates the capability to reduce the learning rate if the rms error does not decrease over a predetermined number of iterations, which should aid the training process in the escape of local minima), as did the ANN with a learning rate of 1 x 10^{-4}. The relatively low validation rms error achieved with a learning rate of 1 x 10^{-6} may be a result of the lower learning rate facilitating the escape of the training process from the local minimum on the error surface that the training processes with higher learning rates could not.

The best performing ANN model is shown in figure 5.6.

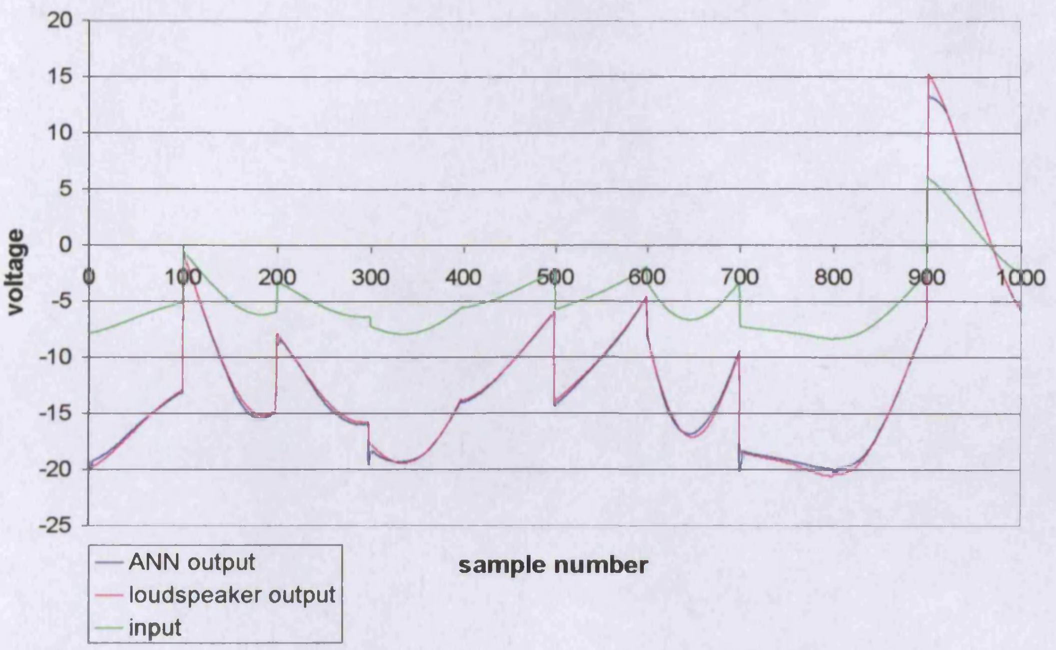


Figure 5.6 – Response of multi layer perceptron model with learning rate 1×10^{-3}

Although this network achieved the lowest validation rms error, it did in fact become trapped in a local minimum. Therefore, 1×10^{-3} was not considered the optimum value for the learning rate; the next best result, 1×10^{-6}, was sufficiently lower to reduce the possibility of the weights becoming trapped in a local minimum and not too low to adversely affect the rate of convergence.

5.1.7 Training set size

The larger the training set size, the more representative of the loudspeaker behaviour it is likely to be. A larger training set should contain data pertaining to a wider range of excitation amplitudes and frequencies than a smaller set. The results of the investigation into the effect of training set size, where the number of training samples in the set was varied between 1×10^3 and 5×10^4, are shown in table 5.10.

Table 5.10 – Effect of training set size on rms error

Number of data samples in training set	Training rms error	Validation rms error
1×10^3	0.0709	0.176
2×10^3	0.0698	0.148
5×10^3	0.0298	0.104
1×10^4	0.0205	0.0811
2×10^4	0.0189	0.0576
5×10^4	**0.0155**	**0.0167**

It can be seen that there is a clear relationship between training set size and rms error; the greater the number of data samples in the training set the lower the training and validation rms error. It was ascertained from figure 5.7 that the relationship is in fact logarithmic.

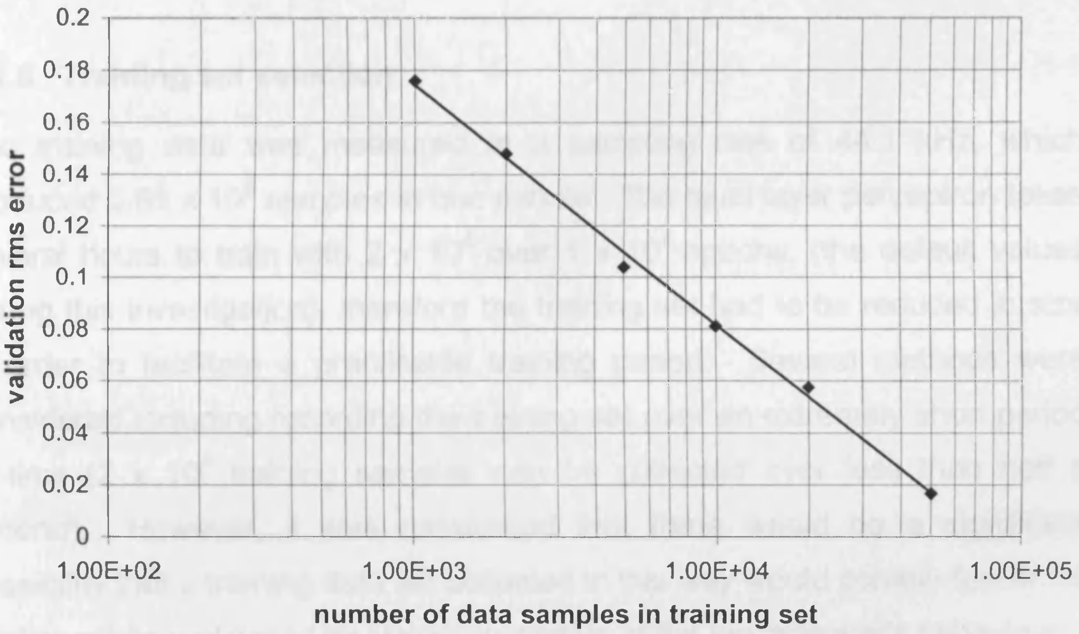


Figure 5.7 – Logarithmic plot of validation rms error versus training set size

Figure 5.8 shows the best performing ANN model from this set of experiments.

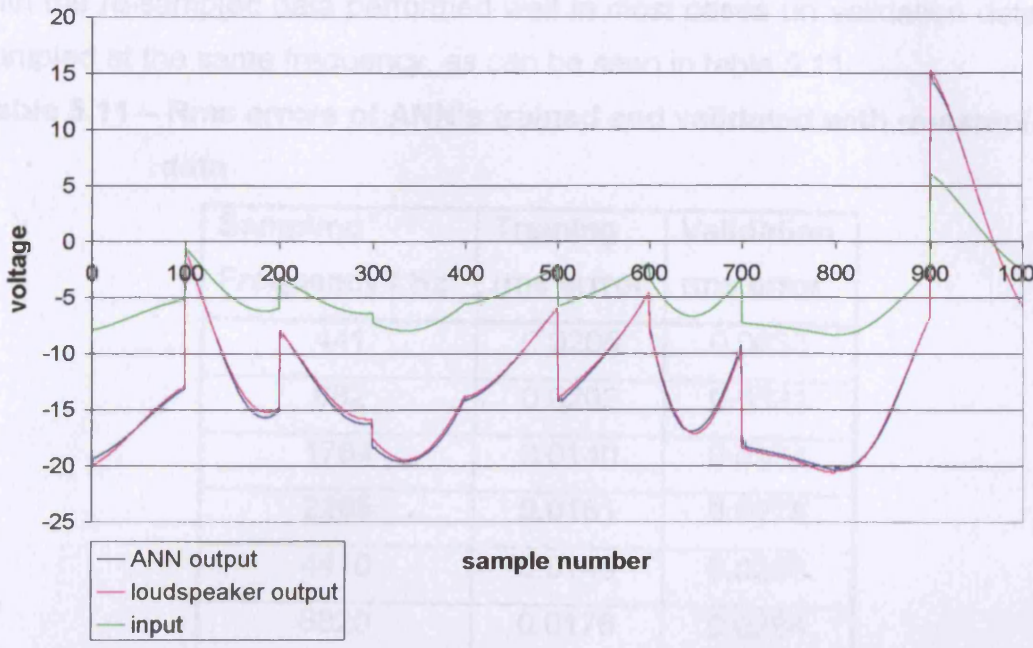


Figure 5.8 - Response of multi layer perceptron model with 5 x 10^4 training samples

5.1.8 Training set selection

The training data was measured at a sampling rate of 44.1 kHz, which produced 2.65 x 10^6 samples in one minute. The multi layer perceptron takes several hours to train with 2 x 10^4 over 1 x 10^4 epochs, (the default values during this investigation), therefore the training set had to be reduced in size in order to facilitate a practicable training period. Several methods were considered including recording the training set over an extremely short period of time (2 x 10^4 training samples can be collected over less than half a second). However, it was considered that there would be a significant possibility that a training data set collected in this way would contain few or no nonlinearities and would be unrepresentative of the loudspeaker's behaviour.

5.1.8.i Signal re-sampling

Experiments were conducted where the reduction of the training set size was achieved through re-sampling. The original data set was re-sampled to lower frequencies with a simple piece of C++ code (See Appendix 6). ANNs trained with the re-sampled data performed well in most cases on validation data re-sampled at the same frequency, as can be seen in table 5.11.

Table 5.11 – Rms errors of ANN's trained and validated with re-sampled data

Sampling Frequency / Hz	Training rms errror	Validation rms error
441	0.0205	0.0837
882	0.0202	0.0531
1764	0.0140	0.0377
2205	**0.0161**	**0.0278**
4410	0.0149	0.0286
8820	0.0176	0.0284
22050	0.0159	0.0767
44100	0.0205	0.0811

The response of the best performing ANN to large amplitude input data sampled at the same frequency is shown in figure 5.9.

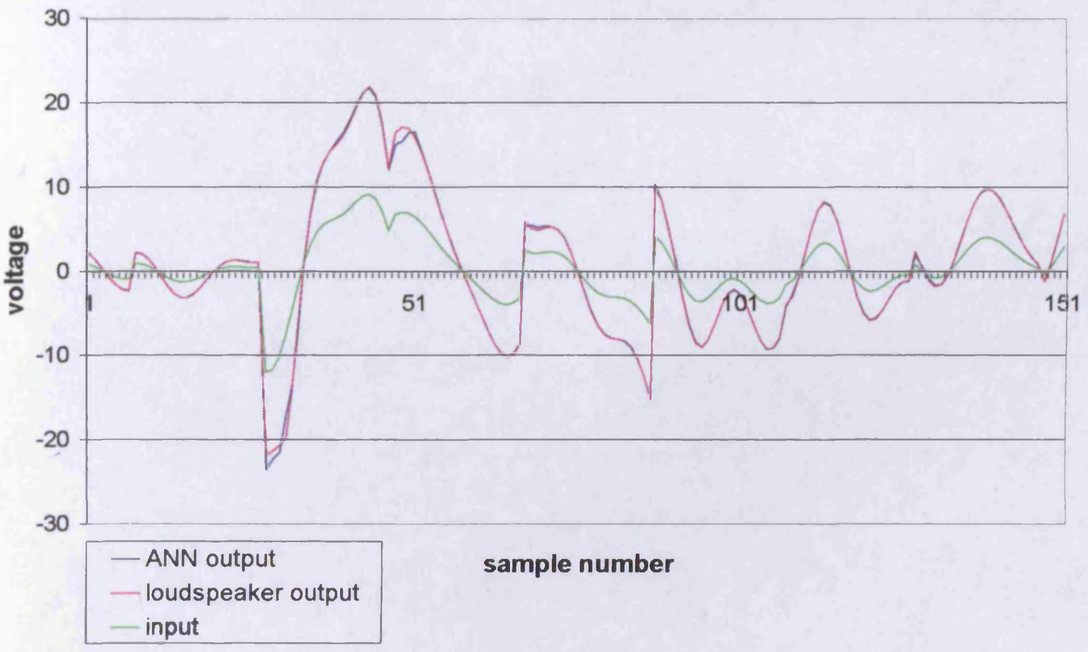


Figure 5.9 – Response of multi layer perceptron trained with data sampled at 2205 Hz

However, when a Fourier analysis was performed on sine wave inputs to the ANN model, the resultant frequency response curves were significantly different to the measured response. The low frequency response of a loudspeaker may be considered equivalent to that of a high pass filter. At higher frequencies the response eventually drops off, which in effect results in an overall frequency response homologous to that of a band pass filter. The frequency response curves obtained from the ANN models trained with data sampled at the lower frequencies (up to 8820Hz) show closer correlation to a low pass filter, as can be seen in figure 5.10. ANN models trained with data sampled above 8820Hz showed better correlation to a high pass filter, as illustrated in figure 5.11.

A possible explanation was that information was lost from the data set in the re-sampling process, hence the re-sampled data did not contain sufficient information to model the frequency response of the loudspeaker accurately. This led to an investigation into the nonlinear properties of the data. It was determined that a test for nonlinearity, such as that suggested by Kantz, Screiber, (2004) with the use of surrogate data would be extremely complex and time consuming to implement. Therefore, linear regression was applied to subsets of the training set in order to gain some indication of the relationship between the input and output data. Linear regression should only strictly be used with linear data, however, it does give a correlation coefficient, which if low could be interpreted as the absence of a direct relationship between input and output, which could in turn could be assumed to infer a nonlinear relationship.

The input/output data of the ANN models were split into further subsets and the correlation coefficient (Pearson Product Moment Correlation Coefficient) between the input and output of the subsets calculated. The number of subsets with correlation coefficients below various thresholds (0.9 to 0.4, in 0.1 intervals) was determined. The most significant conclusion of this investigation was that the nonlinear content of all of the data sets was low, including the original sampled at 44.1kHz, as no more than 10 per cent of the subsets in all cases had correlation coefficients below 0.9 (a coefficient of 1 signifying perfect linear correlation between input and output). However, the training data sets with higher sampling frequencies tended to have relatively higher nonlinear content, with the original data set with the highest. In theory, the re-sampling of the signal would result in the loss of the nonlinearities if their frequency were above the Nyquist frequency, which at very low sampling frequencies may well be the case, as many nonlinearities appear as harmonics of the fundamental frequency.

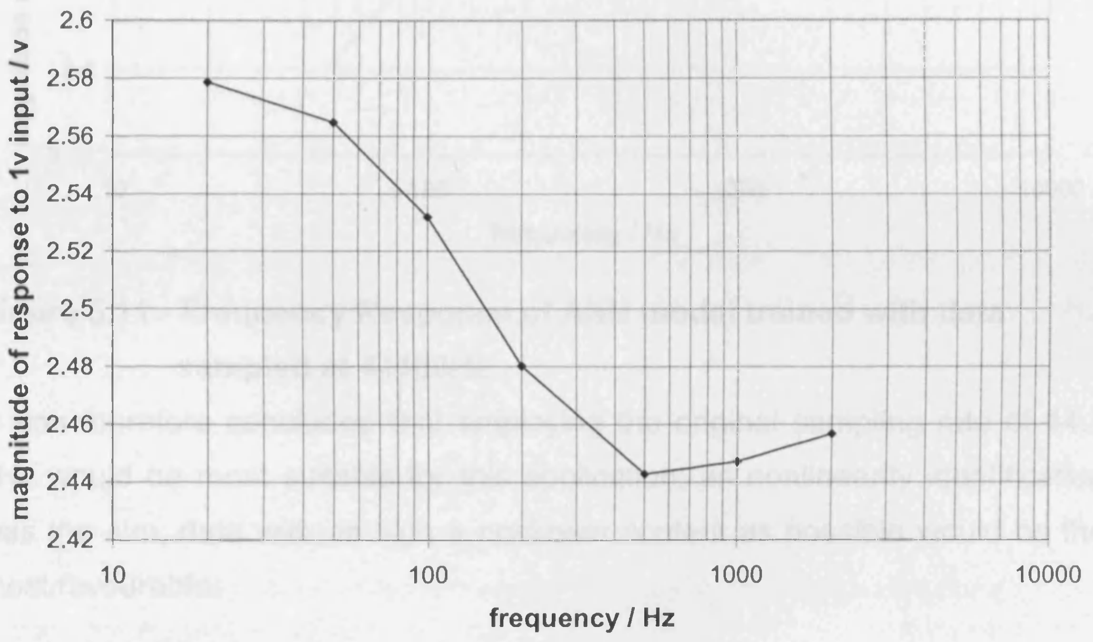


Figure 5.10 – Frequency Response of ANN model trained with data sampled at 8820Hz

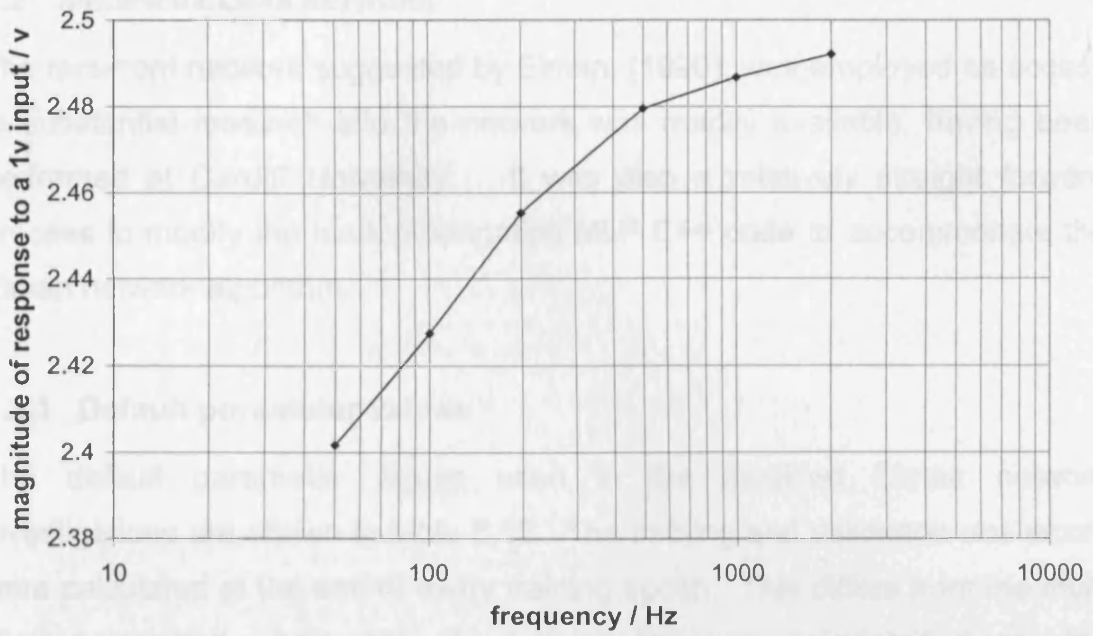


Figure 5.11 - Frequency Response of ANN model trained with data sampled at 44100Hz

It was therefore concluded that employing the original sampling rate of 44.1 kHz would be most suitable for this application; as nonlinearity identification was the aim, data with as high a nonlinear content as possible would be the most favourable.

5.1.8.ii Selection of subsets

Another possibility was to collect data over a longer period of time and use subsets of this large data set to train the ANN. This would improve the probability that the training data would contain nonlinearities and also a broader range of excitation amplitudes and frequencies, thus resulting in a more representative ANN model. This was the method employed throughout all other investigations.

5.1.8.iii Training epoch reduction

A further alternative to reduce training time was to use a large volume of data samples and forfeit a high number of training epochs. This is discussed in section 6.3 with respect to online training.

5.2 MODIFIED ELMAN NETWORK

The recurrent network suggested by Elman, (1990), was employed as access to substantial research into the network was readily available, having been performed at Cardiff University. It was also a relatively straight forward process to modify the back propagation MLP C++ code to accommodate the Elman network algorithm.

5.2.1 Default parameter values

The default parameter values used in the modified Elman network investigations are shown in table 5.12. The training and validation rms errors were calculated at the end of every training epoch. This differs from the multi layer perceptron where only the training rms was calculated during the training process; the validation rms was calculated after the termination of the training process, for the weights that produced the lowest training rms error. Calculating the validation rms error during the training process has the advantage that a weight configuration that is optimum for the validation data but not necessarily for the training data set can be identified.

Table 5.12 – Default parameter values for modified Elman network

Parameter	**Default value**
Topology	1-3-1 plus 3 neurons in context layer
Epoch number	5×10^2
Training set size	2.56×10^4
Validation set size	2.56×10^4
Learning rate – context layer	1×10^{-6}
Learning rate – rest of network	1×10^{-5}
Momentum	0
α value	0.7
Input file format	in_t
Sampling frequency	44.1 kHz

5.2.2 Topology

The topology of the modified Elman network was investigated by altering the number of neurons in the hidden layer and the context layer, which always corresponds to the number of neurons in the hidden layer. The topologies tested and the resultant rms errors are shown in table 5.13.

Table 5.13 – Effect of topology on rms error

Number of Neurons in hidden layer (and context layer)	Training rms error	Validation rms error
1	0.111	0.157
2	0.117	0.150
3	**0.0893**	**0.108**
4	0.131	0.176
5	0.177	0.230
6	0.219	0.280
8	0.153	0.191
10	0.264	0.320
20	0.224	0.246
30	0.247	0.277

The response of the best performing ANN model to larger amplitude inputs is shown in figure 5.12.

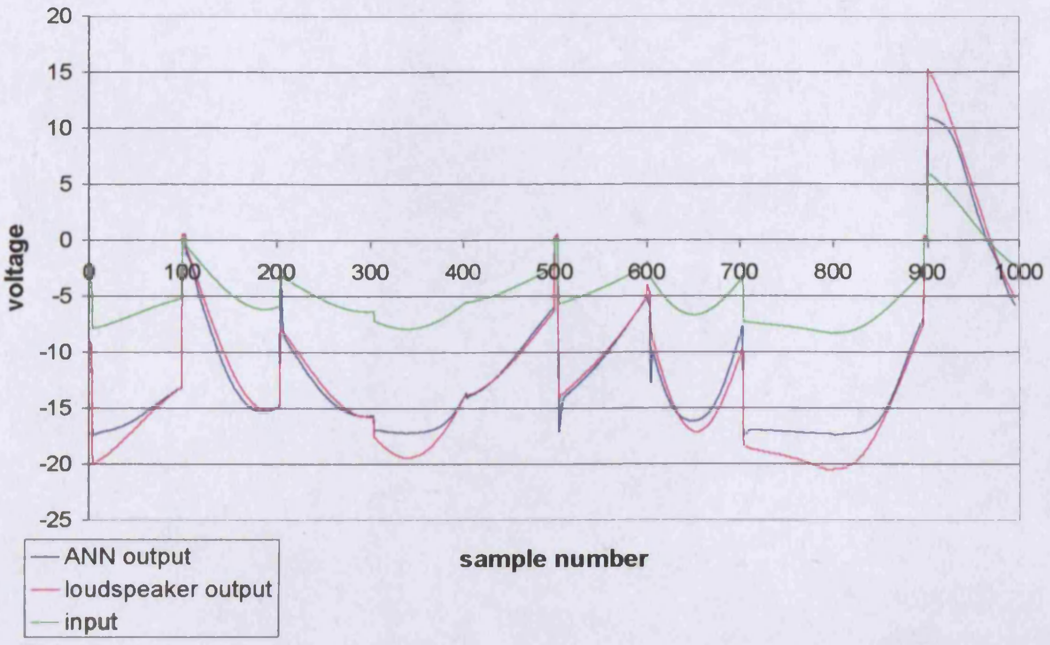


Figure 5.12 – Response of modified Elman network with 3 neurons in the hidden and context layers

Miller, (1999) proposed that the number of neurons required in the hidden and context layers increased with the order of the system being modelled up to a certain limit where increasing the number of neurons resulted in little benefit. However, in this instance the optimum number of neurons in the hidden and context layers was found to be relatively low. As with the multi layer perceptron, the optimum architecture for the modified Elman network was not the largest, but that which was most compatible with the loudspeaker system.

5.2.3 Training epochs

As previously discussed with respect to the multi layer perceptron, increasing the number of training epochs improves the probability that the ANN will learn the association between the input and output of the training data. During the experiment the number of training epochs was varied between 1 and 1 x 10^4. The resultant rms error values, as shown in table 5.14 and figure 5.13, illustrate that significant improvements in the ANN performance can be achieved by increasing the number of training epochs up until approximately 1 x 10^2 epochs, however, the gain in ANN performance beyond this point constantly decreases. Therefore, there is little benefit in extending the training session beyond 5 x 10^3 epochs, especially when the significantly prolonged training period is considered.

Table 5.14 – Effect of number of training epochs on rms error values

Number of training epochs	Training rms error	Validation rms error
1	0.705	0.724
10	0.629	0.67
1 x 10^2	0.141	0.2
2 x 10^2	0.109	0.149
3 x 10^2	0.0984	0.128
4 x 10^2	0.0929	0.116
5 x 10^2	0.0893	0.108
1 x 10^3	0.0792	0.0892
2 x 10^3	0.0666	0.0751
5 x 10^3	0.0493	0.0557
1 x 10^4	**0.0461**	**0.0502**

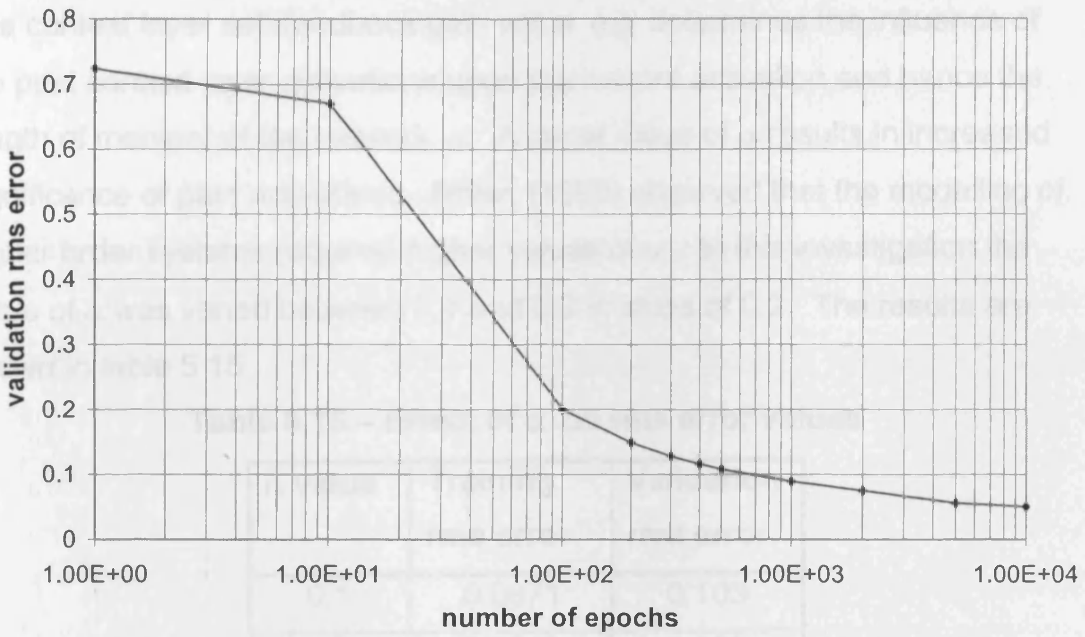


Figure 5.13 – Plot of validation rms error value versus number of training epochs

The response of the best performing ANN, trained over 1 x 10^4 epochs is shown in figure 5.14.

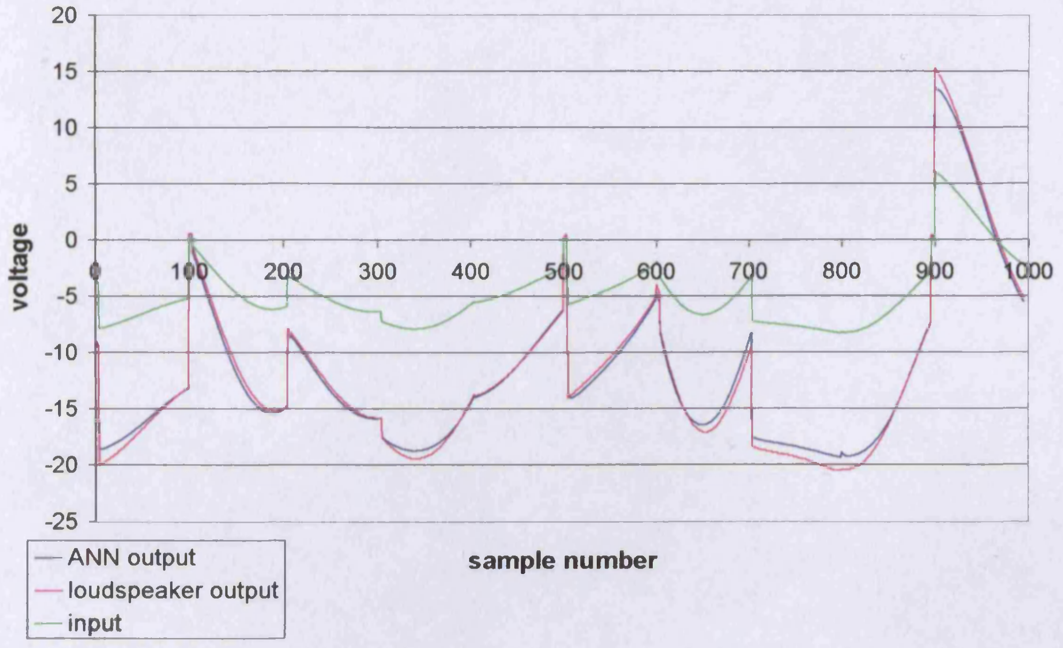


Figure 5.14 – Response of modified Elman network trained over 1 x 10^4 epochs

5.2.4 Context layer self-feedback gain value

The context layer self-feedback gain value (α) determines the influence of the past context layer activations upon the current activation and hence the length of memory of the network. α. A larger value of α results in increased significance of past activations. Miller, (1999) observed that the modelling of higher order systems required higher values of α. In this investigation the value of α was varied between 0.1 and 0.9 in steps of 0.2. The results are shown in table 5.15.

Table 5.15 – Effect of α on rms error values

α value	**Training rms error**	**Validation rms error**
0.1	0.0871	0.103
0.3	**0.0869**	**0.103**
0.5	0.0810	0.107
0.7	0.0893	0.108
0.9	0.152	0.180

An α value of 0.3 was found to produce optimum ANN performance, the response of this ANN to large amplitude inputs is shown in figure 5.15.

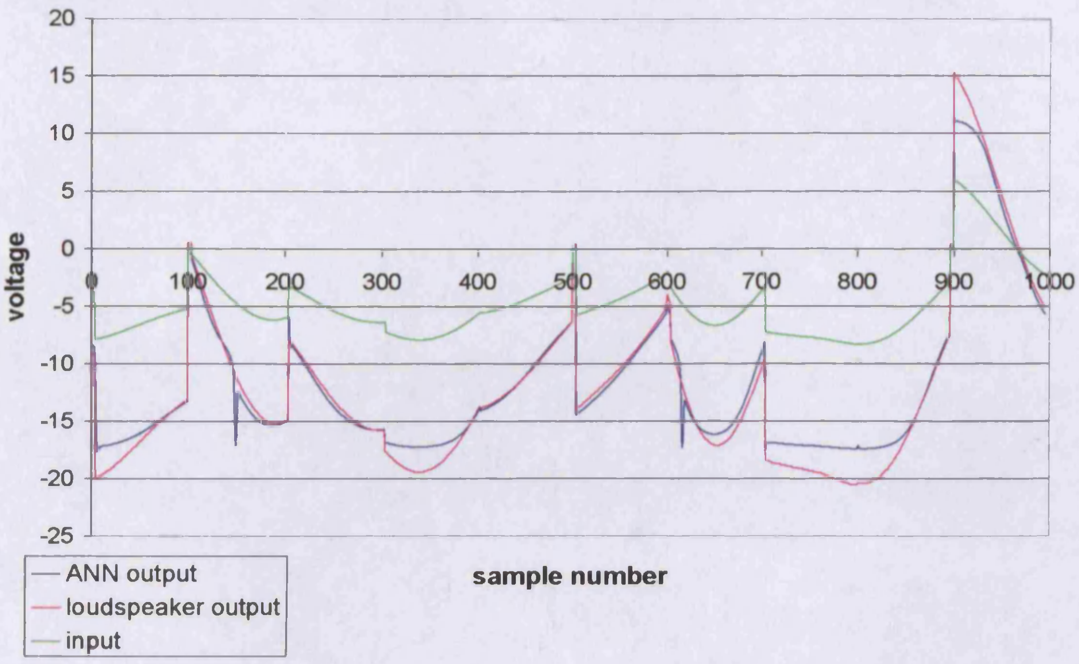


Figure 5.15 – Response of modified Elman network with α value of 0.3

5.2.5 Momentum term

The momentum term determines the degree to which the previous iteration influences the change in the weights for the current iteration in the modified Elman network, as with the multi layer perceptron. The higher probability of instability in the training process of the modified Elman network increases the significance of the momentum term and the importance of determining an optimum value that results in a stable training process. Networks were trained with momentum values that varied between 0 and 0.9. The results of the training sessions are shown in table 5.16.

As can be seen in table 5.16, the momentum does not significantly affect the minimum rms error values until it is above 1 x 10^{-1}. Instability was not experienced in any of the training sessions described in table 5.16. The response of the best performing ANN model to larger amplitude inputs is shown in figure 5.16.

Table 5.16 – Effect of momentum on rms error values

Momentum value	Training rms error	Validation rms error
0	0.0893	0.108
1×10^{-3}	0.0893	0.108
5×10^{-3}	0.0892	0.108
1×10^{-2}	0.0892	0.107
5×10^{-2}	0.0886	0.106
1×10^{-1}	0.0879	0.104
5×10^{-1}	0.0814	0.0904
7.5×10^{-1}	**0.0739**	**0.0805**
9×10^{-1}	0.0720	0.0813

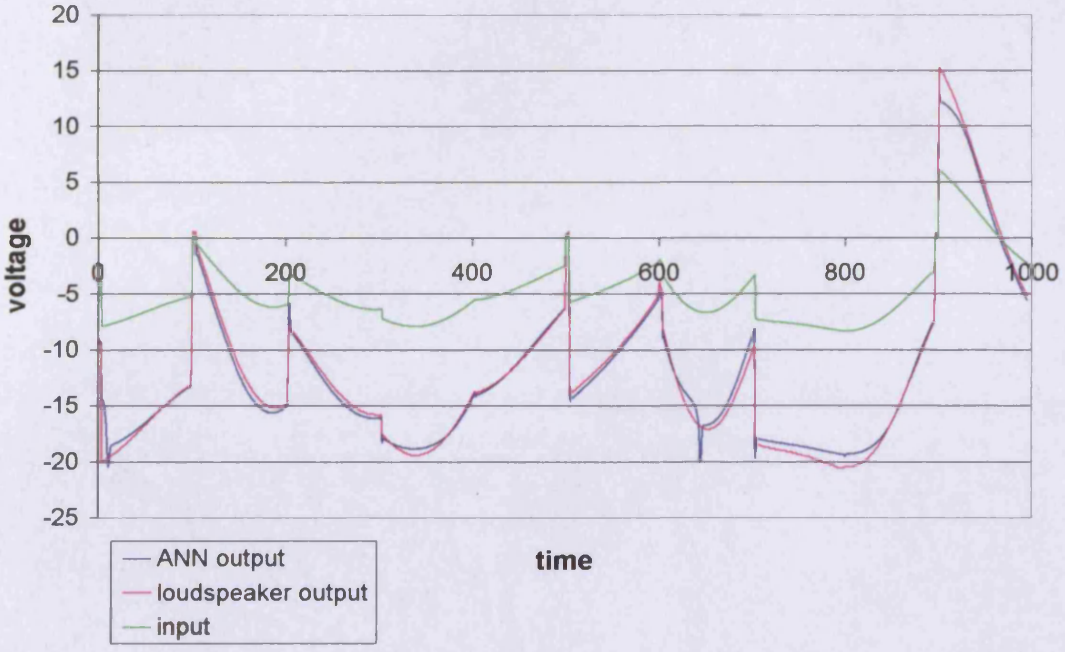


Figure 5.16 – Response of modified Elman network with momentum value of 0.75

5.2.6 Learning rate

The modified Elman network is inherently more unstable than the multi layer perceptron due to the feedback of the hidden layer's activations, hence employing suitable learning rates is of increased importance. The stability of the network can be improved by employing a lower learning rate for the weights connecting the context layer to the hidden layer, thus reducing the rate of change of the weights that control the feedback. The training process therefore approaches that of simpler, more stable, feedforward multi layer perceptron (Miller, 1999).

Experiments were conducted investigating a large number of learning rate combinations. It was determined that the stability of the training process was highly dependent upon the learning rate values and only one combination of those investigated resulted in a constantly decreasing training and validation rms error over the 500 epoch training period. Figures 5.17 and 5.18 illustrate examples of stable and unstable ANN training processes respectively.

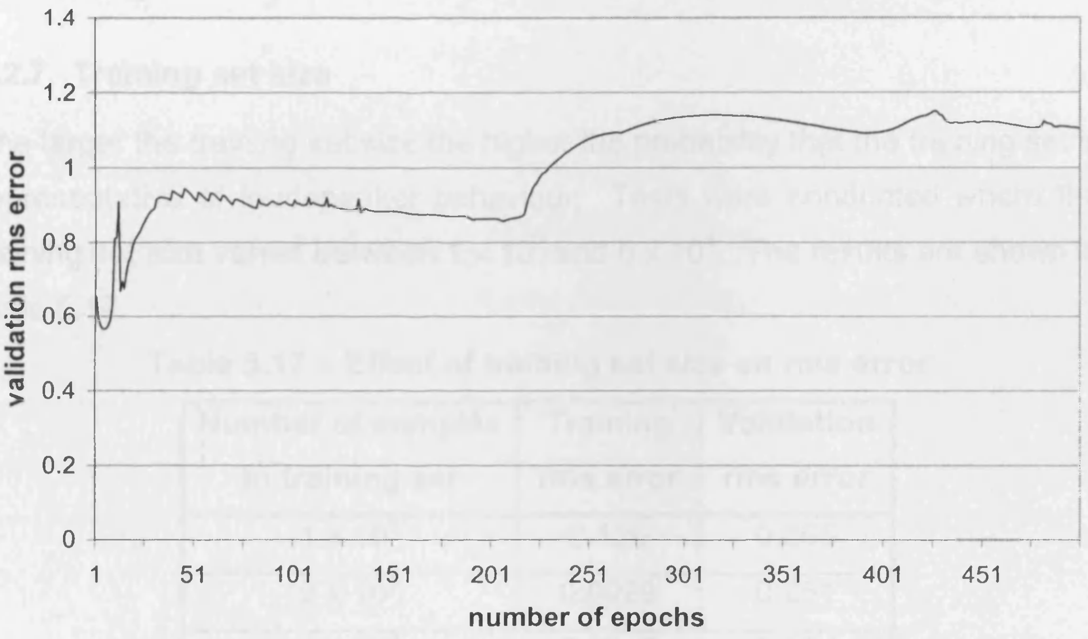


Figure 5.17 – Unstable training process

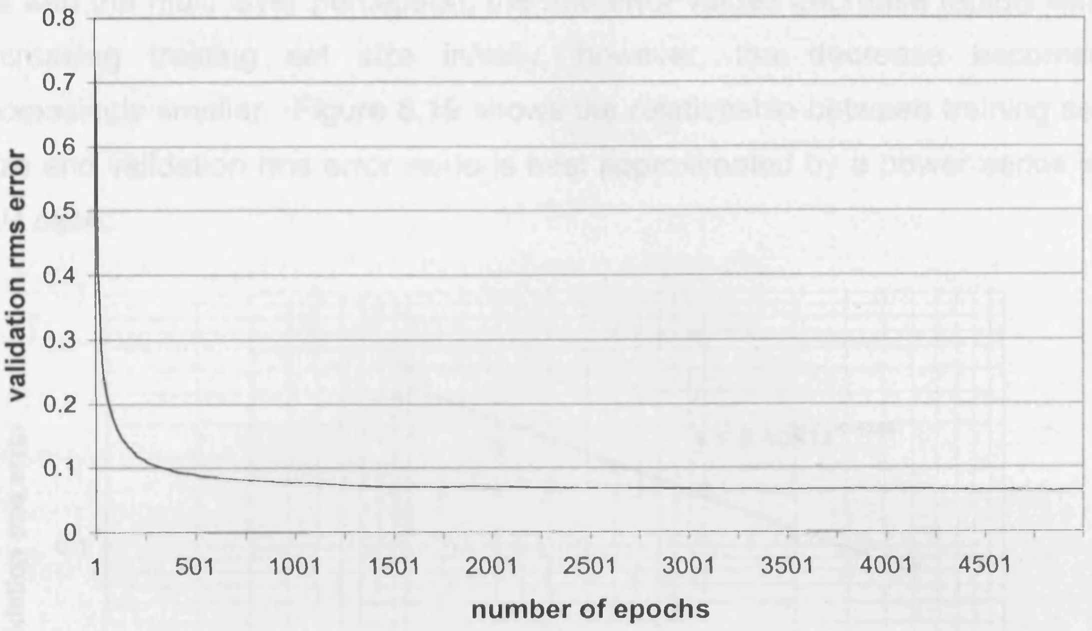


Figure 5.18 – Stable training process

The optimum values for the learning rates were therefore determined by the stability of the network rather than the resultant rms error values. The optimum learning rate for the context layer was 1 x 10^{-6} and for the rest of the network 1 x 10^{-5}.

5.2.7 Training set size

The larger the training set size the higher the probability that the training set is representative of loudspeaker behaviour. Tests were conducted where the training set size varied between 1 x 10^3 and 5 x 10^4. The results are shown in table 5.17.

Table 5.17 – Effect of training set size on rms error

Number of samples in training set	Training rms error	Validation rms error
1 x 10^3	0.126	0.505
2 x 10^3	0.0929	0.251
5 x 10^3	0.0847	0.204
1 x 10^4	0.806	0.174
2 x 10^4	0.0898	0.108
5 x 10^4	**0.0764**	**0.0815**

As with the multi layer perceptron, the rms error values decrease rapidly with increasing training set size initially, however, the decrease becomes increasingly smaller. Figure 5.19 shows the relationship between training set size and validation rms error value is best approximated by a power series in this case.

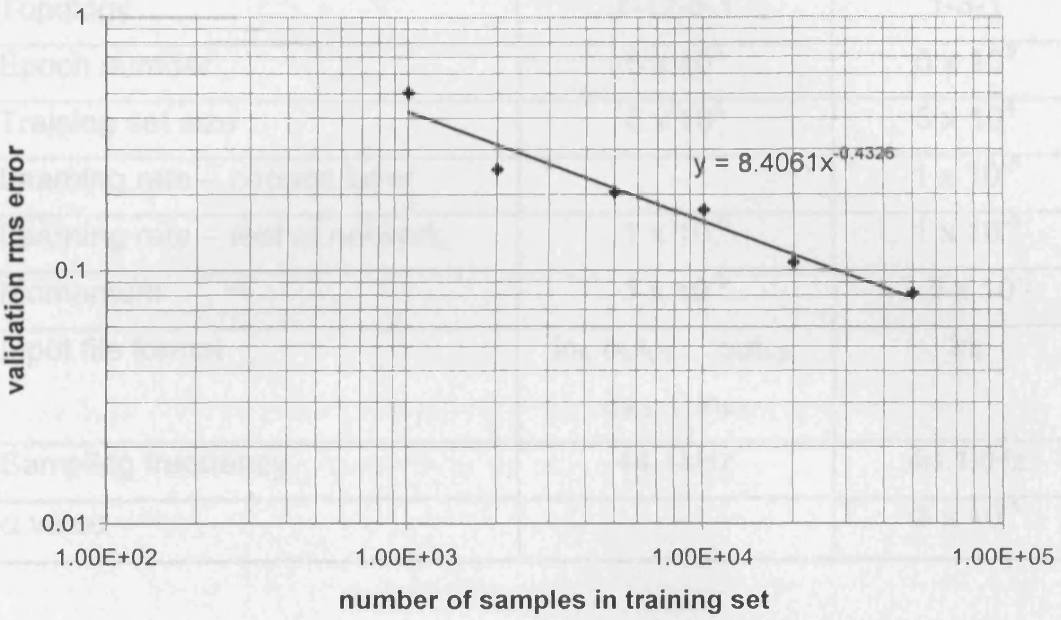


Figure 5.19 – Double logarithmic plot of validation rms error versus number of samples in training set

5.3 OPTIMUM CONFIGURATIONS

Networks were trained with the optimum parameter values determined in the preceding sections, which are outlined for each architecture in table 5.18. The optimum values for training set size and number of epochs were determined to be those where the gain in increasing the value further was outweighed by the increase in the training period. The resultant ANN model performances are summarised in table 5.19 and demonstrated in figures 5.20 and 5.21.

Table 5.18 – Optimum parameter values

Parameter	Optimum value for BPMLP network	Optimum value for modified Elman network
Topology	7-12-5-1	1-3-1
Epoch number	5×10^3	5×10^3
Training set size	5×10^4	5×10^4
Learning rate – context layer	-	1×10^{-6}
Learning rate – rest of network	1×10^{-6}	1×10^{-5}
Momentum	1×10^{-3}	7.5×10^{-1}
Input file format	in_t, $out_{t-1},\ldots out_{t-3}$, $in_{t-1},\ldots in_{t-3}$	in_t
Sampling frequency	44.1kHz	44.1kHz
α value	-	3×10^{-1}

Table 5.19 – ANN model performance when trained with optimum parameter values

Network architecture	Training rms error	Validation rms error
Multi layer perceptron	0.0239	0.0132
Modified Elman	0.0534	0.0591

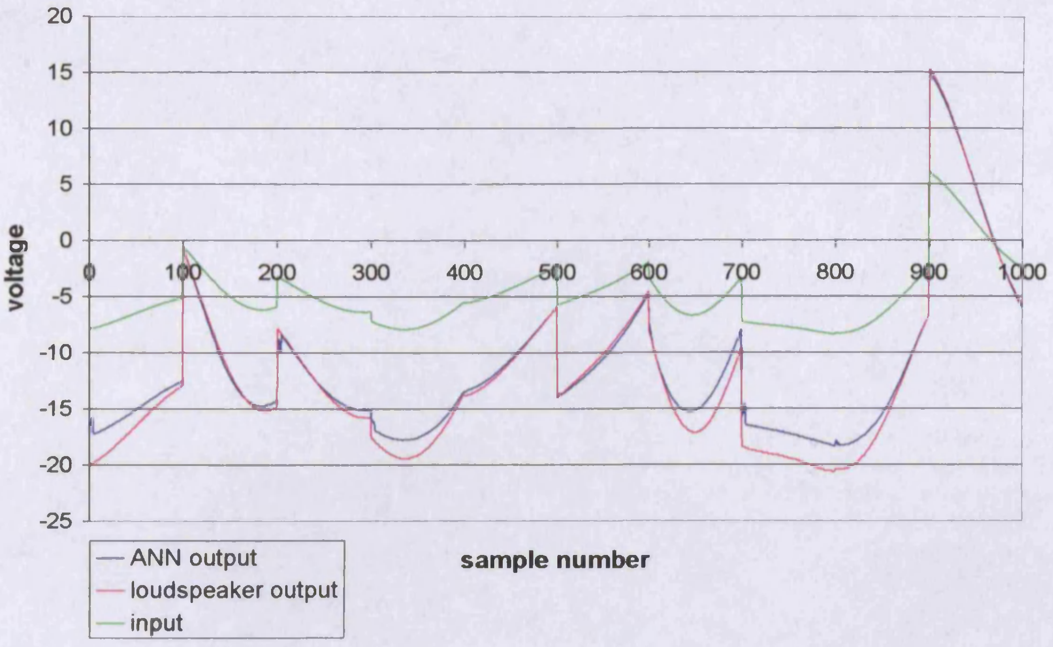


Figure 5.20 – Response of multi layer perceptron network trained with optimum parameter values

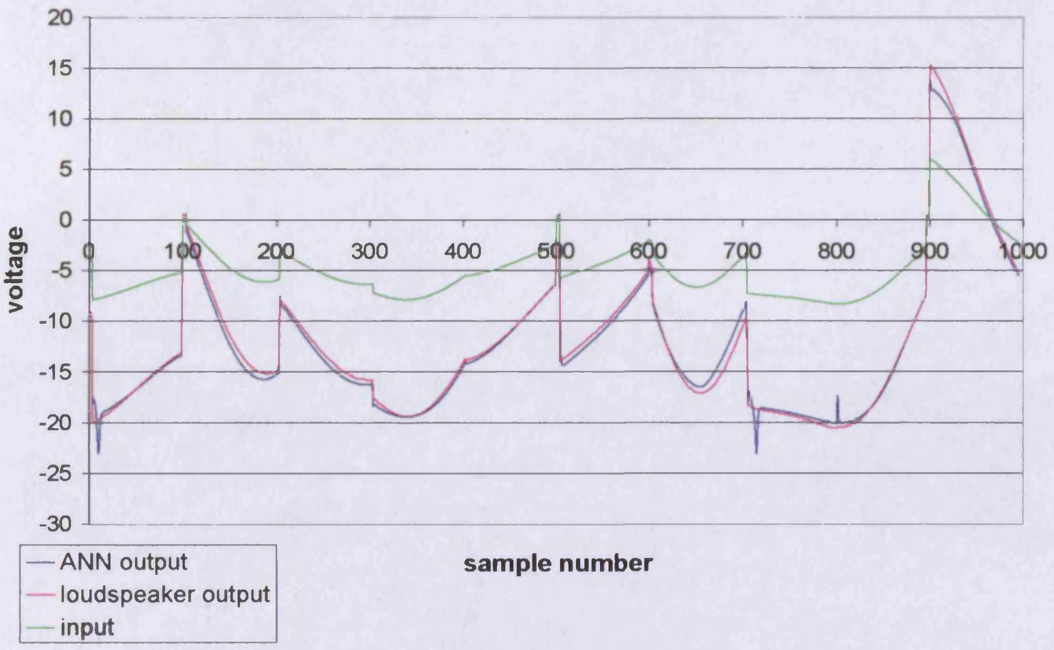


Figure 5.21 – Response of modified Elman network trained with optimum parameter values

It is likely that the irregularities displayed around time steps 0 and 700 in figure 5.21 are due to the disjointed test signal, which results in unrepresentative context unit values and thus an irregular output. This would not occur in practice when a continuous signal would be applied to the ANN.

The validation rms error for the modified Elman network was the lowest achieved thus far, as would be expected with the use of an optimum parameter configuration, however, in the case of the multi layer perceptron, lower validation rms error values were achieved with alternative configurations. The parameter configuration resulting in the lowest validation rms error achieved for the multi layer perceptron during this investigation is shown in table 5.20.

Table 5.20 – Parameter values for multi layer perceptron network with lowest achieved validation rms error

Parameter	Value
Topology	9-12-5-1
Epoch number	5×10^3
Training set size	5.12×10^4
Learning rate	1×10^{-5}
Momentum	1×10^{-3}
Input file format	in_t, out_{t-1}, in_{t-1}, … out_{t-4}, in_{t-4}
Sampling frequency	44.1kHz

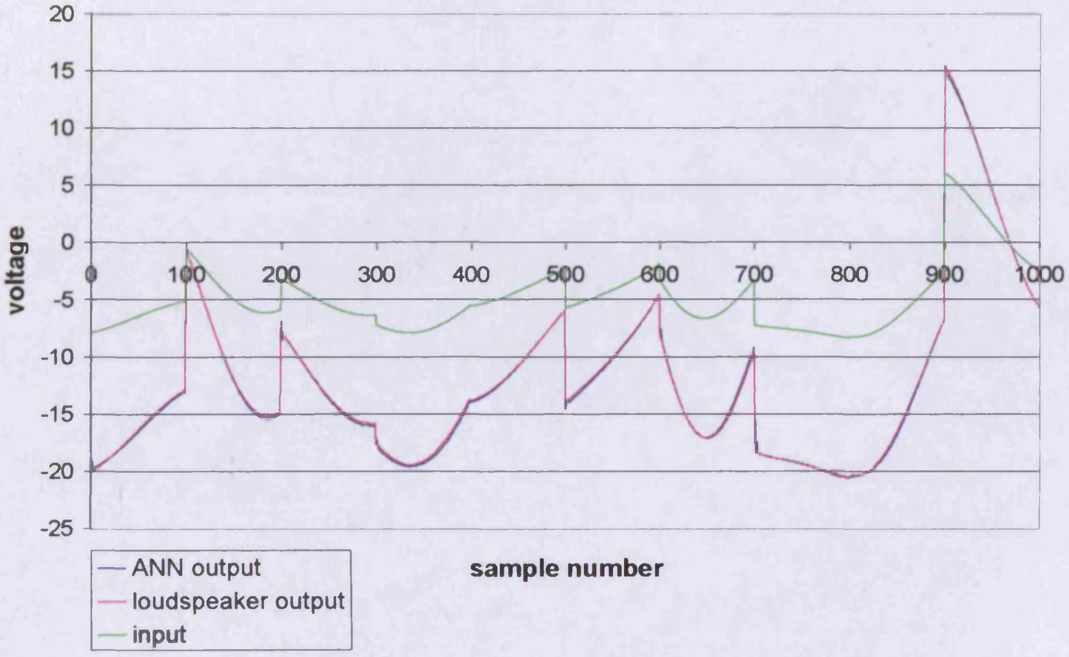


Figure 5.22 – Response of best performing multi layer perceptron ANN Model

This substantiates Miller's (1999) assertion that there exists an interaction between ANN parameters, therefore determining the optimum parameter configuration is not a simple case of identifying suitable parameters individually.

The ANN model performance to larger amplitude inputs is shown in figure 5.22. The training rms error value for this network was 0.0124 and the validation rms error, 0.0129.

5.4 NONLINEARITY IDENTIFICATION CAPABILITY OF ANN MODEL

The validation rms error implied that there was good correlation between ANN model output and actual loudspeaker output, however, to confirm that the ANN model had identified the nonlinearities in the loudspeaker transfer function, an analysis of the data was performed using Pearson's Product Moment Correlation Coefficient (R^2) to identify subsets within the output data sets that had low correlation with the input (as discussed in section 5.1.8.1, this was assumed to suggest nonlinearity). The results from actual loudspeaker output were compared to the ANN model output to determine if they concurred, which would indicate that the ANN model behaved nonlinearly in the same instances as the actual loudspeaker. However, it was determined that the vast majority of the nonlinearity in the training and validation data sets occurred in the low amplitude regions, which was considered more likely to be caused by measurement noise rather than actual nonlinear behaviour of the loudspeaker. Therefore this investigation gave little insight into the performance of the ANN model in identifying nonlinearities.

5.5 FREQUENCY RESPONSE OF ANN MODEL

In order to evaluate the ANN model's frequency response curve, a C++ program, (see Appendix 7), was used to generate a set of sine wave input signals at the sampling frequency corresponding to that of the ANN training data. The program then determined the ANN model's output to the sine waves, which were then analysed using Mathworks MathCAD 11 Fourier transform function. The digital Fourier transform (DFT) was used as the sample length, dictated by the number of points required to form an integer number of periods in the sample at the necessary sampling frequency, did not conform to that required by a fast Fourier transform. However, the increased processing time inherent in a DFT analysis was insignificant for this application. The magnitude responses at each frequency were then compiled to produce a frequency response curve, shown in figures 5.23 and 5.24 for the best performing multi layer perceptron and modified Elman ANN models respectively.

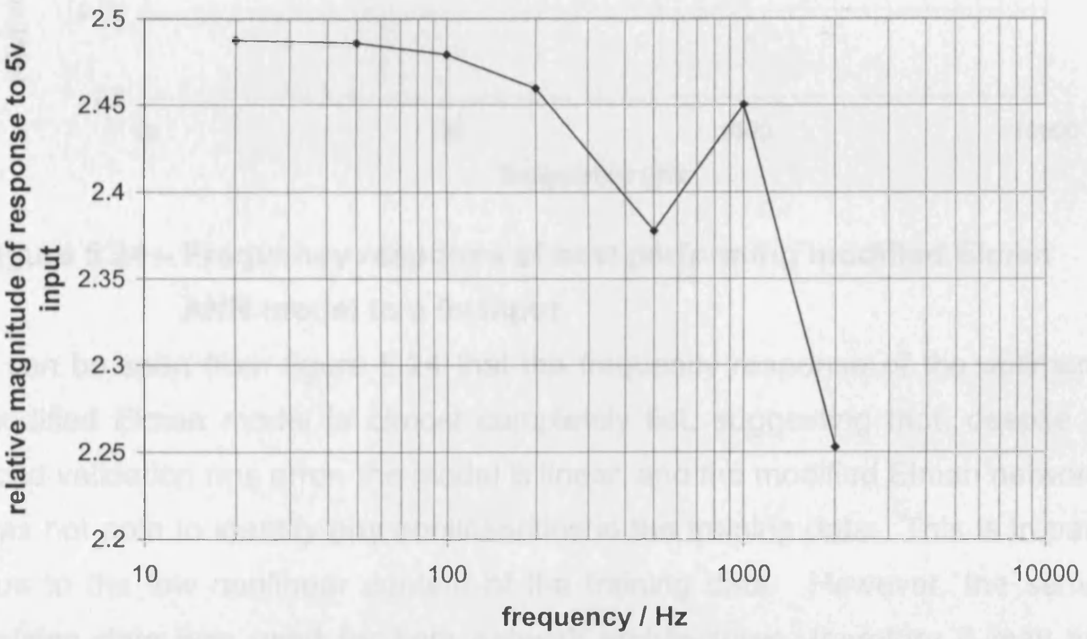


Figure 5.23 – Frequency response of best performing multi layer perceptron ANN model to a 5v input

It can be seen from figure 5.23 that the frequency response of the best performing multi layer perceptron model resembles a low pass filter. However, the response of the actual loudspeaker more closely resembles a band-pass filter, with the response increasing significantly at the lower frequencies and slowly diminishing at higher frequencies. Also, the frequency response generated by the multi layer perceptron model is considerably flatter than the actual loudspeaker response.

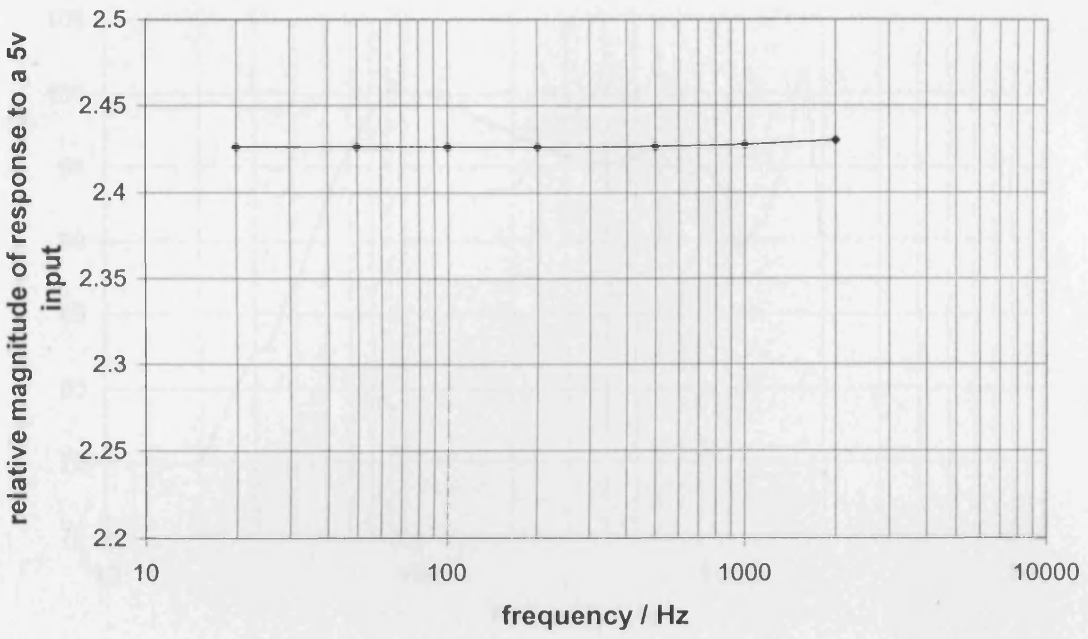


Figure 5.24 – Frequency response of best performing modified Elman ANN model to a 5v input

It can be seen from figure 5.24 that the frequency response of the optimum modified Elman model is almost completely flat, suggesting that, despite a good validation rms error, the model is linear, and the modified Elman network was not able to identify any nonlinearities in the training data. This is in part due to the low nonlinear content of the training data. However, the same training data was used for both network architectures, therefore it may be concluded that the multilayer perceptron was more successful in modelling nonlinear loudspeaker behaviour than the modified Elman architecture.

Further frequency response curves, shown in figures 5.26 and 5.27, were generated to determine the response of the multi layer perceptron model to higher input amplitudes. The maximum input amplitude in the training data was approximately 25v, therefore the model's response to a 20v input, just below the threshold and to a 50v input, well above the threshold was tested.

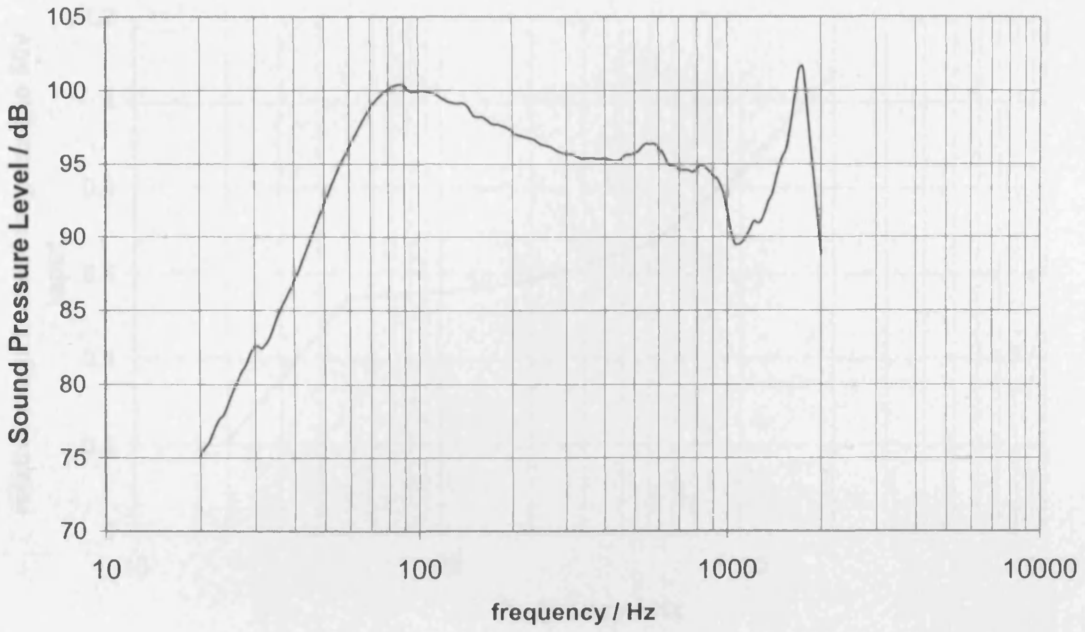


Figure 5.25 – Measured Frequency Response of Loudspeaker

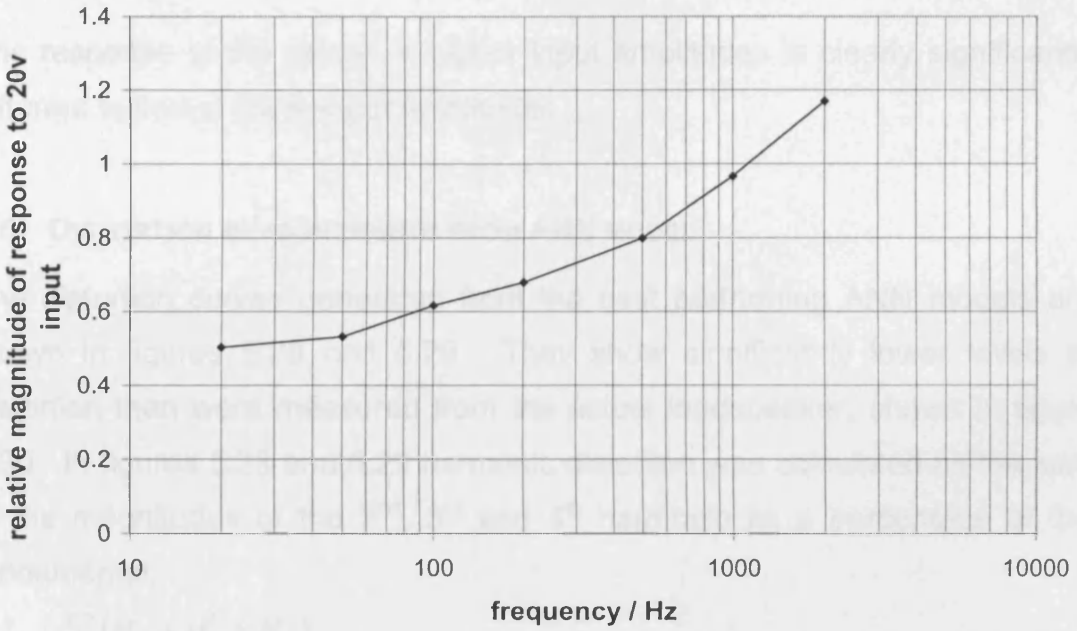


Figure 5.26 - Frequency response of best performing multi layer perceptron ANN model to a 20v input

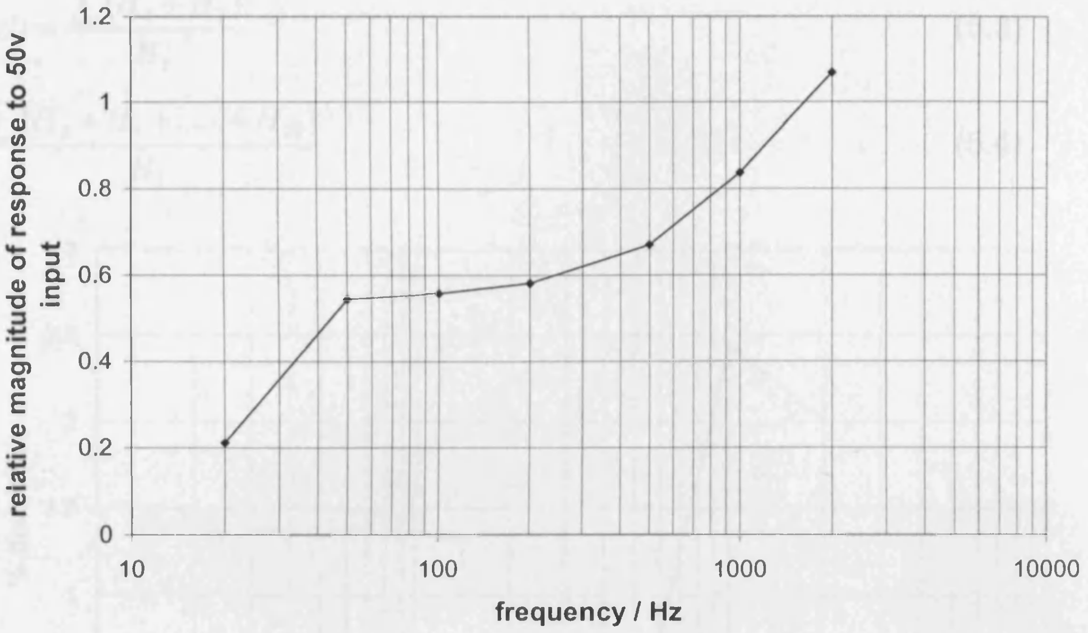


Figure 5.27 - Frequency response of best performing multi layer perceptron ANN model to a 50v input

The response of the model at higher input amplitudes is clearly significantly different to that at lower input amplitudes.

5.6 DISTORTION MEASUREMENTS FROM ANN MODEL

The distortion curves generated from the best performing ANN models are shown in figures 5.28 and 5.29. They show significantly lower levels of distortion than were measured from the actual loudspeaker, shown in figure 5.30. In figures 5.28 and 5.29 harmonic distortion was calculated as the sum of the magnitudes of the 2nd, 3rd and 4th harmonic as a percentage of the fundamental:

$$HD = \frac{\sum(H_2 + H_3 + H_4)}{H_1} \tag{5.1}$$

where *HD* is harmonic distortion and H_n is the magnitude of the nth harmonic. In figure 5.30 the 2nd and 3rd harmonics were calculated as in equations 5.2 and 5.3 respectively and total harmonic distortion as in equation 5.4.

$$HD = \frac{\sum H_2}{H_1} \tag{5.2}$$

$$HD = \frac{\sum(H_2 + H_3)}{H_1} \tag{5.3}$$

$$\frac{\sum(H_2 + H_3 + \ldots\ldots + H_{35})}{H_1} \tag{5.4}$$

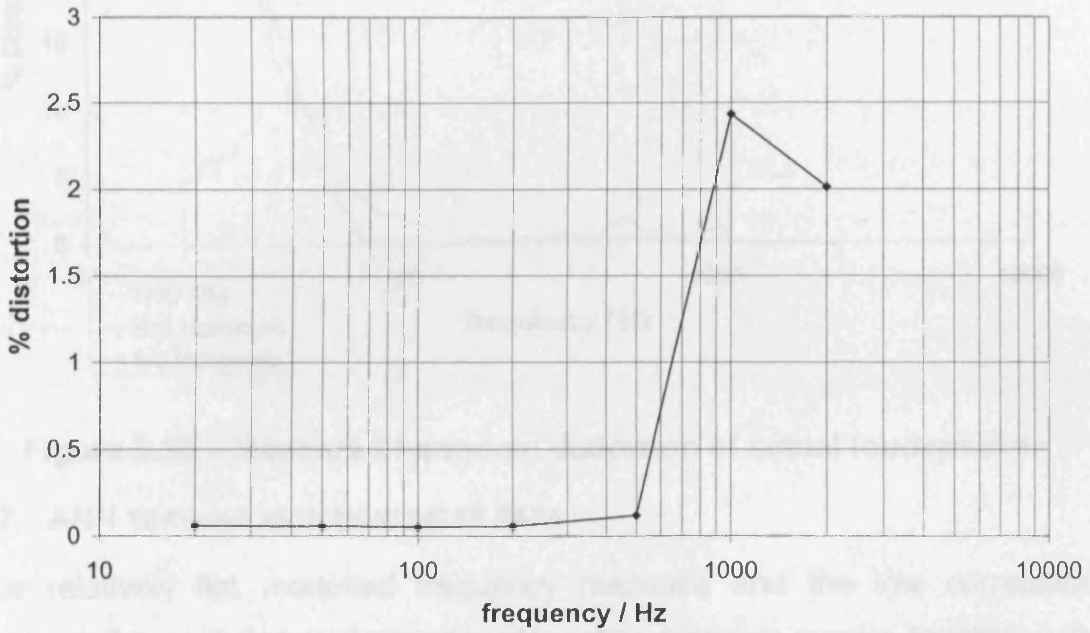


Figure 5.28 – Harmonic distortion of best performing multi layer perceptron ANN model

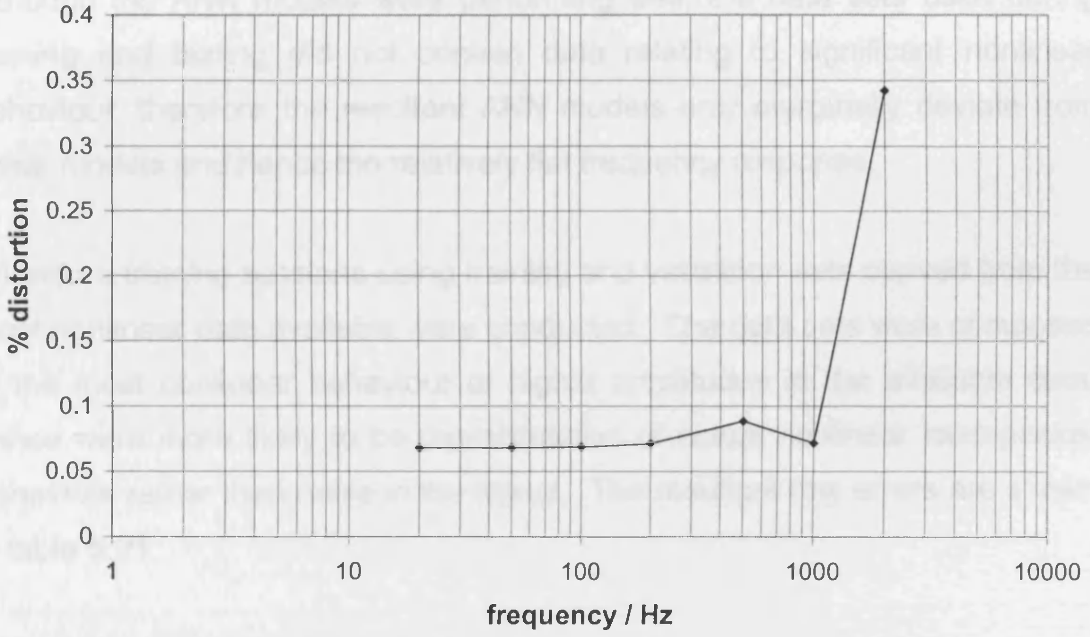


Figure 5.29 – Harmonic distortion of best performing modified Elman ANN model

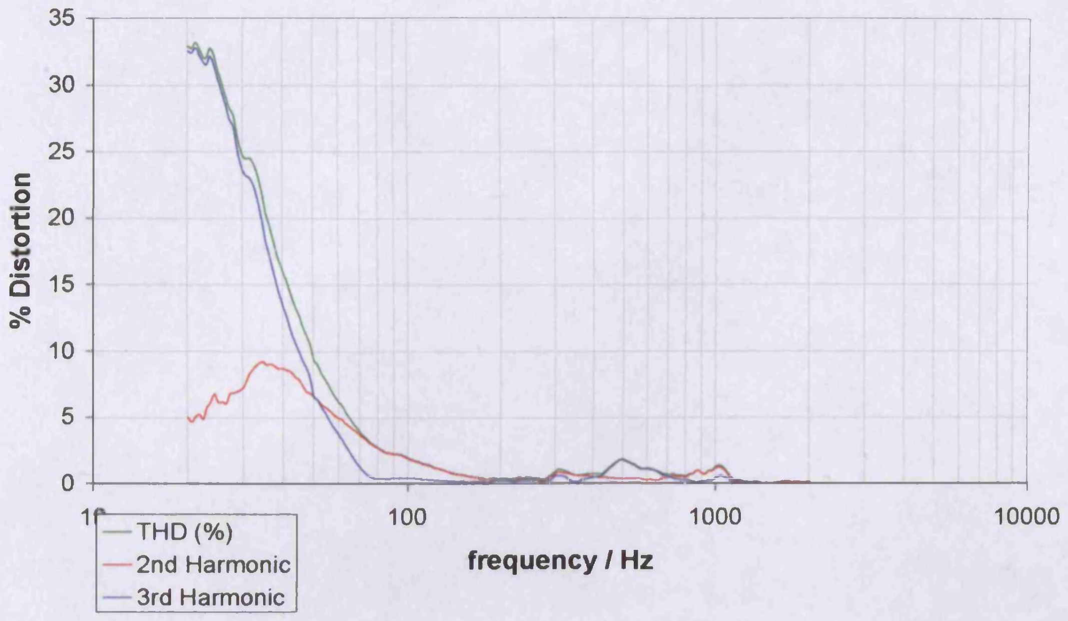


Figure 5.30 – Measured harmonic distortion of actual loudspeaker

5.7 ANN TRAINING WITH NONLINEAR DATA

The relatively flat modelled frequency response and the low correlation between the modelled and actual loudspeaker distortion curves, together with the analysis of the nonlinear content of the training data which suggested the training data contained very few nonlinearities led to the conclusion that although the ANN models were performing well, the data sets used during training and testing did not contain data relating to significant nonlinear behaviour, therefore the resultant ANN models only marginally deviate from linear models and hence the relatively flat frequency response.

Therefore training sessions using training and validation sets derived from the most nonlinear data available were conducted. The data sets were composed of the most nonlinear behaviour at higher amplitudes in the available data, hence were more likely to be representative of actual nonlinear loudspeaker behaviour rather than noise in the signal. The resultant rms errors are shown in table 5.21.

Table 5.21 – Results of training sessions with nonlinear data

Network architecture	Training rms error	Validation rms error
Multi layer perceptron	0.0195	0.0196
Modified Elman	0.0420	0.0442

The frequency response and distortion curves generated from the multi layer perceptron model trained with nonlinear data were a significant improvement, as illustrated in figures 5.31 and 5.32. However, the results from the modified Elman network showed no improvement.

Analysis of the nonlinear content of the multi layer perceptron ANN model output determined that it corresponded well to the nonlinear content of the actual loudspeaker.

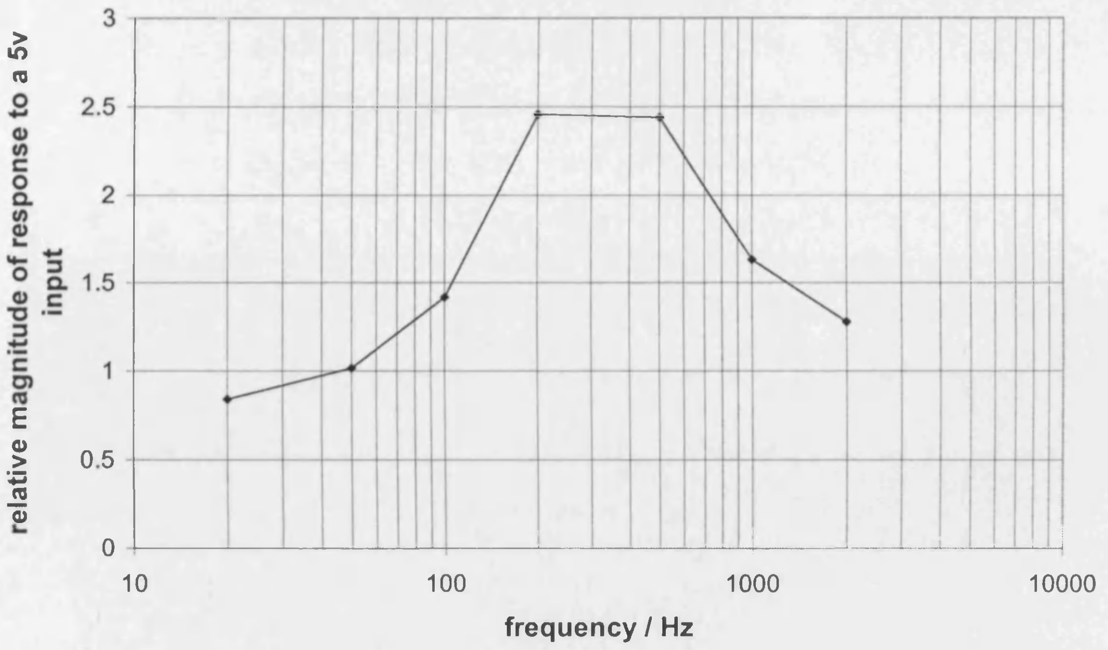


Figure 5.31 – Frequency response of multi layer perceptron ANN model trained with nonlinear data to a 5v input

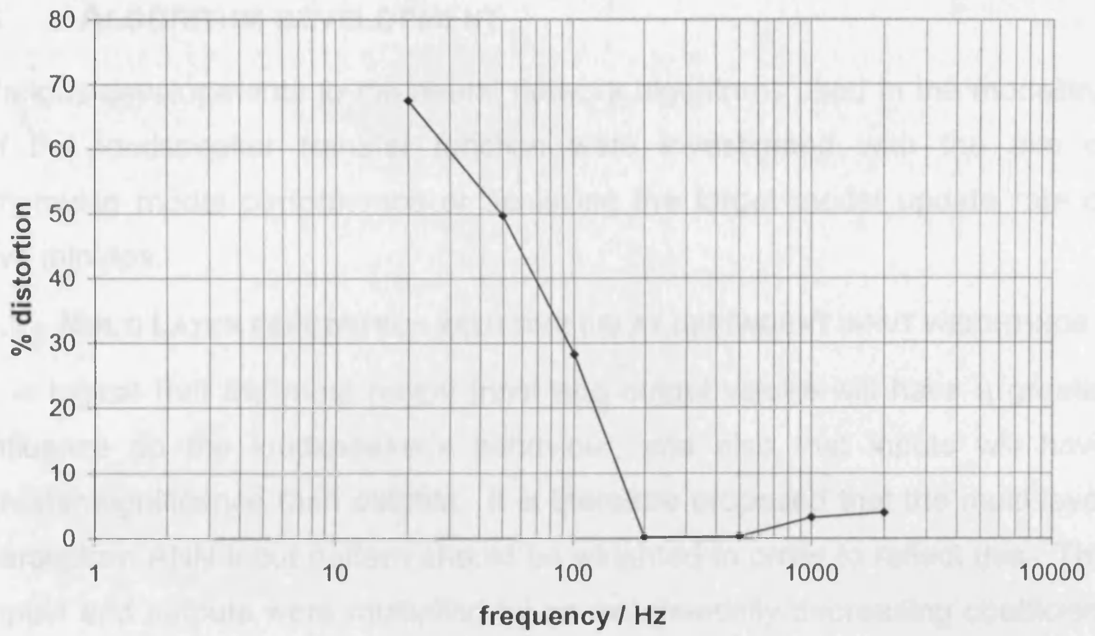


Figure 5.32 – Harmonic distortion of multi layer perceptron ANN model trained with nonlinear data

6 ALGORITHM DEVELOPMENT

Various developments to the neural network algorithms used in the modelling of the loudspeaker transfer function were investigated with the aim of improving model performance or achieving the target model update rate of five minutes.

6.1 MULTI LAYER PERCEPTRON WITH TIME DELAY DEPENDENT INPUT WEIGHTINGS

It is logical that the most recent input and output values will have a greater influence on the loudspeaker's behaviour, and also that inputs will have greater significance than outputs. It is therefore proposed that the multi layer perceptron ANN input pattern should be weighted in order to reflect this. The inputs and outputs were multiplied by an exponentially decreasing coefficient between 0 and 1, and the outputs multiplied by a further constant coefficient between 0 and 1. This would provide some rating of importance to the data. The optimum parameter configuration determined for the multi layer perceptron was employed during these experiments. The resultant ANN model performance is summarised in table 6.1.

Table 6.1 – Comparison of ANN model performance when trained with and without time delay dependent weightings

	Training rms error	Validation rms error
With weightings	0.0166	0.0340
Without weightings	0.0124	0.0129

It was therefore concluded that time delay dependent weightings did not improve the multi layer perceptron's performance.

6.2 ANN TRAINING DURATION

The project aim was to realise a model update rate of 5 minutes. This necessitated that each subsequent ANN model be trained within those 5 minutes. There exists a direct relationship between the time required for the completion of ANN training and the following variables:

- number of neurons employed in the ANN architecture
- number of training epochs
- size of the training set.

However, sections 5.1.7 and 5.2.7 established that there is an inverse relationship between size of the training set and achievable rms error for multi layer perceptrons and modified Elman networks. This is also true for the number of training epochs for the modified Elman network and for the multi layer perceptron below the threshold of overtraining (see sections 5.1.3 and 5.2.3). The optimum number of neurons was found to be system dependent rather than a function of size (see sections 5.1.2 and 5.2.2). Therefore, the optimum network architectures described in section 5.3 were employed during this investigation and tests were conducted to ascertain a suitable compromise between the number of training epochs and the training set size in order to optimise the network performance over a training period of five minutes.

Two approaches to weight initialisation were investigated:

- Randomly initiating the ANN weights.
- Initiating the ANN weights to previously trained values.

During this investigation the training sets comprised of data collected in small segments (100 samples) over a time period of 4 seconds and the previously trained weights were those resulting in the best performing ANN model as discussed in section 5.3. In the case of the multi layer perceptron, this was the parameter values that resulted in the lowest achieved validation rms error (see table 5.20) and for the modified Elman network, optimum parameter values (see table 5.18). The ANN parameters in these experiments were also set to these configurations. The number of training samples and training epochs used in the experiments with the multi layer perceptron network are shown in table 6.2 with the resultant validation rms errors for randomly initiated weights and previously trained weight initiation.

The number of training samples and training epochs used to train the modified Elman network are shown in table 6.3 with the validation rms errors for the modified Elman network when trained with randomly initiated weights and previously trained weight initiation.

Table 6.2 – Training results obtained over 5 minute training period for multi layer perceptron network

Number of training samples	Number of training epochs	Validation rms error - random weight initiation	Validation rms error – previously trained weight initiation	Validation rms error – previously trained weight initiation plus altered ANN configuration
1600	2880	0.110	0.0280	0.0127
3200	1440	0.159	0.0176	0.0124
6400	720	0.128	0.0199	0.0124
12800	360	0.250	0.0308	0.0134
25600	180	0.246	0.0162	0.0131
51200	90	0.179	0.0144	0.0130

Table 6.3 – Training results obtained over 5 minute training period for modified Elman network

Number of training samples	Number of training epochs	Validation rms error - random weight initiation	Validation rms error – previously trained weight initiation
1600	10000	0.121	0.0693
3200	5000	0.101	0.0536
6400	2500	0.141	0.0539
12800	1250	0.142	0.0591
25600	625	0.0967	0.0622
51200	375	0.106	0.0649

It was observed that the performance of both architectures was significantly improved relative to the performance of randomly initiated weights when the weights were initiated to previously trained values.

However, the data used in this experiment was collected over a relatively small time period, which may limit the conclusions that may be drawn from the results. It is possible that over longer time periods the response of the loudspeaker will alter more radically and thus substantially larger alterations to the weights will be required, which may be more difficult to achieve over the target training period. Despite this it is anticipated that there would be greater advantage in utilising previously trained weight values, as they are likely to be closer to an effectively performing configuration than weights that are randomly initiated.

In the case of the modified Elman the second training session improved the performance of the ANN model when the training set contained 3200 and 6400 samples and was trained over 5000 and 2500 epochs respectively, however, the validation rms errors attained by the multi layer perceptron network with the ANN weights initiated to the previously trained values were higher than that of original training session, hence the further training worsened the performance of the ANN rather than improved it.

Therefore, several of the multi layer perceptron ANN parameters were adjusted, including the learning rate and momentum term, in order to determine their effect upon the training results with the aim of improving the ANN model performance after the second training session. Adjusting the momentum term resulted in no significant difference in the ANN model performance, however, when the learning rate was reduced to 1 x 10^{-8}, the resultant validation rms errors were significantly reduced for all training set size and epoch combinations, as shown in the right hand column of table 6.2. However, the lowest validation rms error attained was only equal to that of the original training session, no improvement in ANN performance was achieved with the second training session.

6.3 ONLINE TRAINING

As mentioned in section 3.4.6, the modelling of the loudspeaker transfer function using ANNs has already been accomplished; the aim of this project was to develop a method of training the ANN to a satisfactory level of performance over a short time period so as to accommodate the time variant parameters of the loudspeaker transfer function in the model. A linearisation scheme based upon such a model would theoretically reduce distortion more efficiently, as the model would be a closer approximation to the loudspeaker's current behaviour than that of a generalised model.

In this section, the development of algorithms to continuously update the ANN model of the loudspeaker transfer function are discussed. Although the multi layer perceptron's performance was superior in comparison to the modified Elman network's when trained with large training data sets over a high number of epochs, the modified Elman network produced a relatively larger improvement in performance when trained a second time with small training sets over a low number of training epochs. The modified Elman network also has the advantage of only requiring the data from the current time step as its memory is integral, and therefore does not require a tapped delay line. Therefore, both architectures were investigated for online training.

The target training time period was 5 minutes. Two methods were considered to update the model active in the linearisation scheme during this period. The first method was to continuously update the ANN weights with errors generated from every sample measured real-time, with each sample used only once. The second method was to train the ANN with a selected training set over a number of epochs and therefore longer training periods, up to the full five minutes, with the new model becoming operative in the linearisation scheme at the end of the training period. The second method would result in a relatively large alteration in the weights once or several times over the 5 minute update period whereas the second method would generate more regular, substantially smaller alterations.

It was considered necessary to include some degree of validation in the algorithm to reduce the possibility that the ANN model diverges from actual loudspeaker response, as this may result in the linearisation scheme introducing additional distortion to the loudspeaker output. The method proposed performs a validation calculation using data measured concurrently with the training sample after every weight alteration to determine if the ANN model performance has been improved by the alteration. If the ANN performance is not improved the previous weight configuration remains active in the linearisation scheme.

The algorithms employed for the online training of the modified Elman network are illustrated in figures 6.1 and 6.2. The modified Elman network is known to have a tendency to be unstable during training due to the inclusion of feedback in the training algorithm, hence the continuously updating algorithm could cause significant fluctuation in the linearisation scheme's performance. Therefore, the inclusion of a validation calculation in the algorithm is of increased importance in this case. The algorithms used to train the multi layer perceptron network were identical to those outlined in figures 6.1 and 6.2 with the exception of the alterations to the context units.

Experiments were conducted to determine optimum parameter values for both methods, the results are shown in tables 6.4 and 6.5. In all cases the weights were initiated to those of the best performing model from section 5.3 and then trained according to the algorithms outlined in figures 6.1 and 6.2.

The first row in tables 6.4 and 6.5 show the results from the training session using the algorithm in figure 6.1, where each training sample is used to update the weights only once and a validation calculation is performed after each of these iterations. The subsequent rows show training sessions where firstly the training set was enlarged so each training sample was still used only once but the validation calculation was only carried out after the weights had been adjusted by each of the samples in the training set and secondly where training sets were used for several training epochs, as in figure 6.2.

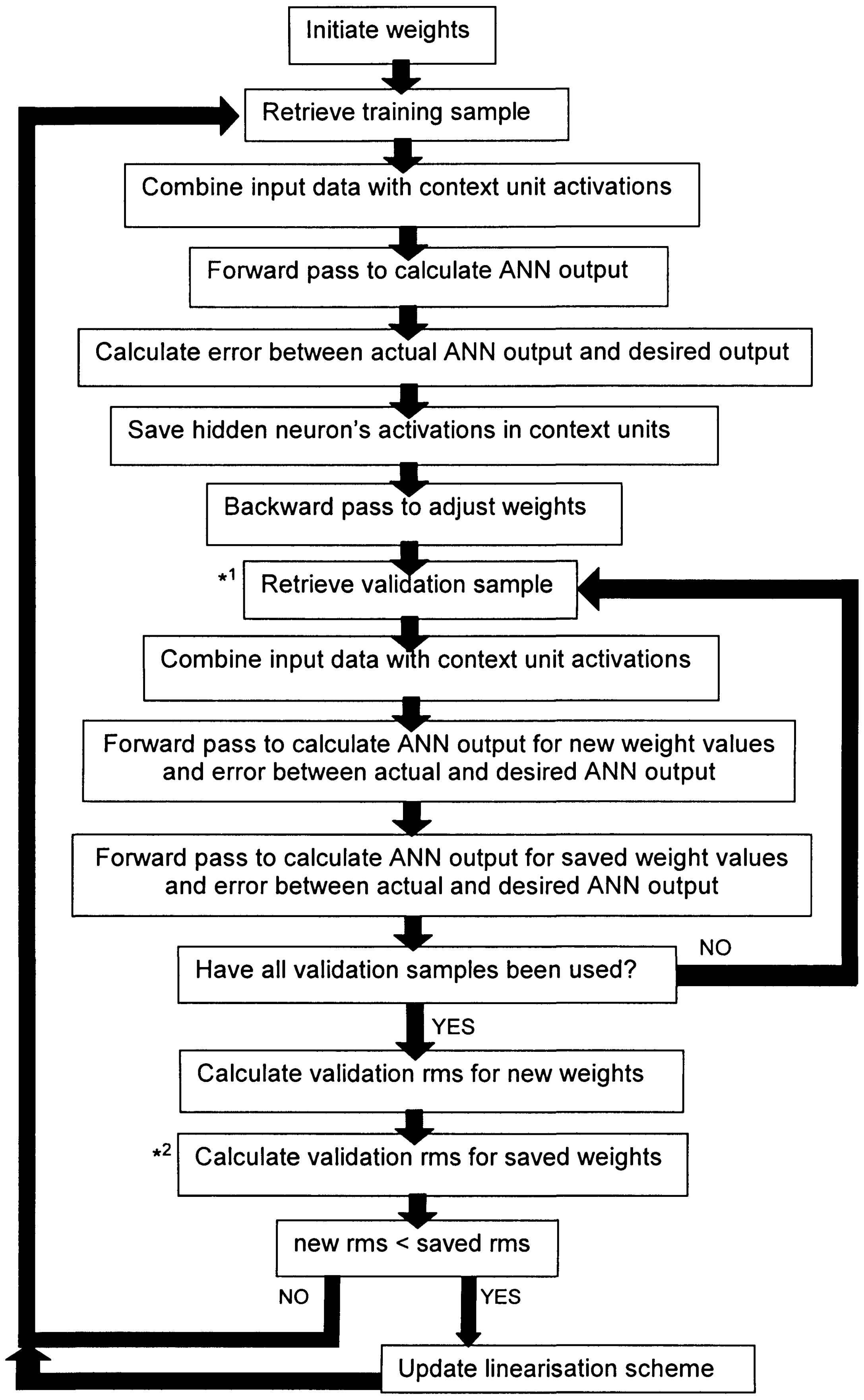

Initiate weights
Retrieve training sample
Combine input data with context unit activations
Forward pass to calculate ANN output
Calculate error between actual ANN output and desired output
Save hidden neuron's activations in context units
Backward pass to adjust weights
*1
Retrieve validation sample
Combine input data with context unit activations
Forward pass to calculate ANN output for new weight values and error between actual and desired ANN output
Forward pass to calculate ANN output for saved weight values and error between actual and desired ANN output
Have all validation samples been used?
NO
YES
Calculate validation rms for new weights
*2
Calculate validation rms for saved weights
new rms < saved rms
NO
YES
Update linearisation scheme

Figure 6.1 – Online training algorithm where linearisation scheme is continuously updated

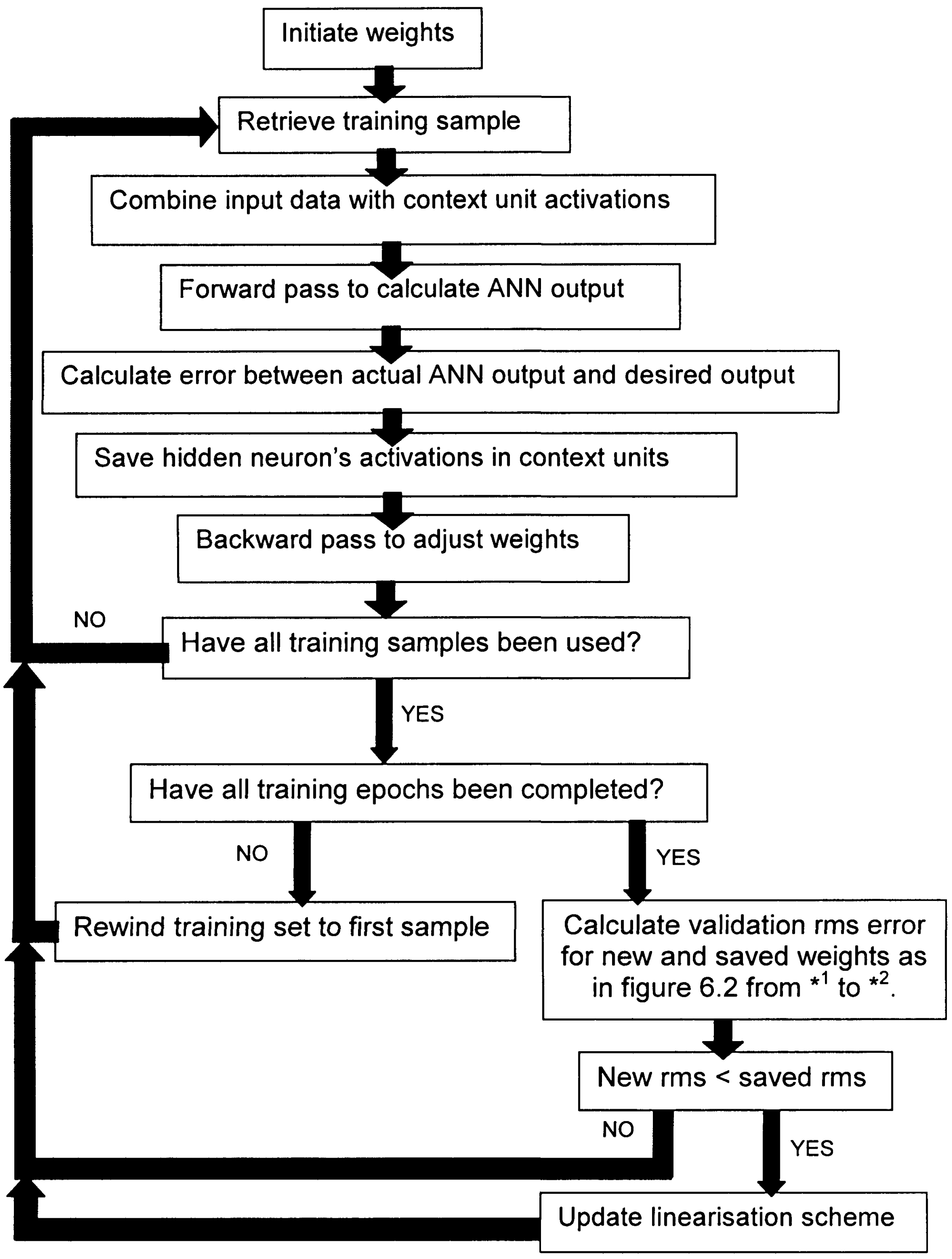

Initiate weights
Retrieve training sample
Combine input data with context unit activations
Forward pass to calculate ANN output
Calculate error between actual ANN output and desired output
Save hidden neuron's activations in context units
Backward pass to adjust weights
NO
Have all training samples been used?
YES
Have all training epochs been completed?
NO
YES
Rewind training set to first sample
Calculate validation rms error for new and saved weights as in figure 6.2 from *1 to *2.
New rms < saved rms
NO
YES
Update linearisation scheme

Figure 6.2 – Online training algorithm where linearisation scheme is updated at the end of the training period

Table 6.4 – Results of continuous training with multilayer perceptron network

Number of samples in training set	Number of samples in validation set	Training epochs	Training time	Weight alterations that result in a weight update / %	Rms error lower than with no training / %	Mean rms error with training	Mean rms error with no training	Reduction in mean rms error / %
1	100	1	-	55.9	36.6	0.0624	0.0604	103
10	100	1	-	57.0	53.6	0.0590	0.0591	99.8
100	100	1	-	58.8	49.7	0.0604	0.0639	94.5
1000	100	1	-	58.9	50.3	0.0692	0.0683	101
10000	100	1	-	47.4	42.1	0.0395	0.0388	101
1000	100	10	55s	57.3	68.6	0.0656	0.0683	96.0
1000	100	50	4m10s	65.9	68.6	0.0609	0.0683	89.2
10	100	100	1m10s	54.8	54.7	0.0568	0.0591	96.1
10	100	500	4m10s	58.8	45.1	0.0542	0.0591	91.7
100	100	100	4m25s	52.5	57.6	0.0565	0.0639	88.4

Table 6.5 – Results of continuous training with modified Elman network

Number of samples in training set	Number of samples in validation set	Training epochs	Training time	Weight alterations that result in a weight update / %	Rms error lower than with no training / %	Mean rms error with training	Mean rms error with no training	Reduction in mean rms error / %
1	100	1	-	69.1	37.7	0.282	0.140	201
10	100	1	-	67.3	50.5	0.140	0.137	102
100	100	1	-	51.2	66.3	0.135	0.145	93.1
1000	100	1	-	47	67.6	0.139	0.152	91.4
10000	100	1	-	47.4	73.7	0.0907	0.128	70.9
1000	100	10	7s	51.9	74.1	0.128	0.152	84.2
1000	100	100	1m10s	44.9	75.7	0.122	0.152	80.3
1000	100	200	2m20s	48.1	77.8	0.120	0.152	78.9
1000	100	400	4m45s	51.4	80.1	0.117	0.152	77.0
10000	100	100	1m20s	47.4	84.2	0.0785	0.128	61.3
10000	100	400	5m10s	57.9	84.2	0.0773	0.128	60.4

For both architectures, the multi layer perceptron and the modified Elman, continuous training as in figure 6.1 resulted in a higher mean validation rms error than if no training occurred i.e. the original weight configuration was used throughout the period. This was also generally the case for most training sessions where only one training epoch was employed, with the exception of the modified Elman network with 10000 training samples, where the mean validation rms error was reduced by 30 per cent by retraining. Significantly improved results were achieved when several training epochs were employed; the best result for the multi layer perceptron model being a reduction in the validation rms error of 11.6 per cent when trained with training sets of 100 samples over 100 epochs, over a 4 minute 25 second time period, and for the modified Elman network, a reduction in the validation rms error of 39.6 per cent when trained with training sets of 10000 samples over 400 epochs, over a 5 minute 10 second time period. The modified Elman network had a significant advantage of speed of training, therefore a substantially larger training set could be employed over the 5 minute training period, hence the larger reduction in the rms error. However, the multi layer perceptron had a significantly lower validation rms error overall, hence may be considered the more effective of the two ANN architectures.

7 ARTIFICIAL NEURAL NETWORK FOR LOUDSPEAKER FAULT DETECTION

7.1 INTRODUCTION

The aim of this project was to train neural networks with data from the end of line test and to investigate methods of using the ANNs to improve the classification performance of the test. The data used during the investigation were the results of the frequency response and rub and buzz tests. The network architecture employed was a multi-layer perceptron trained with the backpropagation algorithm. Data were analysed to ascertain the appropriateness for ANN training and suitable preprocessing techniques investigated. ANN parameters were also investigated in order to determine the optimum configuration for this application. Two loudspeaker models were used during this investigation; the Harman/Becker product codes were 79-65wa35 for the loudspeaker used for frequency response analysis and 99-100bm16 for the loudspeaker used for rub and buzz analysis. Developments were made to improve the sophistication of the ANN to analyse the data and produce a response relating to the frequency band in which the loudspeaker had distortion levels above the test limits.

7.2 END OF LINE TEST

Once production is complete the loudspeakers are tested and those that fail the test discarded. The loudspeaker is placed on a baffle which leads to an anechoic box. Each loudspeaker has a unique baffle with the correct profile cut out of it so the loudspeaker fits perfectly into it. The edge of the hole is lined with a foam gasket that the loudspeaker sits on. This ensures vibration is not transmitted to the baffle, as this would cause audible distortion that would result in the loudspeaker failing the test and being rejected. The anechoic box is lined with triangular wedge-shaped foam, the dimensions of which are designed to absorb low frequency sound waves, and the material the wedges are made of enable the high frequencies to be absorbed. This ensures that no echoes occur, so only the original sound is detected by the microphone, which is located in the anechoic box just below the loudspeaker.

The software used to control the loudspeaker input signal, collect and analyse the measured loudspeaker output is SOUNDCHECK^{TM} 4.1 (developed by S. Temme).

The standard stimulus is R20. This is a sine swept signal with a step size of $^1/_6$ of an octave. The step size can be altered, for example, to $^1/_3$ of an octave. This would mean half as many data points are collected and hence half the processing time is required, but a reduced resolution would be obtained. SOUNDCHECK^{TM} 4.1 then plots or calculates all the parameters discussed in section 7.2.1 and compares them to predefined limits. A display then shows whether the parameters are within their limits. Any loudspeakers that fail the test are tested again as anomalies may be smoothed out during the first test.

The loudspeakers must be tested at the correct volume to identify the harmonics and ambient noise must be kept as constant as possible. Background noise is averaged out of the SOUNDCHECK^{TM} 4.1 test but any abrupt, loud noise may be picked up by the microphone and the loudspeaker would be failed unnecessarily. The test plugs may get dirty and emit an inconsistent signal, they also have fragile electrical connections that fail quite regularly, thus the plugs are replaced regularly under a preventative maintenance scheme. Noise may also be generated by faults in the test box, wires or connectors.

7.2.2 Parameters measured

- FREQUENCY RESPONSE (FR) – 90 to 120 values measured over a suitable frequency range for the loudspeaker.
- SENSITIVITY – an average of 4 or 5 values measured at the fundamental frequency.
- IMPEDANCE (Z) – as many values measured as the frequency response.
- BASS RESONANCE (BR/F_0) – a single value, which is the frequency at which impedance is a maximum. F_0 is identified on the impedance curve in figure 7.1. An algorithm is used to ensure that the first maximum is identified by the test, not any subsequent maximum further along the curve.

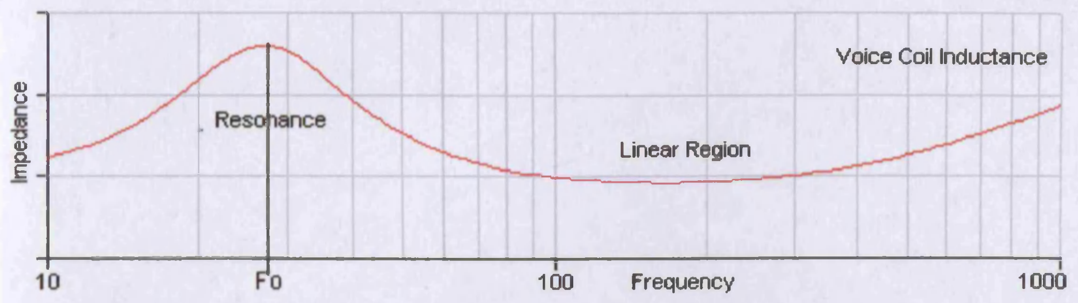


Figure 7.1 – Position of bass resonance on impedance curve (Elliot, 2003)

- DC RESISTANCE (DCR) – The minimum value for impedance, Z_{MIN}, is a good indication of the dc resistance of the loudspeaker. The position of Z_{MIN} on the impedance curve is illustrated in figure 7.2. An algorithm is also used to identify the minimum point on the impedance curve.

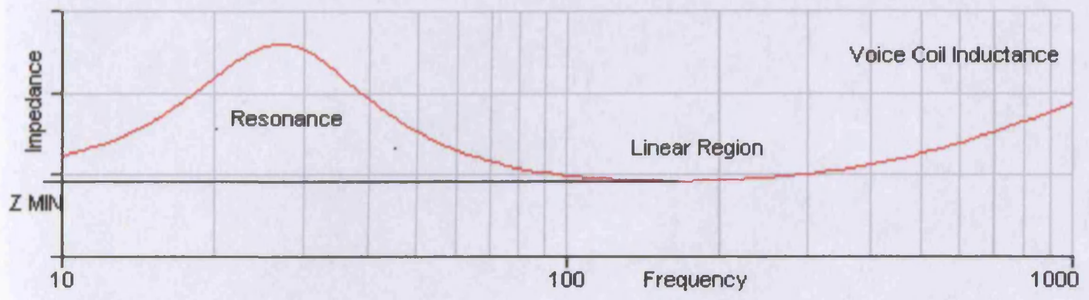


Figure 7.2 – Position of Z $_{MIN}$/DCR on Impedance Curve (Elliot, 2003)

- RUB AND BUZZ – A test for audible harmonic distortion. The number of data points collected is the same as for the frequency response. The 10th to 35th harmonics are recorded and checked against limits. The 2nd and 3rd are sometimes used to find manufacturing faults that are difficult to identify audibly. Harmonic distortion is calculated as a percentage of the fundamental, as shown in equation 7.1.

$$\frac{\sum(H_{10}+H_{11}+......+H_{35})}{H_1} \tag{7.1}$$

- TOTAL HARMONIC DISTORTION (THD) – The number of data points collected is the same as the frequency response. In comparison to the rub and buzz measurement, all harmonics are included in the calculation for total harmonic distortion, as shown in equation (7.2).

$$\frac{\sum(H_2 + H_3 + \ldots\ldots + H_{35})}{H_1} \tag{7.2}$$

7.2.3 Anechoic chamber tests

The small volume, small inflection end of line test does not thoroughly test a loudspeaker. A far more comprehensive test is carried out in a large anechoic chamber at the development stage of the loudspeaker and at regular intervals to validate the end of line test. The tests are conducted in an anechoic chamber, which is a small room that is lined with triangular wedge-shaped foam, as in the end of line test box, except the wedges are much larger. A microphone is held a specific distance away from the loudspeaker, depending upon whether a near field or far field measurement is required. The loudspeaker is mounted on a baffle between the anechoic chamber and the antechamber adjacent to it, which houses the rear loudspeaker assembly and allows the loudspeaker to 'breathe', i.e. provides the appropriate pressures to the rear of the cone. This facility enables loudspeakers to be tested with accuracy at large volumes and inflections. The chamber is also used for design verification for customers.

7.3 LOUDSPEAKER FAULTS

Loudspeaker faults caused during production can be generally categorised as mechanical, chemical or placement asymmetry. Some faults can be tracked though a whole batch but more frequently there is a significant degree of disparity in the faults detected in any one batch. Harman/Becker has classified 82 possible loudspeaker fault diagnoses (see Appendix 2), many of which are quite general descriptions of the failure mode. The most common of these faults are summarised below.

7.3.2 Coil assembly

Wire is wound around a former on very accurate mandrills. There are many turns and often several layers of wire that make up the coil. In some models the braids are soldered directly on to the former, in which case soldering faults may occur. The dominant fault at this stage is a non-symmetric coil, which will cause harmonic distortion and therefore lead to a rub and buzz failure at the end of line test. Many of these non-symmetric coils are identified by eye as the coil is connected to the spider, and are discarded at this stage. However, small deviations may not be detected and the faulty coil will continue down the production line. Other faults include uneven coil winding and winding occurring at the wrong place on the former, however, both are extremely rare.

7.3.3 Spider

Glue is applied to the coil on a rotating machine and the spider placed on the coil by hand. The spider may not be attached perpendicularly to the coil or the glue may not take properly, which would result in harmonic distortion and therefore a rub and buzz failure. This is often identified on the production line, but small imperfections may not be visible and will affect the final product. The spider is impregnated with a compound to make it stiff. This may be applied unevenly, which would give an unevenly distributed compliance of the spider and unpredictable characteristics of the final loudspeaker, identified during the frequency response test. There have also been occasions when the wrong spider has been used for a production run, resulting in incorrect properties of the loudspeaker. This was due to the similar appearance of many of the different spiders.

7.3.4 Chassis

The magnet and the plate are placed into the chassis using gap gauges to ensure exact placement. At both stages glue is applied by an automatic robotic arm and pressure applied to aid adhesion. If the gap gauge is worn the magnet may not be placed precisely, leading to an incorrect magnetic field. Glue is then applied around the pole piece to receive the spider, which is placed by hand. The amount of glue the robot delivers is critical and is tested periodically by weighing it. Too little glue will leave segments where the components have not adhered completely and too much can leak into the spider and alter its compliance. The coil former ensures that the coil sits at the correct height in the magnetic field and a gap gauge centres the coil in the gap. At this stage many loudspeakers have to be handled as they must be removed from the production line to allow the water based glue to cure, creating a potential opportunity for damage to occur.

7.3.5 Cone and surround

Most cone and surround assemblies are supplied and quality controlled by other manufacturers. Glue is applied to the rim of the chassis and the cone and surround dropped on by hand and pressure is applied to aid adhesion. The neck joint is then glued and cured and the gap gauge removed. Insufficient glue can lead to harmonic distortion and therefore a rub and buzz test failure, and excessive glue can alter the compliance of the surround and therefore the response of the loudspeaker, which is identified during the frequency response test. A damaged cone can result in high frequency distortions, making the loudspeaker sound out of tune.

7.3.6 Dome

Some loudspeaker models have their coil vacuumed out at this stage to reduce the possibility of debris in the gap. More glue is applied at the neck joint and the dome inserted. If the dome is offset it will change the frequency response of the loudspeaker.

7.3.7 Braid connection

The coil wires either terminate on the coil former underneath the cone or they are threaded though the cone and terminate part way up the cone. The first method involves the coil wire braids being threaded though holes in a panel in the chassis, they are then formed with a special tool by the operator, cut, and the tabs closed to make the connection. With tab connections there can be problems with damaged tabs causing an open circuit. Alternatively, the braids terminate on the cone, in which case the connection is soldered. If too little solder is used the connection may fail, and not necessarily at the end of line test; it may take some time to fail and have to be returned under warrenty by the customer. If the solder is not heated enough a dry joint may occur which will also lead to a connection failure.

The braids being cut too long may result in an audible ticking, caused when the cone hits the braids.

7.3.8 Pad ring

A foam pad ring is glued to the rim of the loudspeaker for the grill to rest on. Too little glue can lead to edge buzzes and too much can leak on to the surround and impede the vibration of the loudspeaker.

7.3.9 Doping

Some loudspeaker models have a mixture of wood glue and water applied to the joint between the surround and the cone to improve the joints properties. Irregularities in this can lead to a deviation from the desired loudspeaker response.

7.3.10 Magnetisation

The final operation on the production line is to magnetise the loudspeaker. This is carried out last to allow easier movement along the production line. A 1.5kA current is passed through a coil and the 1.2 T vertical field saturates the magnet. Occasionally the polarity of the loudspeaker is incorrect, which is identified at the end of line test.

7.4 DATA ANALYSIS

7.4.2 Frequency response data

Initial observations of data from the 79-65wa35 loudspeaker frequency response test determined that the accepted loudspeaker data have a relatively flat response, contained within a small range between 100 and 130 dB approximately, as shown in figure 7.3.

The rejected loudspeaker data, shown in figure 7.4, appear to have three distinct patterns; some rejects have values extremely close to the test limits but one or several points are just outside the limits, another set appear to have values approximately 50 dB less than the accepted loudspeakers' values, and the remainder have values mainly between 0 and 50 dB. This could indicate a correlation between the data pattern profile and the failure mode, which could be utilised in the classification of specific faults.

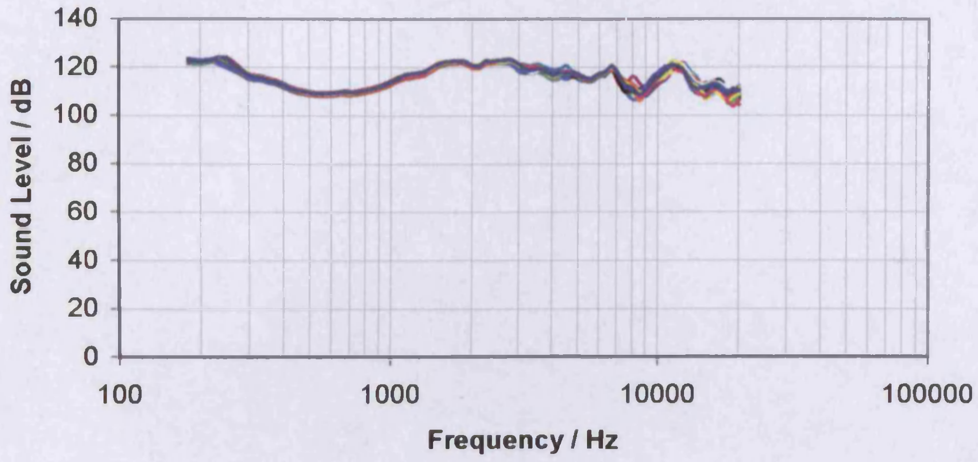


Figure 7.3 – Frequency response accepted data

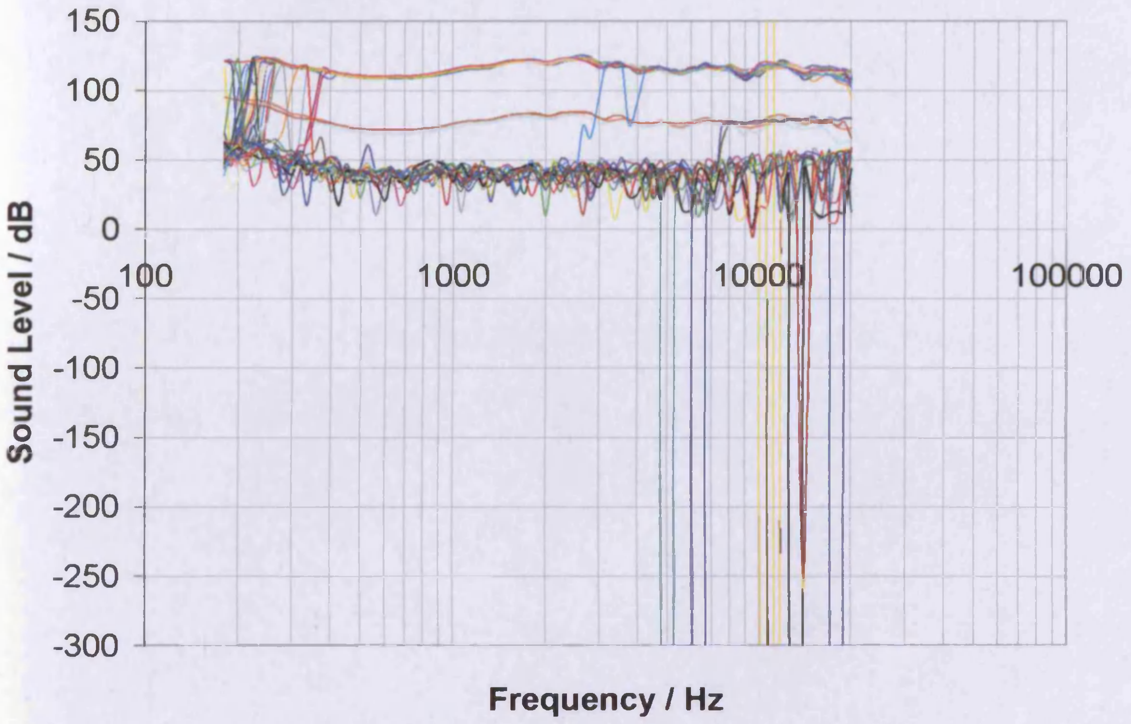


Figure 7.4 –Frequency response reject data

It was ascertained that several patterns in the set of reject data with values mainly between 0 and 50 dB have recorded values of negative infinity. SOUNDCHECK^{TM} records this value if there is an open circuit. In order to incorporate these data into the ANN, a value that the network could recognise was required, so a value of negative 10000 was substituted for negative infinity in the interim to observe how the network would react.

The network did not classify the rejects containing infinity blips with adequate accuracy. On further inspection the classification of other data patterns was also unsatisfactory. It is proposed that it was not possible for the weights to adjust to an idiosyncrasy with such an extreme value that only occurred in a small number of training patterns. However, the idiosyncrasy was sufficient to corrupt the configuration of the weights for classification of other data patterns. It is feasible that with longer training periods and a larger number of training patterns containing the idiosyncrasy, the network could be configured to classify these patterns adequately. However, in order to progress, the patterns containing negative infinity blips were removed from the training set.

The desired output of the frequency response test is not absolute values but the shape of the response curve; a relatively flat frequency response is desirable and thus would result in a loudspeaker being accepted, and a response with considerable variation over the frequency range is undesirable and would result in the loudspeaker being rejected. SOUNDCHECK^{TM} 4.1 uses a system of floating test limits to achieve this, thus so long as the magnitude of the loudspeaker's frequency response is contained within a predetermined range, the loudspeaker will be accepted, as illustrated in figure 7.5.

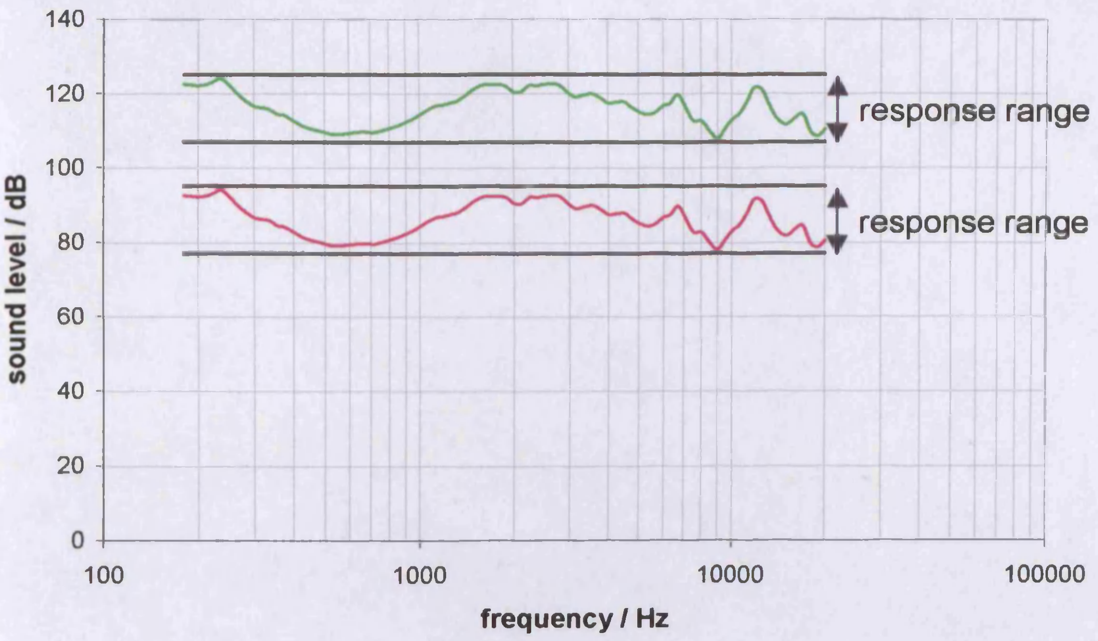


Figure 7.5 – Illustration of two accepted loudspeaker frequency responses within the floating test limits but with different absolute values

It was anticipated that the ANN would have difficulty emulating this form of test limits as it relies upon the absolute values of the data pattern to determine its output. The loudspeaker data patterns would require normalisation prior to being input to the ANN in order to over come this.

7.4.3 Rub and buzz data

The investigation into data from the rub and buzz test of the 99-100bm16 loudspeaker established that the majority of the data in the accepted loudspeaker data patterns are contained in a very small range that has a peak of around 0.6 per cent distortion at approximately 100Hz. This rapidly reduces to below 0.1 per cent distortion after 200Hz and remains well below this level with a gradual taper to nearly zero at 2150Hz, as shown in figure 7.6.

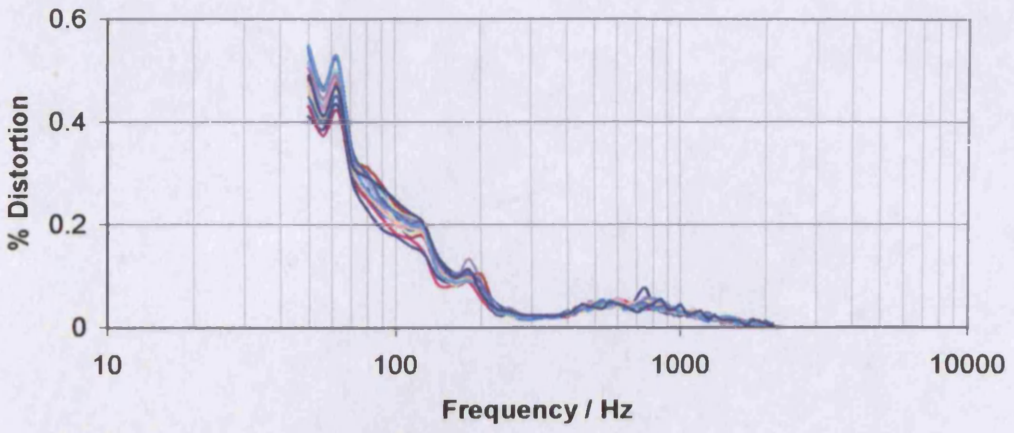


Figure 7.6 – Rub and buzz accepted data

A significant proportion of the reject data patterns are contained in a relatively small range in the vicinity of the accepted loudspeaker data; with the remainder consisting of random noise with up to 100 per cent distortion. This is illustrated in figure 7.7 which shows 37 rejected data patterns.

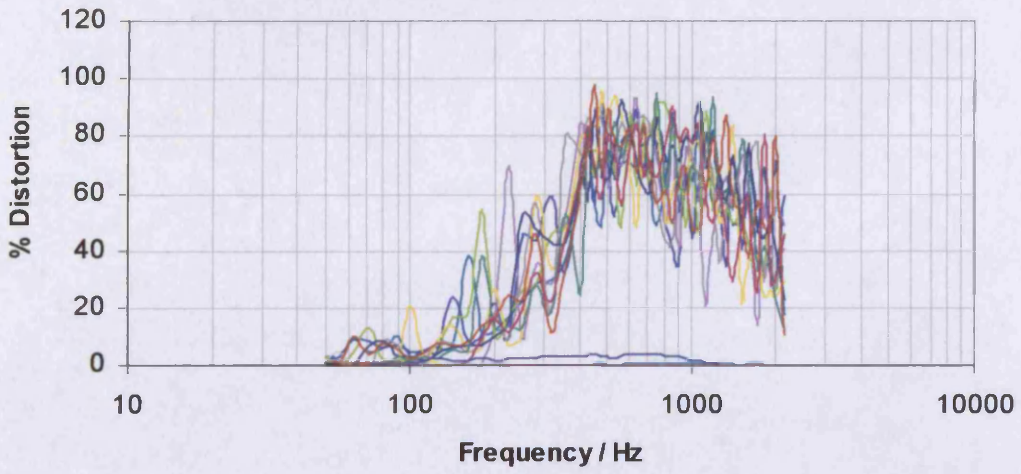


Figure 7.7 – Rub and buzz reject data

7.5 ANN TO EMULATE END OF LINE TEST

It was decided to focus training ANNs to emulate the rub and buzz test during this project as Harman/Becker indicated that improvements to the rub and buzz test were of more value than to the frequency response test.

Initial experiments involved training ANNs to emulate the output of the rub and buzz test i.e. to provide an output that indicates whether the loudspeaker should be accepted or rejected.

During ANN training the 48 point rub and buzz data pattern recorded during the end of line test was entered into the ANN input layer. Each data pattern was assigned a desired response according to whether it had been accepted or rejected by SOUNDCHECK^{TM} 4.1 during the end of line test. The training process is illustrated in figure 7.8. Each input layer neuron received one data point from the data pattern. The activations from the input layer neurons were then passed through the fully connected hidden layers and the output used to generate an error signal between the actual and desired response of the ANN. The error was then used to adjust the ANN weights using the back propagation algorithm.

Classification accuracy of the ANN was defined as the percentage of previously unseen data patterns that the network correctly classified as accepted or rejected loudspeakers, based on the results from the end of line test. Training and validation rms errors were also considered, however, a measure of how the network classified the loudspeaker data patterns was considered more descriptive for this application. The boundary for a correct classification was above 0 for an accepted loudspeaker and below 0 for a rejected loudspeaker.

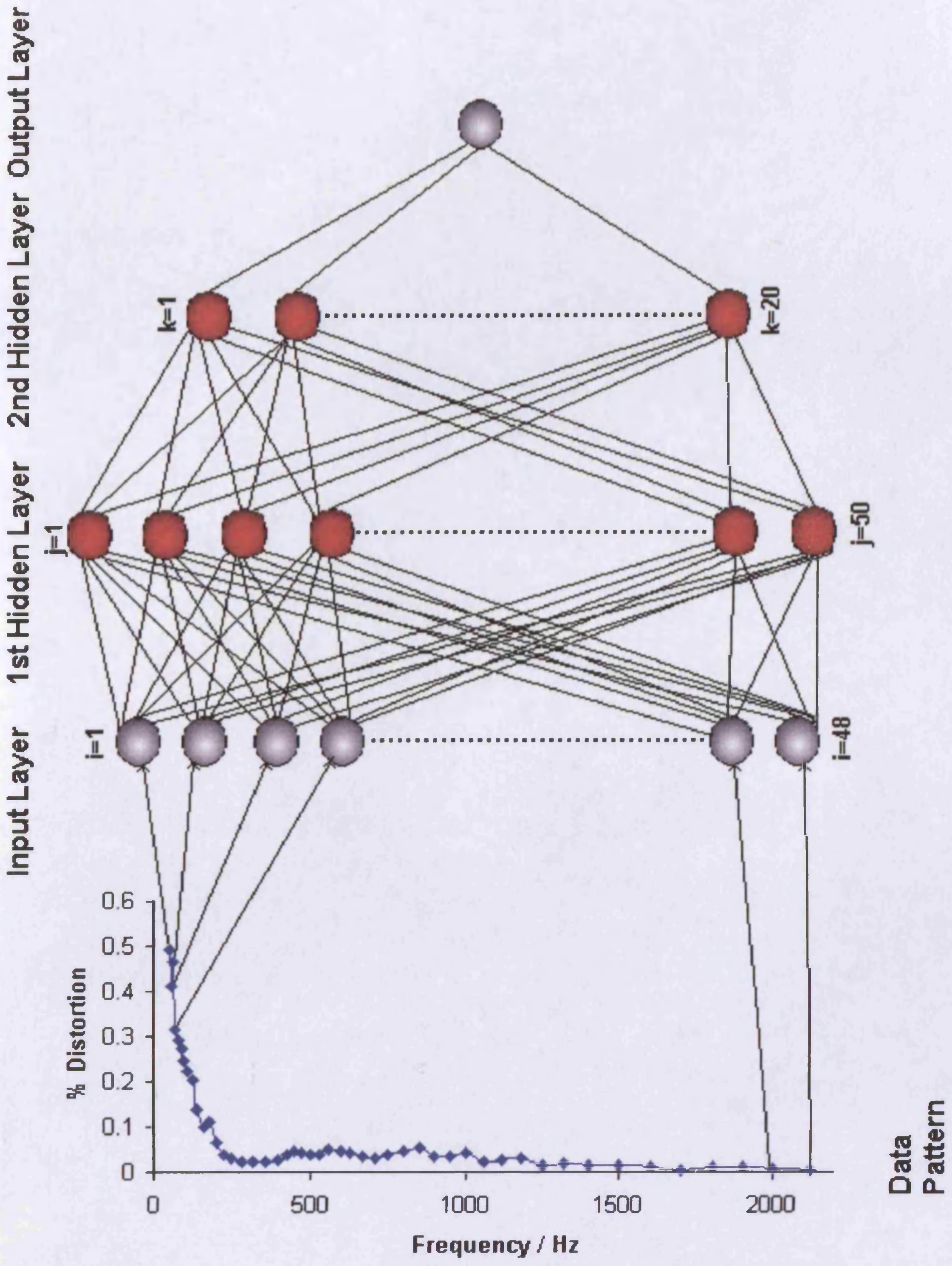


Figure 7.8 – Schematic of ANN training process

7.6 DATA PREPROCESSING

7.6.2 Elimination of gross rejects

It is generally agreed that preprocessing the data can improve the ANN performance (Haykin, 1999).

A significant proportion of the rejected loudspeaker data patterns consisted of random, high level noise that is easy to detect and indicates clearly that the loudspeaker should be rejected; these were categorised as gross rejects. It was decided to eliminate gross rejects from the data due to their heterogeneity; as it was anticipated that such training patterns may reduce the ability of the ANN to generalise and to classify loudspeaker data patterns that should be rejected but are only marginally dissimilar to a data pattern that should be accepted. An investigation was conducted to substantiate this. Figure 7.9 illustrates the response of an ANN trained with a data set containing accepted loudspeaker data patterns and gross reject data patterns, to a similar validation set.

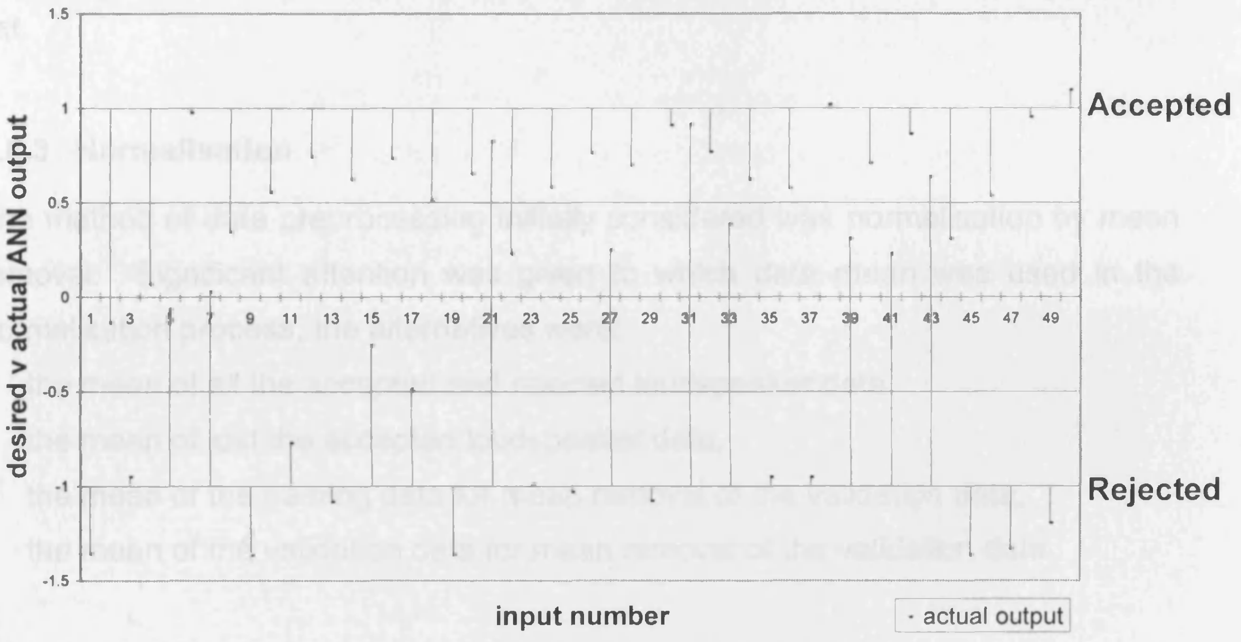


Figure 7.9 – Response of ANN trained with gross rejects

The classification accuracy of the ANN was 82 per cent, significantly lower than the ANNs trained with data sets with gross rejects eliminated. The rejected data patterns all contained at least one distortion value greater than 20 per cent, which may be considered an extreme case as this is a relatively high level of distortion.

However, an ANN trained with a data set with just 3 per cent of the rejected loudspeaker data patterns containing distortion values greater than 20 per cent and validated with a similar data set resulted in a classification accuracy of 93 per cent, which is 3.5 per cent lower than an ANN trained with all gross rejects with values greater than the limits + 6 standard deviations (σ) removed, (the default boundary during the ANN parameter investigation, see section 7.8), and 5.5 per cent lower than an ANN trained with gross rejects with values greater than the optimum boundary determined during the investigation (the limits + 10 σ).

It can therefore be concluded that gross rejects unfavourably influence the response of the ANN. The disparate nature of the gross rejects meant that during the training process the ANN could not converge upon a satisfactory solution to the classification problem.

The boundary used initially to categorise a gross reject was 6σ from the mean (μ). Any patterns that had values outside this boundary were deleted from the training set.

7.6.3 Normalisation

The method of data preprocessing initially considered was normalisation by mean removal. Significant attention was given to which data mean was used in the normalisation process, the alternatives were:

- the mean of all the accepted and rejected loudspeaker data,
- the mean of just the accepted loudspeaker data,
- the mean of the training data for mean removal of the validation data,
- the mean of the validation data for mean removal of the validation data.

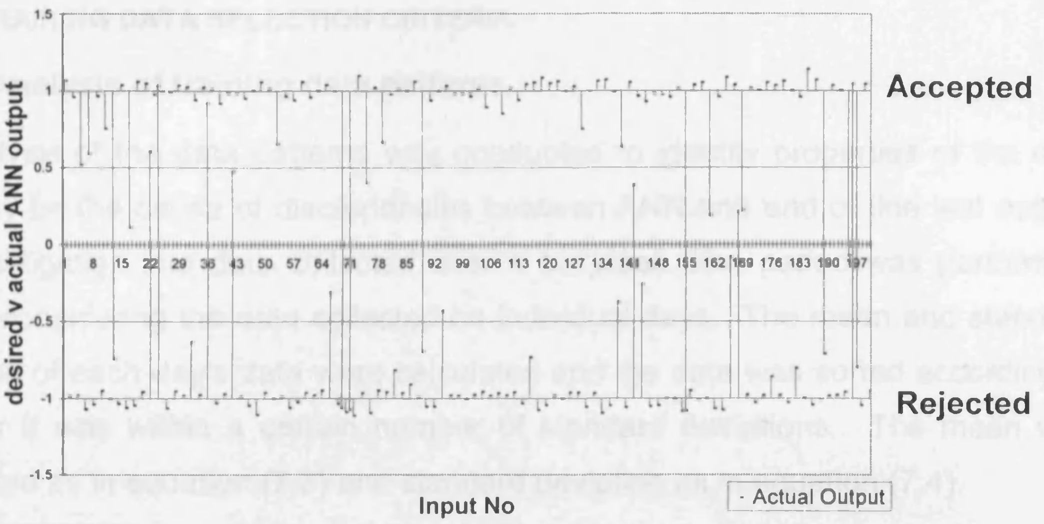
Including the gross reject data in the calculation of the mean artificially distorted the pattern of the accepted loudspeaker data, making it less recognisable. Including only the reject data within 6 standard deviations still led to a degree of distortion, hence it was determined that the most suitable method was to use the mean of the accepted loudspeaker data only. There was more correlation between the mean removed training data and the validation data with the training data mean removed than the validation data with the validation data mean removed, hence the former was used.

The mean of the accepted loudspeaker data was calculated for each frequency as in equation (7.3) and deducted from each data point at the corresponding frequency for all accepted and rejected data patterns in the training and validation sets.

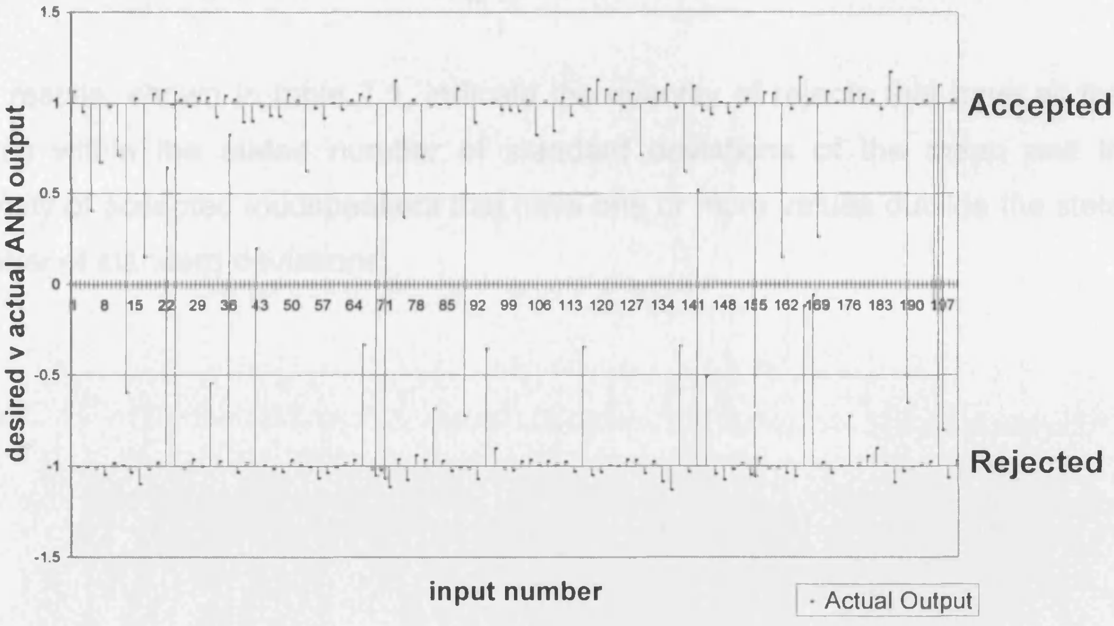
$$\bar{x} = \sqrt{\sum x_f^{\,2}} \qquad (7.3)$$

The data was subsequently sorted into a suitable format for training; alternating accepted and rejected data patterns. This reduced the possibility of bias weight alterations resulting from a group of very similar training patterns being shown to the network in succession.

Figure 7.10 shows two networks trained with data from the 99-100bm16 loudspeaker with the gross rejects removed, one with data that had been pre-processed and one with data that had not. The classification accuracy for the 'with processing' network was 92 per cent and for the 'no processing' network, 92.5 per cent. The validation rms errors were also almost identical. It was therefore concluded that removing the mean from the data patterns had little effect upon the ANN training results.



With preprocessing



No preprocessing

Figure 7.10 – Comparison of network results using data with and without preprocessing

7.7 TRAINING DATA SELECTION CRITERIA

7.7.2 Analysis of training data patterns

An analysis of the data patterns was conducted to identify properties of the data that may be the cause of discrepancies between ANN and end of line test output. An investigation into data collected over a six week time period was performed, initially considering the data collected on individual days. The mean and standard deviation of each day's data were calculated and the data was sorted according to whether it was within a certain number of standard deviations. The mean was calculated as in equation (7.3) and standard deviation as in equation (7.4).

$$\delta_f = \sqrt{\frac{\sum (x_f - \bar{x})}{n}} \tag{7.4}$$

where n is the total number of training patterns.

The results, shown in table 7.1, indicate the quantity of rejects that have all their values within the stated number of standard deviations of the mean and the quantity of accepted loudspeakers that have one or more values outside the stated number of standard deviations.

Table 7.1 – Summary of rub and buzz data variation for individual days

Date	Total accepted	Total rejected	Rejects within +/-1σ	Rejects within +/-2σ	Rejects within +3σ	Rejects within +6σ	Accepted outside+3σ	Accepted Outside +6σ
15.04.02	1486	283	0	1	45	94	154	17
16.04.02	1358	345	0	1	59	162	178	16
23.04.02	790	130	0	0	0	2	134	11
24.04.02	2120	487	0	7	67	122	361	52
30.04.02	3357	679	0	0	56	303	549	44
01.05.02	1915	387	0	4	92	193	253	31
08.05.02	999	132	0	0	1	50	145	22
09.05.02	1774	188	0	0	0	27	286	30
13.05.02	354	64	0	0	3	12	60	10
14.05.02	6158	500	0	0	20	183	990	97
15.05.02	810	46	0	0	0	7	116	21
21.05.02	3947	354	0	0	0	66	701	69
22.05.02	3040	166	0	0	0	3	522	48
23.05.02	757	135	0	0	0	8	132	12
25.05.02	257	6	0	0	0	1	51	4
27.05.02	797	80	0	0	0	0	124	11
28.05.02	3546	528	0	0	69	202	552	60
29.05.02	5251	323	0	0	7	66	841	86

Table 7.2 - Summary of rub and buzz data variation for all data

Date	Total Accepted	Total Rejected	Rejects within +/-1σ	Rejects within +/-2σ	Rejects within +3σ	Rejects within +6σ	Accepted outside +3σ	Accepted outside +6σ
15.04.02 to 29.05.02	38716	4833	0	29	368	1543	6661	624

This analysis revealed significant problems with the supplied data:

- On each day of sampling there were a significant number of accepted loudspeaker data patterns with values that were outside 6σ from the mean.
- On several days there were rejects that were within 3σ from the mean and on four days within 2σ.

After each day had been analysed individually, all the data were amalgamated and studied as a whole, and the results are shown in table 7.2. It was anticipated that taking a larger population would result in a reduction in outlying accepted data, as the mean and standard deviation would be more representative and would eliminate any daily variations that could have affected the analysis of data from individual days. However this was not the case, the number of accepted loudspeakers outside 6σ was actually greater than the sum of the individual days. This was also true for the rejects within 2 and 3σ. This may be explained by the fact that on some days, production may have corresponded quite closely with the population that was used to formulate the test limits.

An explanation for the extraneous data may be derived from the method used to determine the test limits. When a new loudspeaker comes into production 3 initial production runs are performed to determine the end of line test limits. Between 50 and 200 units are constructed and tested in order to generate test limits. The rub and buzz test limits are set to a level such that loudspeakers with audible distortion are rejected. The limits may be further modified when a production trend is identified that may have distortion levels above the limits but which is inaudible (Anthony, 2003). Examination of the test limit history revealed nine alterations over the data collection period 15.04.02 to 29.05.02 for loudspeaker 99-100bm16, all of which were to raise the limits, resulting in loudspeakers that would previously have been rejected under the former limits being accepted under the amended limits. Also, a rejected loudspeaker may have had very few or even just a single value above the test limits, which may not necessarily have been significantly higher than the limits. Under these circumstances, the difference between an accepted and rejected data pattern would be negligible. Therefore, if the test limits were raised to a level where the previously rejected loudspeaker would be accepted, the rejected data pattern could be almost identical to, or even have lower values than, an accepted data pattern recorded under the subsequent test limit version. Attempting to train an ANN with this conflicting data would inevitably lead to discrepancies in ANN output when compared to the end of line test results.

The reject data also contained retest data. When the end of line test results in a rejection, the operator retests the loudspeaker to ensure the fault is permanent, as some loudspeakers exhibit high levels of distortion in the first test, caused by misalignment of components, that is rectified after the first operation. Although the test rig includes a retest button that should be used instead of the conventional test button when the loudspeaker fails the first test, a significant number of operators do not use it, thus resulting in data derived from the same loudspeaker being written to the reject file twice or even three times.

Increasing the test limits would result in an increase in the mean and the standard deviation of the accepted loudspeaker data patterns collected. During the initial training data selection, the mean and standard deviation of the accepted loudspeaker sample (comprised of data collected over several different test limits versions) were calculated and data patterns (accepted and rejected) selected for training according to their inclusion in a range relating to the mean plus a certain number of standard deviations, as illustrated in figure 7.11. In most cases the data selection criteria was all values below the mean + 3σ for accepted loudspeaker data patterns and all values below the mean + 6σ for rejected loudspeaker data patterns.

The overall sample mean and standard deviation were likely to be lower than those of the data patterns collected under the test limit versions with higher values, resulting in these patterns being rejected under the training data selection criteria. Hence, what may be argued as perfectly valid training data was being excluded from the ANN training process.

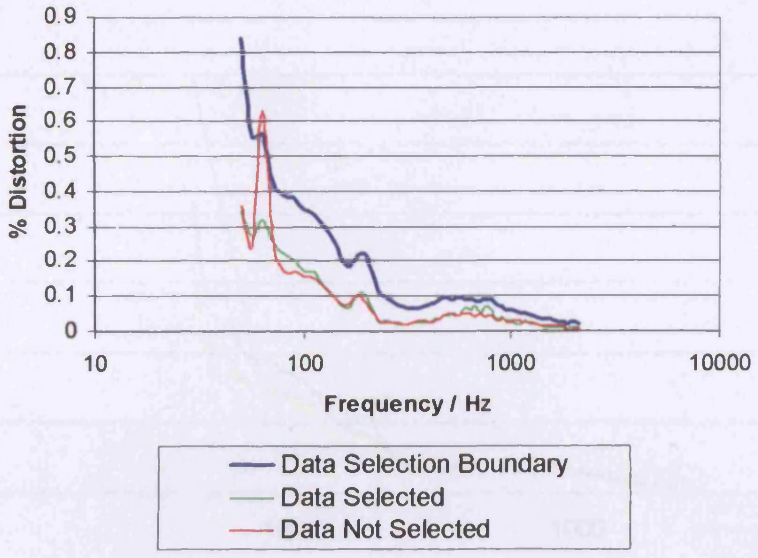


Figure 7.11 – Data selection process

7.7.3 Modified training data selection criteria

It was anticipated that training the ANN with data collected under a single version of test limits would decrease the number of loudspeaker data patterns the ANN classified differently to the EOL test, as the scenarios described above would not exhibit. It was understood that the test limits were not changed between 19.06.02 and 13.10.02. However, data collected in this period provided similar classification results to previous data. Further investigation was therefore required.

An evaluation of the test limits between 19.06.02 and 13.10.02 determined that the training data selection criteria possessed little correlation to the actual end of line test limits, as they assumed the test limits were in the region of μ+3σ. It revealed that the test limits values corresponded to the original assumption of μ+3σ only in a narrow range of frequencies; the majority of values were far higher than this, in the region of μ+ 6σ, and even higher in many instances as illustrated in figure 7.12.

The training data selection criteria used previously incorporated the assumption that the test limits were in the region of μ+3σ, hence only accepted loudspeaker data patterns with all values less than μ+3σ were selected. Rejected loudspeaker data patterns were selected if all their values were below μ+6σ. As previously

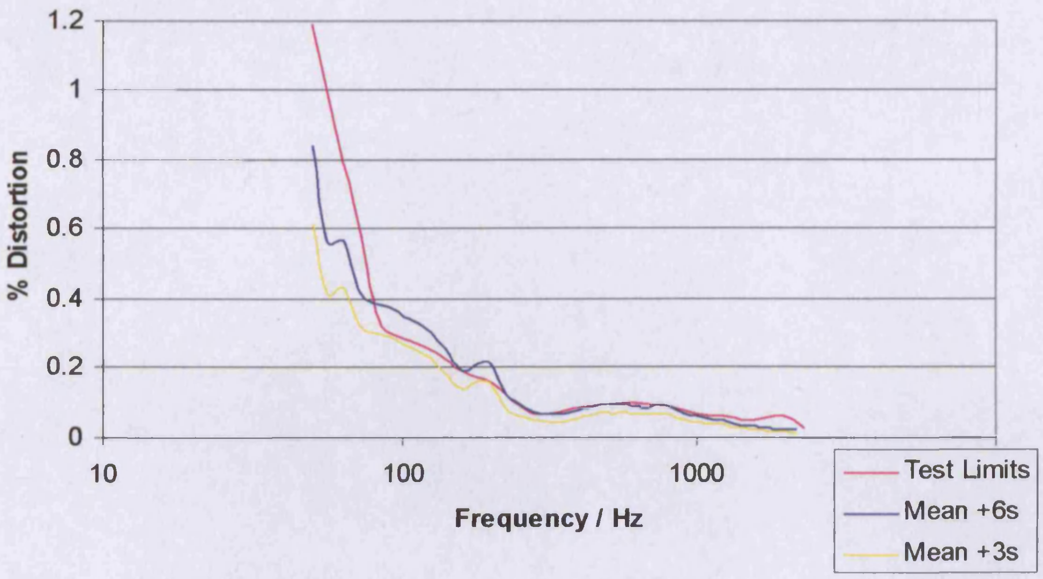


Figure 7.12 – Comparison of test limit values to $\mu+3\sigma$ and $\mu+6\sigma$

discussed, if only a few values were above the limits, and if these values were only marginally above the limits (knowledge of the limits inferred that these values must have been extremely close to the limits, as the limits were in the region of the selection criterion for rejected loudspeaker patterns), the problem of decipherability of data patterns by the ANN would have been compounded as the selected data for accepted and rejected loudspeakers would have existed in common regions.

This therefore necessitated a modification to the training data selection criteria. As the actual test limits were now available it was decided to link the selection criteria directly to the limits. All accepted loudspeaker data patterns were included and rejected loudspeaker data patterns with values less than the limits plus a certain number of standard deviations were selected. In most cases the number of standard deviations was 6. This had a significant effect on the classification results of the ANN; classification correlation between the ANN and the EOL test increased from 91 to 96 per cent. The ANN validation results are shown in figure 7.13.

The initial selection criteria implemented, as described in section 7.7 were extremely unsuitable as all the training data, both accepted and rejected loudspeaker patterns, would have had extremely similar values, making many data patterns practically undecipherable from each other.

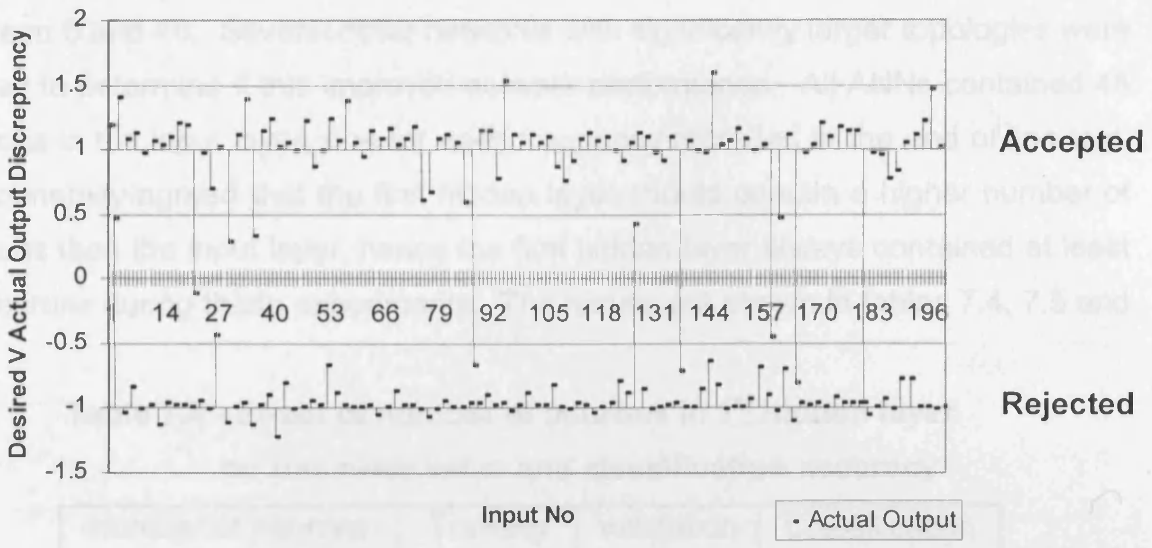


Figure 7.13 – Network results using revised data selection criteria

7.8 ANN OPTIMISATION

An investigation to determine the optimum parameter values for the ANN trained to emulate the rub and buzz end of line test was conducted. The default ANN parameter values are shown in table 7.3.

Table 7.3 – Default parameter values

Parameter	Default value
Topology	48-50-20-1
Epoch number	1×10^4
Training set size	2×10^3
Validation set size	2×10^2
Learning rate	1×10^{-5}
Momentum	5×10^{-2}
Data boundary	limits + 6σ

7.8.2 Topology

An investigation was conducted to identify whether or not an optimum topology for classification accuracy of the ANN existed. Tests were performed with the number of neurons in the first hidden layer varied between 50 and 100 and those in the second hidden layer kept constant at 20. The number of neurons in the first hidden layer was then kept constant at 50 and those in the second hidden layer varied between 5 and 40. Several other networks with significantly larger topologies were trained to determine if this improved network performance. All ANNs contained 48 neurons in the input layer, one for each frequency recorded in the end of line test. It is generally agreed that the first hidden layer should contain a higher number of neurons than the input layer, hence the first hidden layer always contained at least 50 neurons during these experiments. The results are shown in tables 7.4, 7.5 and 7.6.

Table 7.4 - Effect of number of neurons in 1st hidden layer on rms error value and classification accuracy

Number of neurons in 1st hidden layer	Training rms error	Validation rms error	Classification accuracy / %
50	0.629	0.495	96.5
60	**0.636**	**0.472**	**98.0**
70	0.645	0.486	97.0
60	0.629	0.501	95.0
90	0.635	0.477	97.0
100	0.629	0.486	96.5

Table 7.5 - Effect of number of neurons in 2nd hidden layer on rms error value and classification accuracy

Number of neurons in 2nd hidden layer	Training rms error	Validation rms error	Classification accuracy / %
5	**0.645**	**0.473**	**98.0**
10	0.640	0.476	97.5
20	0.645	0.495	96.5
30	0.645	0.499	95.5
40	0.645	0.499	95.0

Table 7.6 – Effect of employing larger topologies on rms error value and classification accuracy

Number of neurons in 1st hidden layer	Number of neurons in 2nd hidden layer	Training rms error	Validation rms error	Classification accuracy / %
100	**25**	**0.642**	**0.498**	**95.0**
100	50	0.644	0.504	95.0
100	75	0.645	0.505	95.0

Figure 7.14 shows the response of the best performing network to previously unseen data. The chart shows the ANN output for each data pattern plotted as a rectangular marker, which is connected to the desired output for that data pattern by a line. Therefore, the longer the line, the larger the error between ANN output and desired output.

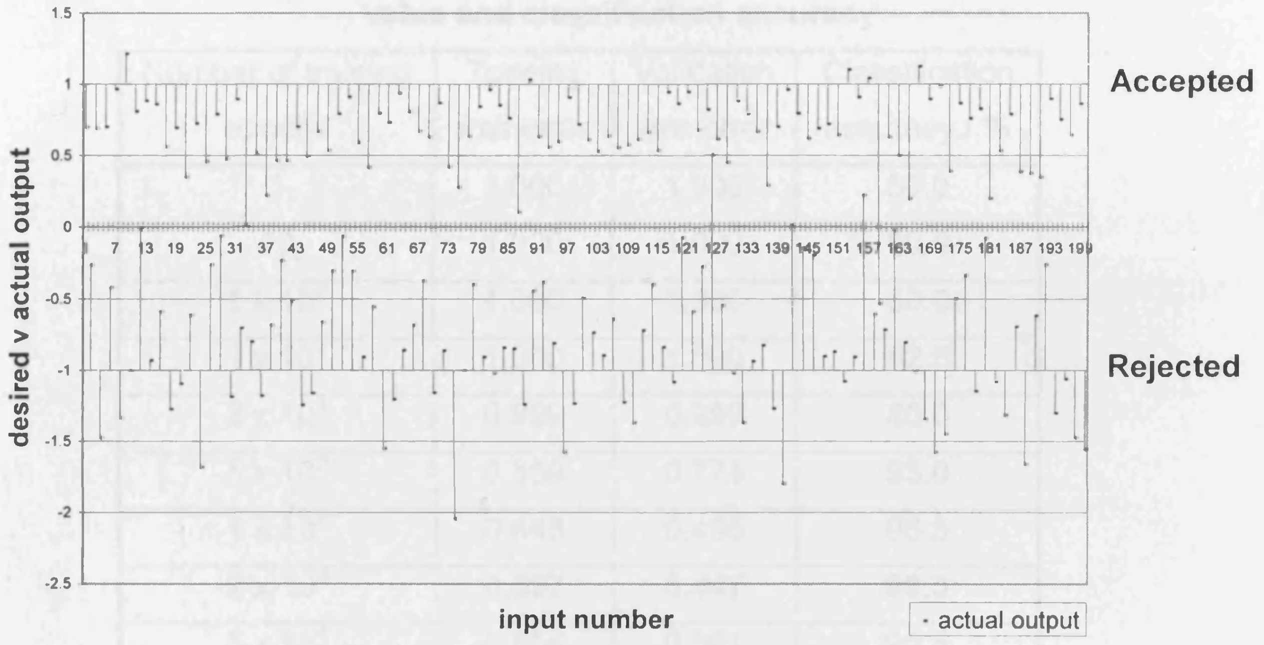


Figure 7.14 – ANN output with 48-60-20-1 topology

It was determined that a considerable variation in the number of neurons in the first and second hidden layers between two networks caused very little difference in the validation rms error or the classification accuracy. In many cases the increased training time required for larger topology networks was unjustified, for example, despite having more than twice as many neurons, the 48-100-50-1 (the number of neurons in the input layer, first hidden layer, second hidden layer and output layer respectively) network had a lower classification accuracy (95 per cent) than the 48-50-20-1 network (96.5 per cent). As discussed in section 5.1.2, this may be due to the parallelism of the network and that ANN performance is related to the compatibility of the ANN architecture with the system being modelled.

7.8.3 Training epochs

Increasing the number of training epochs improves the probability of the ANN learning the correct association between each data pattern and the desired output, as the weights are adjusted to incorporate the association between each data pattern and desired output combination a greater number of times. During this experiment the number of training epochs was varied between 1 and 5 x 10^4. The results are shown in table 7.7.

Table 7.7 – Effect of epoch number on rms error value and classification accuracy

Number of training epochs	Training rms error	Validation rms error	Classification accuracy / %
1	1.000	1.000	50.0
10	1.000	1.000	50.0
1 x 10^2	1.000	1.000	50.0
1 x 10^3	1.000	1.000	62.5
2 x 10^3	0.999	0.999	85.0
5 x 10^3	0.859	0.774	93.0
1 x 10^4	0.645	0.495	96.5
2 x 10^4	**0.597**	**0.447**	**98.5**
5 x 10^4	0.504	0.367	95.5

Figures 7.15 and 7.16 show the response to previously unseen data of the best performing network in terms of classification accuracy and validation rms error respectively. It can be seen that the ANN trained over 5 x 10^4 epochs has a lower discrepancy between actual and desired output for the majority of the validation

ıta patterns, hence the lower validation rms error than the ANN trained over 2 x)^4, however the number of training patterns outside the correct classification ›undary was greater, resulting in the lower classification accuracy. As the assification accuracy of the ANN is of more significance to its application, the end line test, the ANN trained over 2 x 10^4 epochs was determined to be the best ırforming ANN overall.

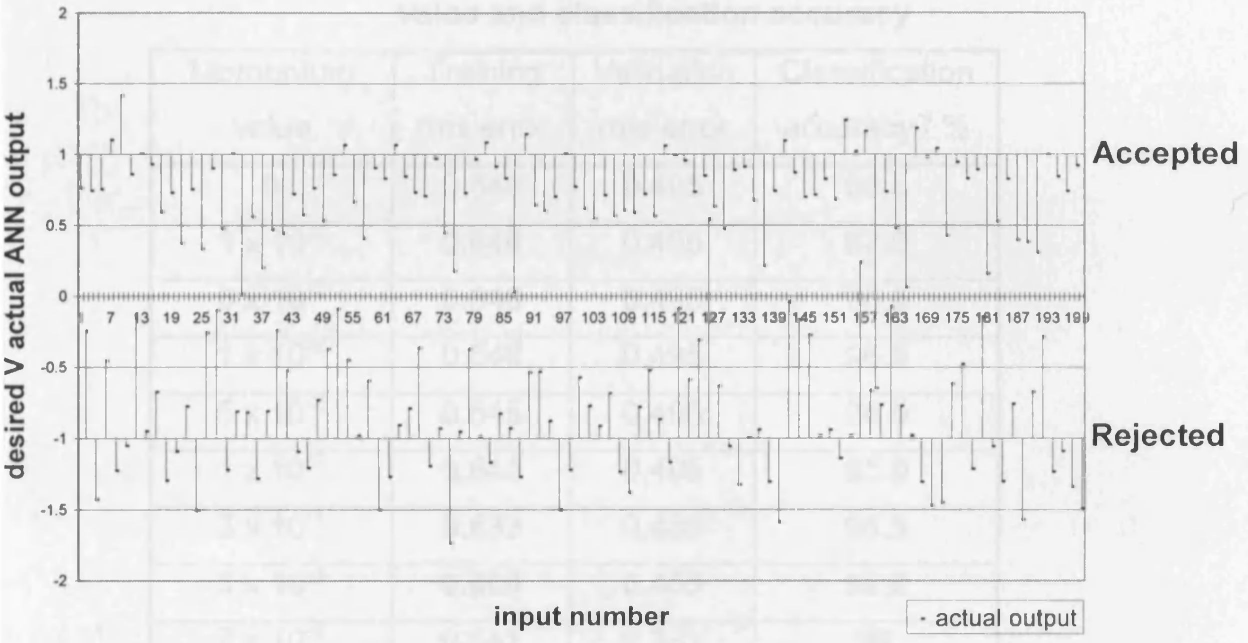


Figure 7.15 – ANN output after training over 2 x 10^4 epochs

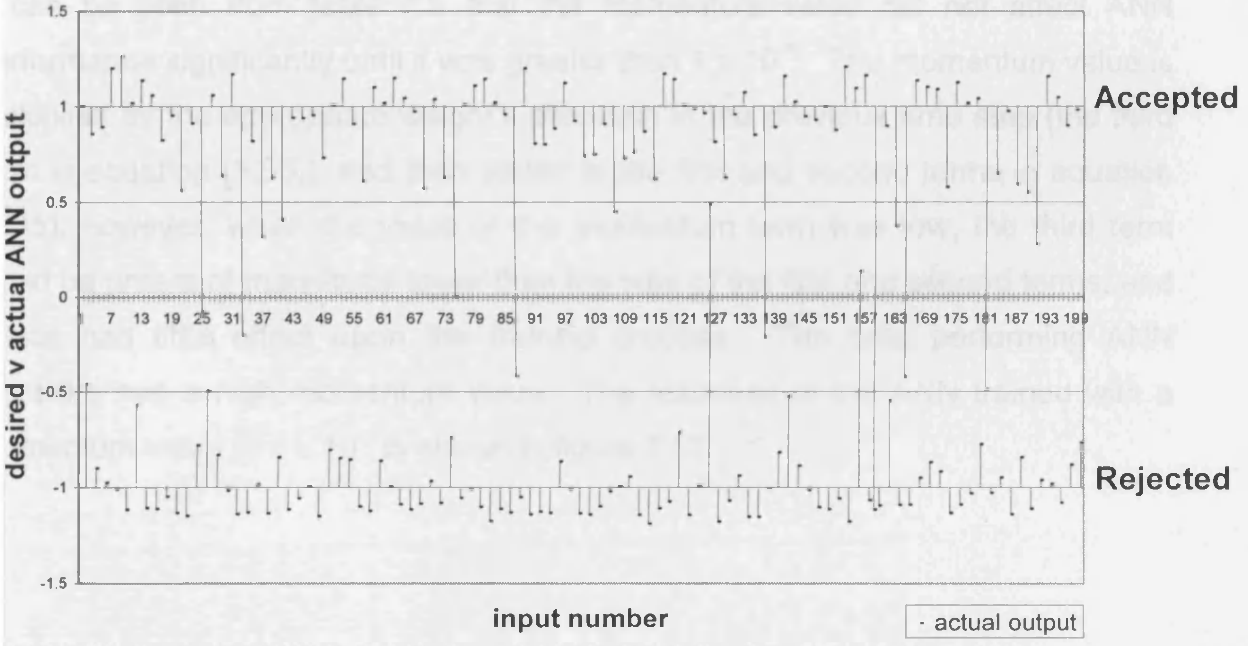


Figure 7.16 – ANN output after training over 5 x 10^4 epochs

7.8.4 Momentum term

As previously discussed, the momentum term determines the degree to which the previous iteration influences the change in the weight values during the current iteration. The results of experiments where the momentum term was varied between 1 x 10^{-3} and 5 x 10^{-1} are shown in table 7.8.

Table 7.8 – Effect of momentum value on rms error value and classification accuracy

Momentum value	Training rms error	Validation rms error	Classification accuracy / %
0	0.646	0.495	96.5
1 x 10^{-3}	0.646	0.495	97.0
5 x 10^{-3}	0.646	0.495	96.5
1 x 10^{-2}	0.646	0.495	96.5
5 x 10^{-2}	0.645	0.495	96.5
1 x 10^{-1}	0.643	0.495	95.0
3 x 10^{-1}	0.633	0.488	95.5
5 x 10^{-1}	**0.606**	**0.453**	**98.0**
7 x 10^{-1}	0.541	0.381	96

It can be seen from table 7.8 that the momentum value did not affect ANN performance significantly until it was greater than 1 x 10^{-1}. The momentum value is multiplied by the appropriate weight's alteration in the previous time step (the third term in equation (3.35)), and then added to the first and second terms in equation (3.35), however, when the value of the momentum term was low, the third term could be orders of magnitude lower than the sum of the first and second terms, and hence had little effect upon the training process. The best performing ANN therefore had a high momentum value. The response of the ANN trained with a momentum value of 5 x 10^{-1} is shown in figure 7.17.

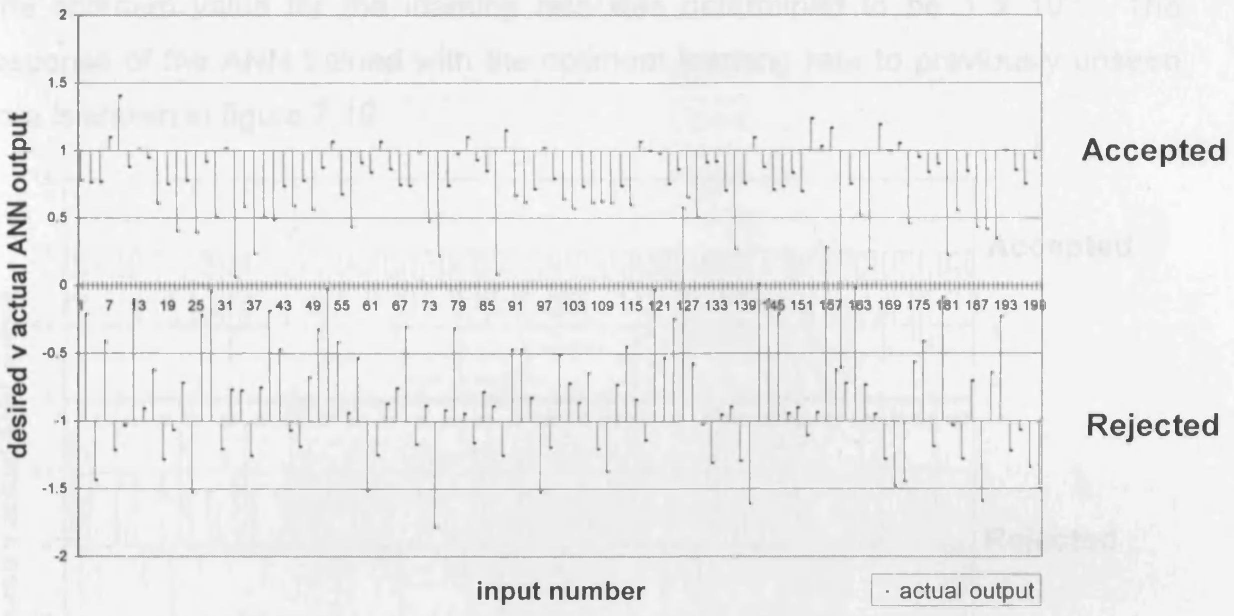


Figure 7.17 – Response of ANN trained with momentum value of 5 x 10^{-1} to previously unseen data

7.8.5 Learning rate

The learning rate determines the influence of the error generated during the forward pass on the weight alteration of the backward pass. It also determines the rate of convergence of the ANN to the optimum solution and affects the stability of this convergence. Tests were conducted where the learning rate was varied between 1 x 10^{-6} and 1 x 10^{-1}, the results are shown in table 7.9.

Table 7.9 – Effect of learning rate on rms error and classification accuracy

Learning rate	Training rms error	Validation rms error	Classification accuracy / %
1 x 10^{-6}	1.000	1.000	62.5
1 x 10^{-5}	**0.645**	**0.495**	**96.5**
1 x 10^{-4}	0.497	0.343	96.0
1 x 10^{-3}	0.433	0.379	95.5
1 x 10^{-2}	0.444*	0.485*	94.0*
1 x 10^{-1}	0.449*	0.435*	93.5*

*The training process reached a local minimum

The optimum value for the learning rate was determined to be 1 x 10^{-5}. The response of the ANN trained with the optimum learning rate to previously unseen data is shown in figure 7.18.

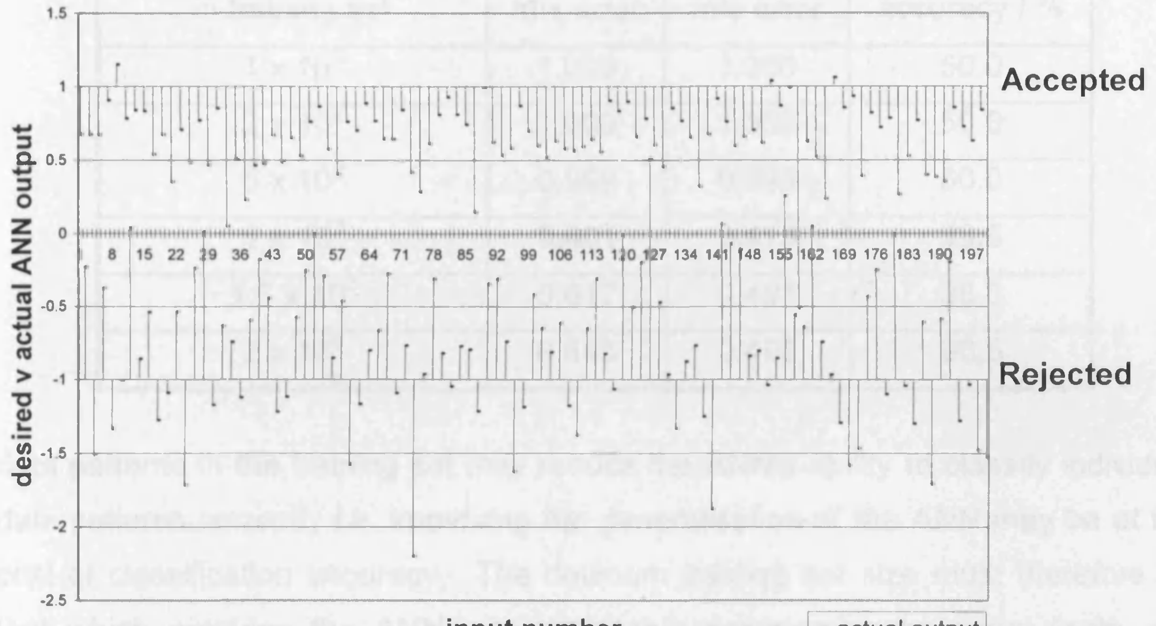


Figure 7.18 – Response of ANN trained with learning rate of 1 x 10^{-5} to previously unseen data

7.8.6 Training set size

There is a high probability that a larger training set will contain a wider range of loudspeaker data patterns, which will result in a wider range of the network's a posteriori knowledge and thus improve the probability the network will be able to classify a previously unseen data pattern correctly. Tests were conducted with the training set varied between 1 x 10^3 and 6 x 10^3. The upper limit was dictated by the number of available data patterns. The results are shown in table 7.10.

The optimum training set size was determined to be 1 x 10^3. This may be due to a high correlation between the data contained in the 1 x 10^3 data set and the validation data. The extra data patterns contained in the larger training sets may be significantly different to those contained in the validation set, hence the ANN is trained away from the optimum solution for the validation data. These results identify a possible limitation of ANNs in this application; increasing the range of

Table 7.10 – Effect of training set size on rms error value and classification accuracy

Number of data patterns in training set	Training rms error	Validation rms error	Classification accuracy / %
1×10^2	1.000	1.000	50.0
2×10^2	1.000	1.000	50.0
5×10^2	0.999	0.999	50.0
1×10^3	**0.601**	**0.479**	**98.5**
1.5×10^3	0.617	0.491	98.0
2×10^3	0.645	0.495	96.5

data patterns in the training set may reduce the ANN's ability to classify individual data patterns correctly i.e. improving the generalisation of the ANN may be at the cost of classification accuracy. The optimum training set size must therefore be that which provides the ANN with a suitable compromise between depth and breadth of knowledge. This may require the periodic retraining of the ANN in order to incorporate new production trends that develop over time and eliminate those that are no longer occurring. The output of the best performing ANN is shown in figure 7.19

Figure 7.19 – Response of ANN trained with training set size of 1×10^3 to previously unseen data

7.8.7 Training data selection criteria

A previous investigation (see section 7.7) determined that eliminating gross rejects from the training set improved the classification accuracy of the ANN. The objective of this investigation was to determine the optimum selection criteria for the training data set. Training sets were compiled using selection criteria that increased in increments of the standard deviation from the mean of the accepted loudspeaker data patterns. The results are shown in table 7.11.

Table 7.11 – Effect of training data selection criteria on rms error and classification accuracy

Data selection boundary	Training rms error	Validation rms error	Classification accuracy / %
no boundary	0.475	0.523	93.0
limits + 4σ	0.643	0.686	86.5
limits + 6σ	0.645	0.495	96.5
limits + 8σ	0.661	0.496	98.5
limits + 10σ	**0.673**	**0.492**	**98.5**
limits + 12σ	0.677	0.585	92.0
limits + 15σ	0.674	0.704	87.5
limits + 20σ	0.655	0.648	89.0

The output of the best performing ANN is shown in figure 7.20.

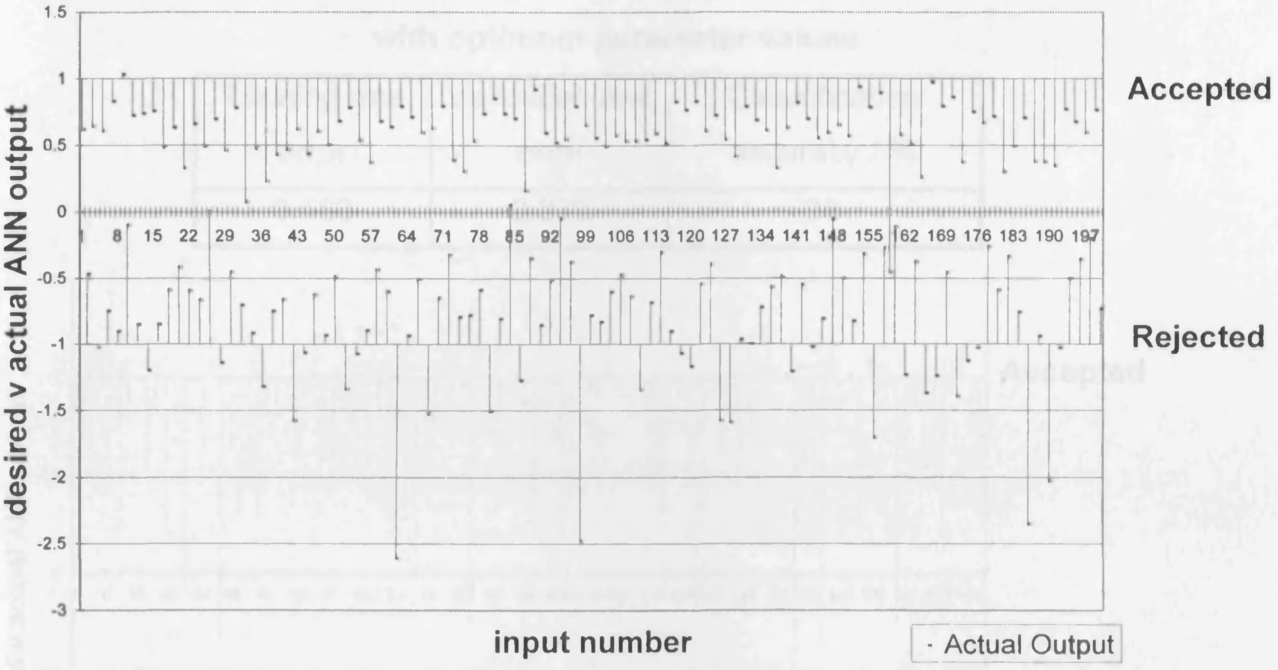


Figure 7.20 – Response of ANN trained with training set selection of limits + 10σ to previously unseen data

7.8.8 Optimum parameter values

The optimum parameter values determined in the preceding investigation are shown in table 7.12. The training results and the response of an ANN trained with these parameter values to previously unseen data are shown in table 7.13 and figure 7.21 respectively.

Table 7.12 – Optimum parameter values

Parameter	Optimum parameter value
Topology	48-60-5-1*
Epoch number	2×10^4
Training set size	1×10^3
Validation set size	2×10^2
Learning rate	1×10^{-5}
Momentum	5×10^{-1}
Data boundary	limits + 10σ

*The optimum number of neurons in the first hidden layer was 60 and in the second 5, hence the optimum topology was taken to be a combination of these results.

Table 7.13 – Training results for ANN trained with optimum parameter values

Training rms error	Validation rms error	Classification accuracy / %
0.483	0.379	96

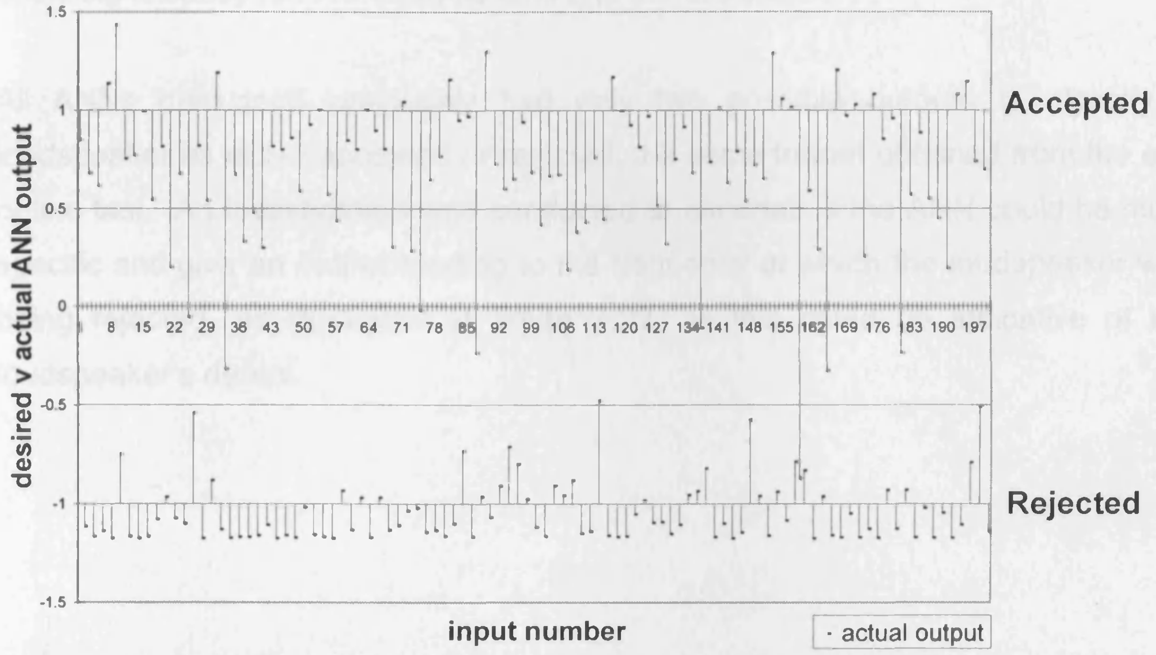


Figure 7.21 - Response of ANN trained with optimum parameter values to previously unseen data

As experienced with the multi layer perceptron ANN trained to model the loudspeaker transfer function, the response of the ANN trained with the optimum parameters determined independently of each other was not the optimum response (see section 5.3), as several ANNs tested during this investigation had classification accuracies of 98.5 per cent (see sections 7.8.3, 7.8.6 and 7.8.7).

7.8.9 Training times

Training times with a 1.5GHz PC varied between 1 and 8 hours per network for 200 to 2000 training patterns respectively and 20 000 iterations. The training time for the ANN with default parameters was 2 hours.

7.9 ANN TO DETERMINE FREQUENCY REGION IN WHICH FAILURE OCCURS

The ANN structures determined to classify a loudspeaker data pattern as accepted or rejected were developed into networks that were trained to distinguish between loudspeakers that have audible distortion in different frequency bands, in an attempt to meet the project objective of intelligently analysing faults. This process is currently carried out by human operatives, hence a successfully trained ANN could significantly reduce analysis time and therefore cost.

All ANNs discussed previously had only two possible outputs to classify a loudspeaker as either accepted or rejected; the same format obtained from the end of line test. An investigation was conducted to ascertain if the ANN could be more specific and give an output relating to the frequency at which the loudspeaker was being rejected, as illustrated in figure 7.22, as this could be indicative of the loudspeaker's defect.

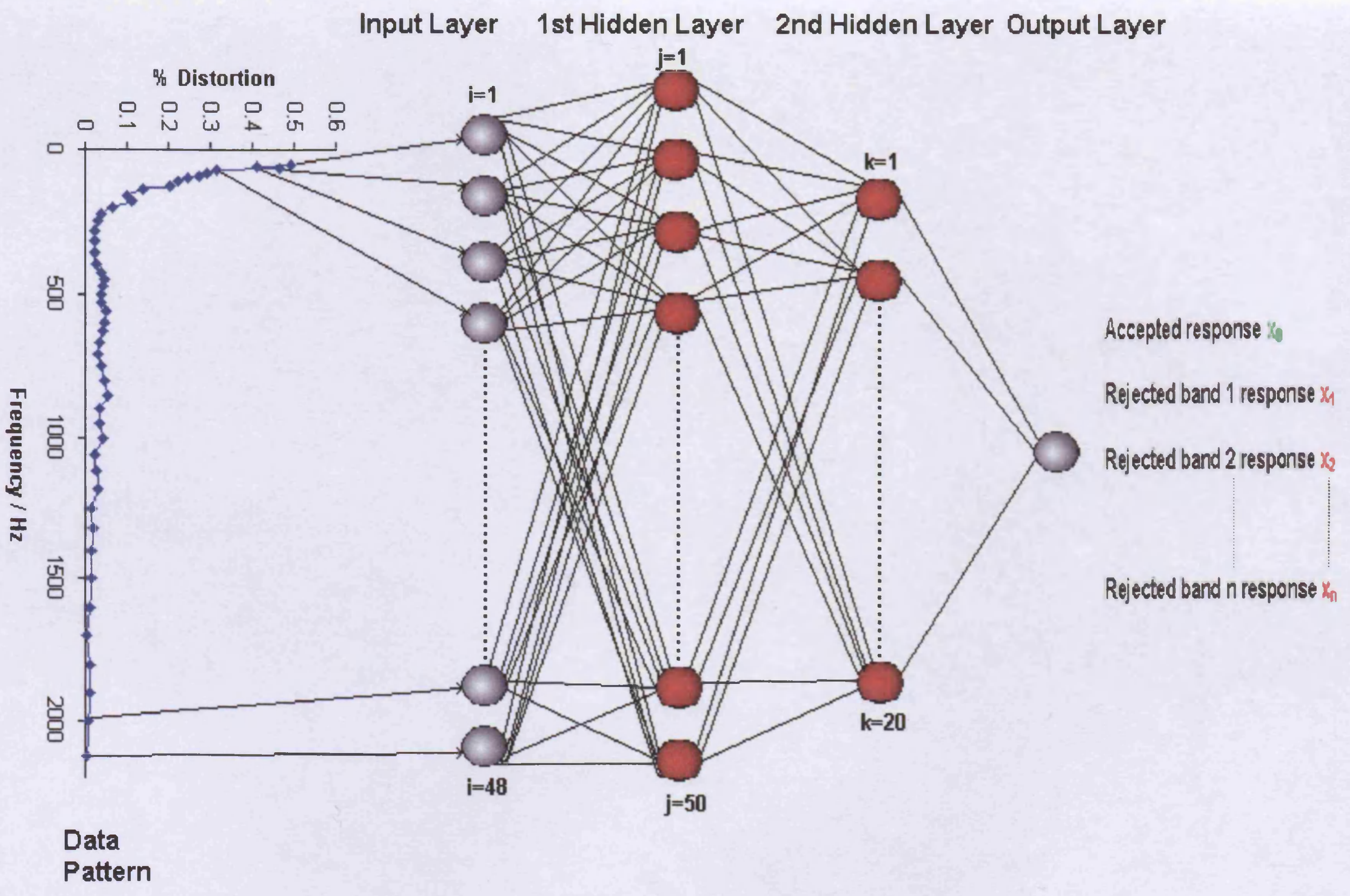


Figure 7.22 – ANN with desired response relating to frequency band in which loudspeaker is rejected

7.9.2 Three frequency bands

Initial experiments involved splitting the frequency range into three equal bands and training the ANN with rejected loudspeaker data patterns that had values above the limits in only one of these frequency bands. A C++ program was developed to determine at which frequencies the loudspeaker data patterns had values above the limits. This simply compared each distortion level to the limits and assigned a value of 1 to a variable, (which was otherwise 0), if the value was greater than the limit value for that frequency. There were three variables which corresponded to the three frequency bands. The data pattern was then exported to a file with the three variables assigned to it. The data patterns could then be sorted according to the frequency bands in which they were rejected. Appendix 8 contains the complete program code.

The data patterns that had values above the limits in just one frequency band were then extracted and assigned a desired output value according to its rejection band. The rejected data patterns were then combined with accepted data patterns in preparation for ANN training.

Figure 7.23 shows the network results where the three frequency bands have been given different desired output values. In this case the desired outputs were -0.9 for loudspeakers rejected in the frequency band 50 - 280 Hz, -0.6 for loudspeakers rejected between 315 - 850 Hz and -0.3 for those rejected between 900 – 2120 Hz. Although it is difficult to quantify the exact correlation between ANN output and desired output there is clearly a distinction between accepted and rejected loudspeakers in the majority of cases. There is also good correlation between the desired output value for loudspeakers rejected in the respective frequency bands and the actual ANN output, which could provide information regarding the fault present in the rejected loudspeaker.

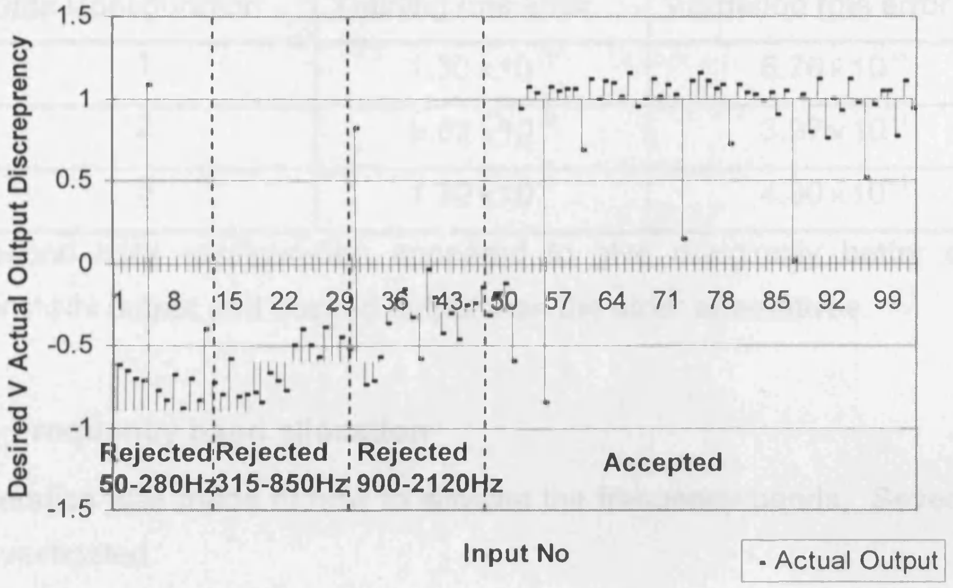


Figure 7.23 – 3 Frequency band network results

7.9.3 Five frequency bands

The frequency band concept was developed further to incorporate five frequency bands. Experiments were conducted to ascertain optimum data configuration, frequency band allocation and desired output values.

7.9.3.i Data configuration

Several data configurations were used, incorporating the same data to allow direct comparison, including:

1. alternating all rejected data patterns with an accepted data pattern,
2. grouping one data pattern from each rejection band together, followed by one accepted data pattern,
3. an equal number of rejected and accepted data patterns in random order.

The previously established methods of preprocessing were applied to the training sets and ANNs trained with each data configuration. The resulting training and validation rms errors are shown in table 7.14.

Table 7.14 – Comparison of data configurations

Data Configuration	Training rms error	Validation rms error
1	1.30×10^{-1}	5.76×10^{-1}
2	9.63×10^{-2}	3.97×10^{-1}
3	1.30×10^{-1}	4.30×10^{-1}

The second data configuration appeared to give marginally better correlation between ANN output and desired output than the other alternatives.

7.9.3.ii Frequency band allocation

Consideration was made of how to allocate the frequency bands. Several options were investigated:

1. allocating an equal number of frequency points to each band, hence in the case of loudspeaker 99-100bm16, 3 frequency bands contained 10 frequencies and 2 contained 9 frequencies, totalling 48 and satisfying the criteria as closely as possible,
2. making the lower frequency bands narrower than the high frequency bands, as a large proportion of rejected loudspeakers have values outside the limits at low frequencies,
3. allocating an equal number of the available data patterns to each frequency band, thus the frequency regions where a large proportion of the loudspeakers show defects will have smaller bands, hence placing more emphasis on these regions,
4. the above technique was repeated with the exception that the frequency bands were decided after the data patterns with extreme values (gross rejects) had been removed.

The results are displayed in table 7.15.

Table 7.15 – Comparison of frequency band allocation methods

Frequency Band Allocation Method	Training rms error	Validation rms error
1	7.88×10^{-2}	3.25×10^{-1}
2	9.60×10^{-2}	5.27×10^{-1}
3	6.79×10^{-2}	2.50×10^{-1}
4	1.09×10^{-1}	7.20×10^{-1}

The third allocation method gave the best correlation between ANN output and desired output. The frequency bands are shown in table 7.16.

Table 7.16 – Frequency bands for allocation method 3

Frequency Band	Frequency Content / Hz
1	50 – 200
2	224 – 280
3	315 – 450
4	475 – 650
5	670 – 2120

7.9.3.iii Desired ANN output values

An investigation into the output values assigned to the various frequency bands was also undertaken. A wide range of values were employed including large and small values with equal increments and, taking into account the observation that accepted loudspeaker patterns were more clearly defined in these experiments, the accepted patterns were given output values significantly different to the rejected loudspeaker patterns in an attempt to distinguish them more explicitly. Several examples of values used are shown in table 7.16.

Table 7.16 – ANN desired output values

	Rejected 50-180Hz	Rejected 200-250Hz	Rejected 280-425Hz	Rejected 450-630Hz	Rejected 670-2120Hz	Accepted 50-2120Hz
1	-5	-4	-3	-2	-1	1
2	12	10	8	6	4	2
3	-5	-4	-3	-2	-1	100

ANNs trained with the first set of output values resulted in a validation rms error of 3.97×10^{-1}, and the second set 2.50×10^{-1}. Giving accepted loudspeaker data patterns significantly different desired output values did not result in an improvement in their decipherability from rejected loudspeakers, the validation rms error in this case was 6.93×10^{-1}.

7.9.3.iv Optimum 5 frequency band ANN

The optimum data configurations, allocation of frequency bands and output values were determined to be grouping one data pattern from each rejection band together, followed by one accepted data pattern with output values of equal increments between 2 and 12, where the frequency regions had been allocated by assigning an equal number of the available data patterns to each frequency band. The ANN results when trained under these conditions are shown in figure 7.24. Figure 7.24a shows the ANN response to the training data and illustrates that the ANN has learned the data well. Figure 7.24b shows the ANN response to previously unseen data, and although there is an obvious trend in the output, determination of the exact frequency band in which the loudspeaker has audible distortion would be difficult in a significant proportion of cases.

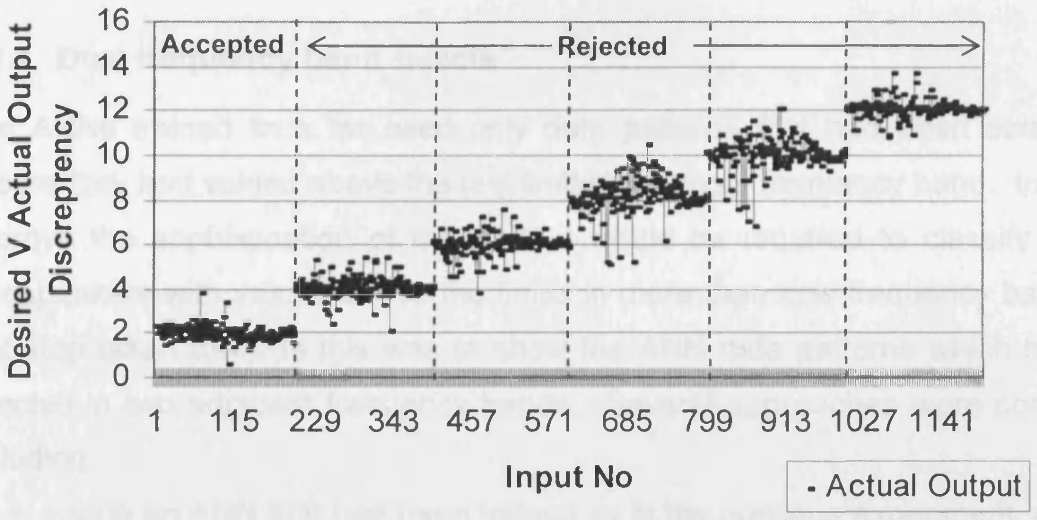


Figure 7.24a - ANN response to training data

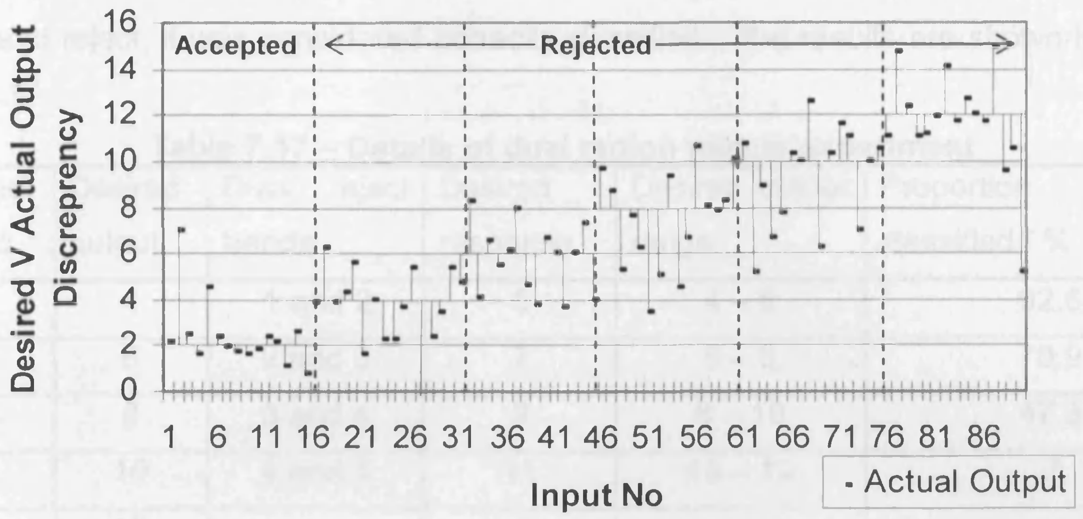


Figure 7.24b – ANN response to previously unseen data

7.9.4 Dual frequency band rejects

The ANNs trained thus far used only data patterns that had been screened to ensure they had values above the test limits in a single frequency band. In order to improve the sophistication of the ANN it would be required to classify rejected loudspeakers with values above the limits in more than one frequency band. The first step taken towards this was to show the ANN data patterns which had been rejected in two adjacent frequency bands. Several approaches were considered, including:

- using an ANN that had been trained as in the previous experiment, with data patterns which had values above the limits in only one frequency band,
- using an ANN trained with both single band reject patterns and those with values above the limits in two adjacent bands, giving each scenario a different output value.

The latter method was considered too complex for a single ANN due to the numerous outputs required, hence the former method was employed. Initially only data patterns with values above the limits in two adjacent frequency bands were shown to the previously trained ANN and the output compared to the average of the desired output values for the two frequency bands used in the previous experiment, for example, the desired output for band 2 was 6 and for band 3, 8, so the desired output value for the dual band reject was approximately 7. If the output value of the dual band rejects was between the two desired responses of the single band reject, it was considered correctly classified. The results are shown in table 7.17.

Table 7.17 – Details of dual region rejects experiment

eject and	Desired output	Dual reject bands	Desired response	Desired output range	Proportion correctly classified / %
1	4	1 and 2	5	4 – 6	92.6
2	6	2 and 3	7	6 – 8	70.9
3	8	3 and 4	9	8 – 10	47.3
4	10	4 and 5	11	10 – 12	-*
5	12				

*There were no rejected loudspeaker data patterns available with values above the limits in both regions 4 and 5.

None of the outputs in the single reject band experiments gave exactly the desired value, hence a certain degree of ambiguity was expected in the results from the dual band rejects. A tolerance of +/- 0.5 outside the desired output range was therefore included in the results shown in table 7.17 in order to gauge the success of this method taking into account the possible variation in output. In practice this overlap of the frequency band output values would of course lead to ambiguity as to which band the rejected loudspeaker should be assigned to. It can be seen that good correlation between desired output and actual output was obtained when bands 1 and 2 were combined, but the accuracy deteriorates significantly in the other combinations.

7.9.5 Multiple ANNs for frequency band analysis

The results obtained from the method described above of using just one ANN to classify all the frequency bands were unsatisfactory, it was therefore proposed to employ several ANNs, each of which would be trained with data from one frequency band. Each ANN would then determine if the loudspeaker should be accepted or rejected in that frequency band and the outputs from all the ANNs would be compiled into a vector of ANN responses that identified all of the frequency bands in which the loudspeaker's rub and buzz measurement was above the test limits. This is illustrated in figure 7.25, which shows the data pattern split into 2 segments and fed to 2 ANNs for simplicity. In the actual experiment the frequency range was split into 5 bands with the data pattern divided as evenly as possible between them (this was 3 bands with 10 data points and 2 bands with 9 data points). Training data sets were compiled from all rejected loudspeaker data patterns, including gross rejects, as there were very few rejects in some frequency bands with values below the boundary used previously (the limits + 6σ). The mean was removed from the data patterns, although this may not have normalised some of the gross reject data patterns. The training sets contained 2000 loudspeaker data patterns each, half of which were rejected in the appropriate frequency band and half of which were accepted. The validation sets contained 400 loudspeaker data patterns, which also consisted of half rejected and half accepted data patterns in the appropriate frequency band. The validation results for each network are shown summarised in table 7.18.

Table 7.18 – Results for multiple ANN training sessions

Frequency band	Training rms error	Validation rms error	Classification accuracy / %
1	0.187	0.170	100
2	0.293	0.301	98
3	0.276	0.227	99.5
4	0.401	0.261	99
5	0.249	0.273	98

In order to determine the effectiveness of the 5 networks acting as a single classification system, all the available rejected loudspeaker data patterns were split into the 5 frequency bands and passed through the networks. From the network outputs an output vector was derived which indicated in which frequency bands the loudspeaker was above the test limits. This vector was compared to that derived from the end of line test output and the correlation between the two determined. This is illustrated in table 7.19.

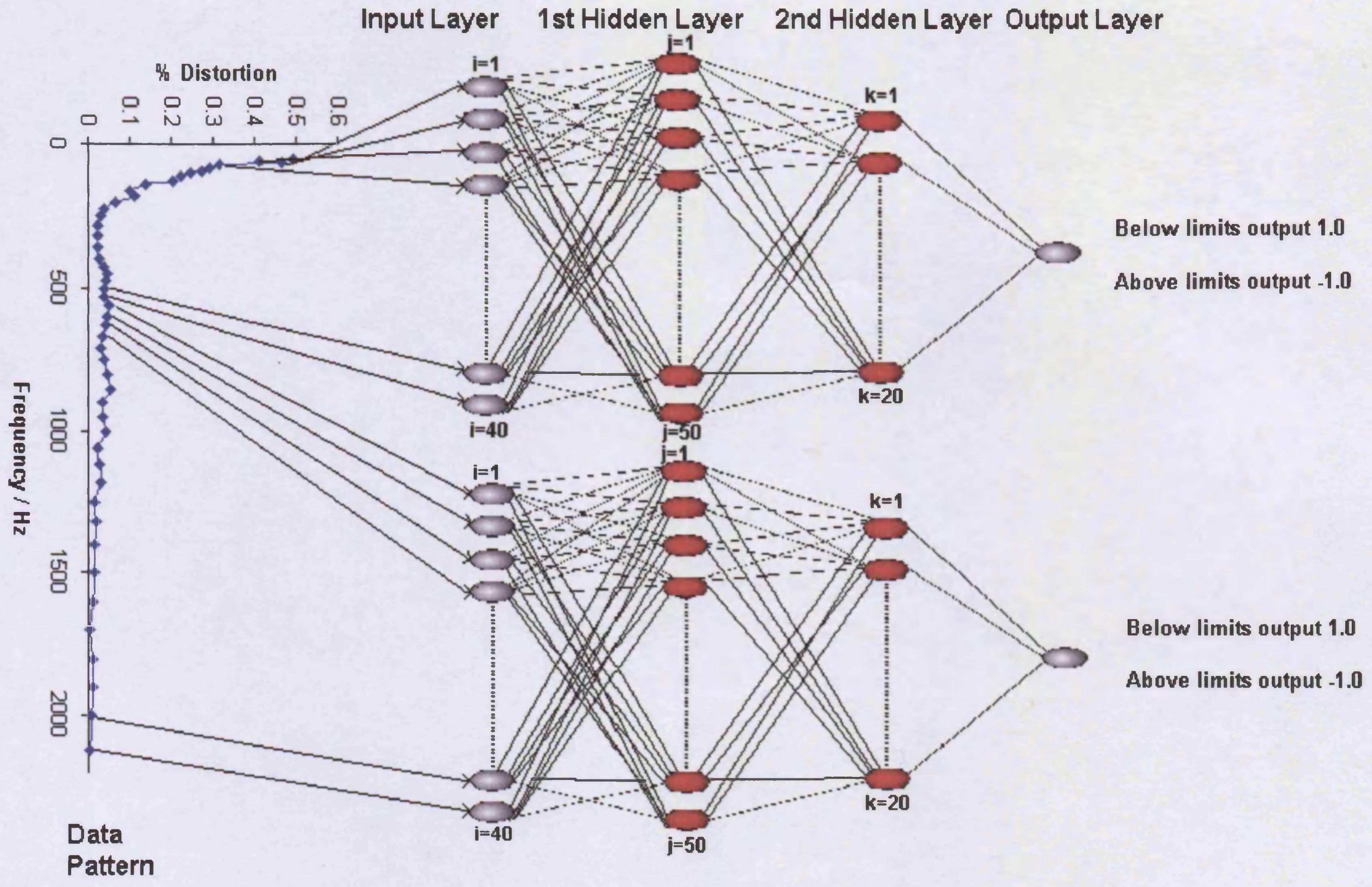


Figure 7.25 – Multiple ANNs used for frequency band analysis

Table 7.19 – Correlation of output vector between EOL test and ANN

Number of frequency bands	Correctly classified / %
5	49.8
4 or more	84.8
3 or more	97.2
2 or more	99.7
1 or more	100

It should be noted that some of the data patterns used here were also used in the training sets, however the majority of the data patterns were previously unseen by the networks.

The results above illustrate that this method of frequency band analysis was a considerable improvement upon the single ANN method as there is good correlation between end of line test results and ANN output. There is also a substantial increase in sophistication in the system in the form of the output vector which describes the state of each of the frequency bands, where the single ANN system was limited to data patterns that were rejected in only one or two adjacent frequency bands.

This method could be developed further to incorporate a larger number of smaller frequency bands which could target frequency ranges where specific faults are known to exhibit. The limitations to this are the decipherability of loudspeaker faults through frequency analysis.

7.10 ANALYSIS OF INORDINATE OUTPUT VALUES

7.10.2 Statistical analysis

In all experiments several data patterns in the validation set produced significantly inordinate output values. Several statistical techniques were employed to determine what differentiated these patterns from those whose output did fall within the correct range including:

- mean of the data pattern,
- mean of the frequency bands,
- high value search – the data patterns were checked for any higher than normal values which may have affected the ANN output. This did not result in any explanations; all the data patterns had been preprocessed to ensure they did not have values above the limits plus 6σ,
- standard deviation of data pattern – to determine if the variation in the data pattern was greater than normal,
- cross-correlation – between two data patterns, one classified correctly (i.e. the output value corresponds to the frequency band in which it has values above the test limits) and one classified incorrectly. The correlation coefficient was compared to the coefficient generated by two data patterns that were both classified correctly,
- Euclidean distance – the inverse of the Euclidean distance between two data patterns gives a measure of the similarity of the two data patterns.

One example, taken from the dual frequency region reject network (see section 7.9.4), will be given here to illustrate the process. A data pattern, to be referred to here as A, had values above the limits in region 2 (450-630Hz) and region 3 (280-425Hz), the desired ANN output value was therefore between 5.5 and 8.5, however the actual output value was 11.46. Pattern A had one value above the limits in region 2, and 3 values above the limits in region 3. A comparison was made to two other data patterns with the same number of values above the limits in the corresponding regions; these will be referred to here as patterns B and C. The data patterns are shown in figure 7.26.

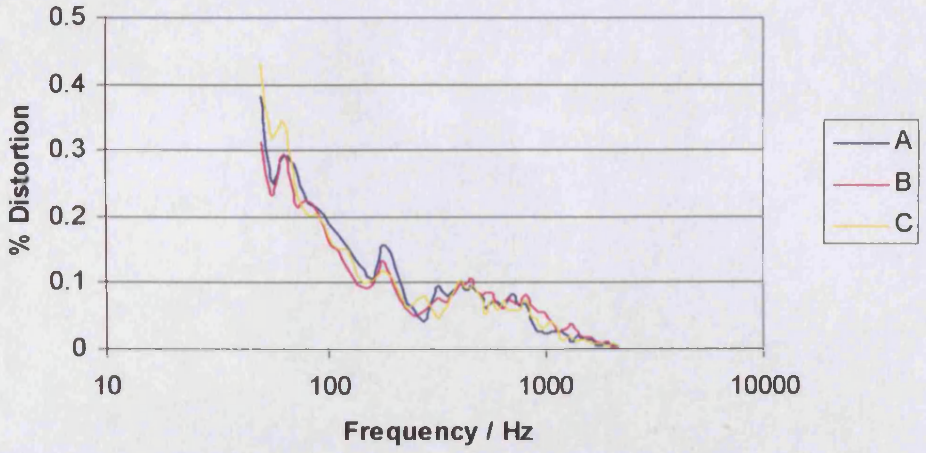


Figure 7.26 – Data patterns under investigation

7.10.2.i Mean of data pattern

The mean value of patterns A, B and C are shown in table 7.20.

Table 7.20 – Mean values

Pattern	Mean
A	9.07×10^{-2}
B	8.87×10^{-2}
C	8.98×10^{-2}

As can be seen from table 7.20, the mean value of pattern A is slightly higher than B and C. This is consistent with other data patterns with inordinate values that were investigated, however in all cases the difference was small and it has not yet been determined if this is the sole explanation for the extreme ANN output.

7.10.2.ii Mean of frequency bands

The mean value of the whole data pattern may not be entirely representative as the pattern contains relatively large values at the lower frequencies which may obscure any information which exists at the higher frequencies, hence a mean was taken of each of the frequency bands, as shown in table 7.21.

Table 7.21 – Mean of individual regions

Pattern	Band 1	Band 2	Band 3	Band 4	Band 5
A	2.90×10^{-2}	7.61×10^{-2}	8.16×10^{-2}	9.00×10^{-2}	2.11×10^{-1}
B	3.84×10^{-2}	8.34×10^{-2}	8.16×10^{-2}	7.03×10^{-2}	1.87×10^{-1}
C	2.92×10^{-2}	7.47×10^{-2}	7.80×10^{-2}	8.10×10^{-2}	2.12×10^{-1}

The means of the frequency bands show that pattern A does not contain consistently higher values than patterns B and C, the mean is only significantly higher in region 4, hence it would not be fair to conclude that pattern A has an inordinate ANN output value because its values are higher than patterns that returned the desired output value.

7.10.2.iii Standard deviation

The standard deviation is indicative of the degree of variation in the data patterns. The values for patterns A, B and C are given in table 7.22. It can be seen that pattern C has the greatest variation and pattern B the lowest, hence no conclusion can be reached from this information.

Table 7.22 – Standard deviations

Pattern	Standard Deviation
A	8.52×10^{-2}
B	7.23×10^{-2}
C	9.13×10^{-2}

7.10.2.iv Cross-correlation

It was anticipated that a cross-correlation calculation may perceive more subtle differences in the data patterns as it compares the data value by value. It was expected that there would be a lower correlation between pattern A and B, and A and C than between patterns B and C. The cross-correlation values are shown in table 7.23.

Table 7.23 – Cross-correlation values

Pattern Combination	Correlation Coefficient
A and B	9.79×10^{-1}
A and C	9.73×10^{-1}
B and C	9.74×10^{-1}

It can be seen from table 7.23 that the correlation coefficient for patterns A and B was actually higher than for patterns B and C. This was not the case for every data pattern investigated, several had a clear difference, however this is obviously not a universal explanation for the disparity between end of line test and ANN output.

7.10.2.v Euclidean distance

The Euclidean distance calculation gives a value corresponding to the difference between two data patterns:

$$d(x_i, x_j) = \|x_i - x_j\| = \left[\sum_{k=1}^{m} (x_{ik} - x_{jk})^2\right]^{1/2}$$

(7.5),Haykin, (1999)

where x_{ik} and x_{jk} are the kth elements of the data patterns x_i and x_j respectively. Hence a measure of similarity is given by the inverse of the Euclidean distance. The Euclidean distances and similarity coefficients for data patterns A, B and C are given in table 7.24.

Table 7.24 – Euclidean distances and similarity coefficients

Data Pattern	Euclidean Distance	Similarity Coefficient
A and B	0.141	7.077
A and C	0.146	6.855
B and C	0.182	5.482

It was anticipated, as with the correlation coefficients, that there would be a lower similarity between pattern A and B, and A and C than between patterns B and C, as the latter both returned similar values when presented to the ANN, however, the results do not support this hypothesis. It can be seen from table 7.24 that the similarity coefficients depict the data slightly differently to the correlation coefficients; the similarity coefficients are higher for A and B, and A and C, than for B and C.

7.10.3 ANN training analysis

As no definitive explanation was determined through the statistical analysis of the ANN's output, an investigation into the data used to train the ANN was performed. The aim of the investigation was to determine if the reason for the ANN misclassifying loudspeaker data patterns was due to the content of the training data set. 16 training data patterns with extremely similar values to that of a validation data pattern that had been misclassified by an ANN trained in section 7.8 (a basic accept/reject ANN) were selected. The training data patterns were presented to the network and their desired and actual output values were examined. This ascertained that the network was extremely consistent in its output. Of the 16 data patterns, 4 were classified by the EOL test as accepted and the remainder as rejected, however, the network classified all of the patterns as rejects, with the output value returned by the network varying by less than 1 per cent. This is illustrated in figure 7.27.

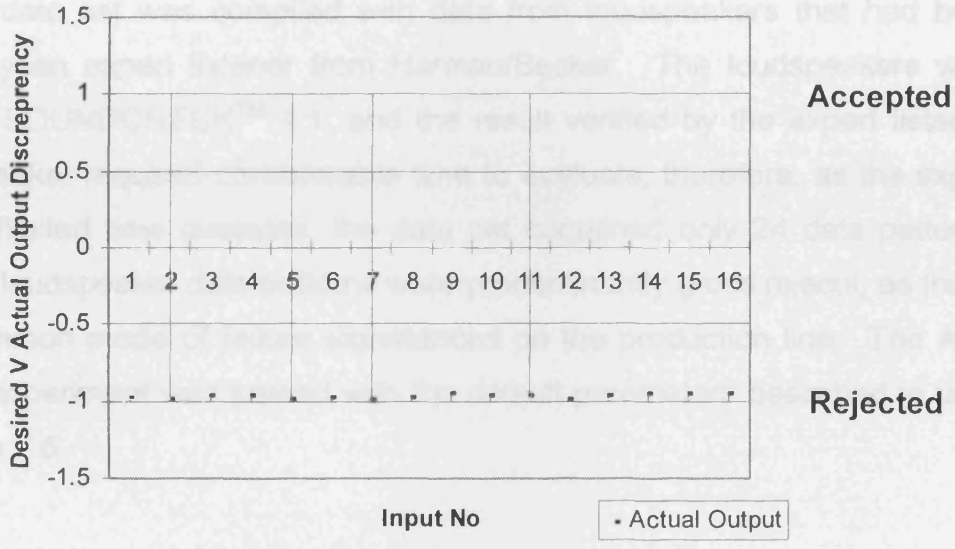


Figure 7.27 – ANN output

When the data patterns were examined it was ascertained that the rejected loudspeaker data patterns had just a single value slightly above the limits, with all other values extremely similar to the accepted loudspeaker data patterns, this is illustrated in figure 7.28.

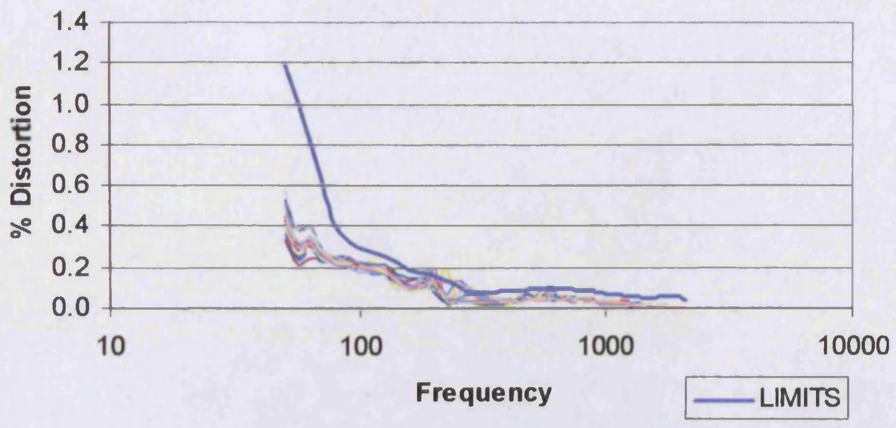


Figure 7.28 – Data patterns under investigation

It is therefore proposed that the reason for misclassification of certain loudspeaker data patterns by the ANN is due to conflicting training data. In this example, of 16 extremely similar data patterns, the majority, 12, had an output relating to rejected loudspeakers and only 4 relating to accepted loudspeakers, the network was therefore trained to recognise all these data patterns as rejects.

7.11 VALIDATION OF ANN WITH DATA VERIFIED BY AN EXPERT LISTENER

A validation data set was compiled with data from loudspeakers that had been listened to by an expert listener from Harman/Becker. The loudspeakers were tested using SOUNDCHECK^{TM} 4.1, and the result verified by the expert listener. Each loudspeaker required considerable time to evaluate, therefore, as the expert listener had limited time available, the data set contained only 24 data patterns. The rejected loudspeaker data patterns were predominantly gross rejects, as this is the most common mode of failure experienced on the production line. The ANN used in this experiment was trained with the default parameters described in table 7.3 of section 7.8.

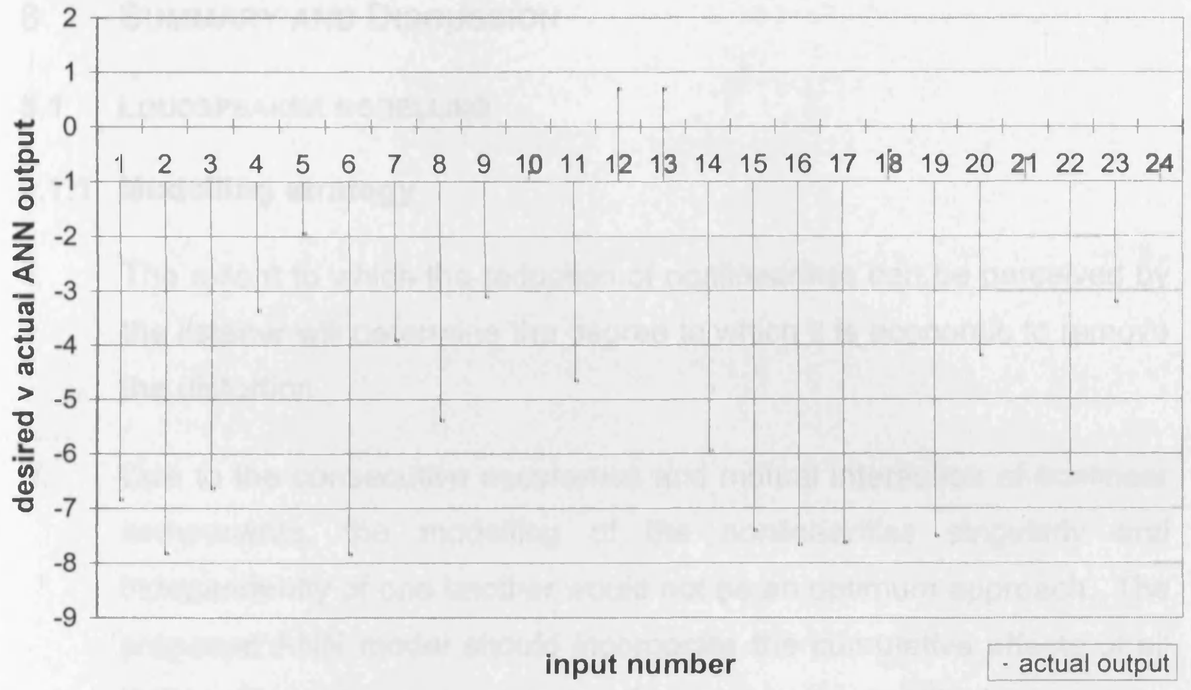


Figure 7.29 – Validation results for expert listener verified data

The ANN was not trained with gross rejects, however the network classified 22 of the 24 data patterns (92 per cent) correctly, as shown in figure 7.29. The magnitude of the majority of the data pattern's outputs were significantly larger than that generated by rejected data patterns with values close to the limits, as the magnitude of the values in the data patterns were significantly larger. The configuration of the ANN weights was such that it could still generate the appropriate negative output to classify the rejected data pattern correctly, despite the increased magnitude of values in the data pattern.

8 SUMMARY AND DISCUSSION

8.1 LOUDSPEAKER MODELLING

8.1.1 Modelling strategy

i. The extent to which the reduction of nonlinearities can be perceived by the listener will determine the degree to which it is economic to remove the distortion.

ii. Due to the consecutive occurrence and mutual interaction of nonlinear components, the modelling of the nonlinearities singularly and independently of one another would not be an optimum approach. The proposed ANN model should incorporate the cumulative effects of all the nonlinearities present in the frequency and amplitude range, as well as any interactions between the nonlinear elements, which may not be assimilated by individual parameter models, assuming that it has been trained with sufficient data to be able to generalise to this extent.

iii. The proposed method is a genuinely black box technique that does not require prior knowledge of any parameters and can therefore be applied universally.

iv. The aim was to update the model at regular intervals during operation in order to incorporate developments in the transfer function that occur over time.

v. In order to employ the ANN distortion reduction scheme the system would be transferred on to a processor chip, which could then be incorporated in to the loudspeaker amplifier circuit. For it to be feasible to use a processor chip, the ANN must be relatively compact, so that a small, cost effective processing unit may be used. Large ANNs would seriously compromise the processing speed of a chip.

8.1.2 ANN training data

i. The acquisition of training data through excitation with a music signal was necessary in order to facilitate the adaptation of the ANN model to alterations in the loudspeaker transfer function during operation. Using a different form of excitation signal, such as a sine sweep or white noise would interrupt the performance of the loudspeaker, which would be unacceptable to the listener who should be completely unaware of the linearisation process.

ii. Options considered for compiling training data for model updates included:

- generating a new training set for each training session;
- accumulating the most nonlinear data over the period of loudspeaker operation;
- accumulating data relating to the full frequency and amplitude range of the loudspeaker;
- using data measured real-time to continuously update the ANN weights.

iii. The only preprocessing applied to the input/output signal was a conversion from millivolts to volts, in order to reduce the magnitude of the ANN inputs in line with the magnitude of the initial values of the weights. This is an extremely simple preprocessing sequence in comparison to many linearisation schemes, for example (Low, Hawksford, 1993) derive cone displacement from the back emf signal through numerical integration which requires the knowledge of several system parameters. Reducing the required preprocessing simplifies implementation and also reduces processing time, hence reducing delay between input and loudspeaker output.

iv. Decreasing the amount of data preprocessing reduces the computing power and thus the size of the processing chip required by the system. This therefore makes the system more commercially viable.

v. Signal measurements were subject to noise, with low amplitude measurements being particularly susceptible. It was observed that below 4.0mV the amplitude was almost indiscernible from the noise in the signal. A comparison of ANN performance resulting from a training session with a training set that contained a full range of amplitude data and one where the low amplitude, noisy data was removed showed little advantage in removing the noisy data, the ANN appeared to be able to generalise so as not to be affected by the inclusion of the noisy data.

vi. The loudspeaker input/output signal had to be compressed in order to facilitate ANN training over the target time period. This inevitably led to the loss of information contained within the signal.

8.1.3 Model optimisation

i. Two ANN structures were considered over the course of this investigation, the Multi Layer Perceptron (MLP) feedforward network and the Elman recurrent network. It was a relatively straightforward process to modify the back propagation MLP C++ code to accommodate the Elman network algorithm.

ii. Each of the network parameters, along with algorithm modifications, were investigated in order to determine the configurations where correlation between model output and actual loudspeaker output was optimised.

iii. All ANN models responded well to low amplitude input data, as the loudspeaker transfer function approaches linearity in this region, however with higher amplitude input the loudspeaker response was more nonlinear and thus it was in this region that the models' performances could be discriminated. The validation data set therefore contained only high amplitude data.

8.1.3.i Multi layer perceptron

i. The optimum ANN topology was determined to be 12 neurons in the first hidden layer and 5 in the second hidden layer. Significantly increasing the size of the ANN architecture did not significantly improve performance, and therefore could not justify the considerably increased training times required for such architectures.

ii. The results of the investigation into ANN topology confirm the assertion by Miller, (1999) that the performance of the ANN model is subject to the suitability of the network architecture to model that particular system.

iii. The investigation into the optimum number of training epochs determined that although the rms error value calculated for the training set consistently decreases as epoch number increases, the validation data rms error begins to increase again after 1 x 10^4 epochs. This is due to a phenomenon known as overfitting. It is therefore unbeneficial to train the ANN beyond 1 x 10^4 epochs.

iv. Loudspeakers are dynamic systems and thus an input-output model requires past inputs and outputs in order to predict the new system output. The optimum number of previous inputs and outputs was determined to be 3. This was a smaller vector than was expected and may be due to the low nonlinear content of the training data. The optimum training data format was found to be in_t, out_{t-1}, out_{t-2},...out_{t-4}, in_{t-1}, in_{t-2},...in_{t-4}.

v. The momentum term did not significantly affect the rms values, however, a momentum value of 1 x 10^{-3} resulted in a marginally lower validation rms error.

vi. Despite achieving the lowest rms error of the investigation, the ANN with a learning rate of 1 x 10^{-3} reached a local minimum that it could not escape within the training period. Therefore, 1 x 10^{-3} was not considered the optimum value for the learning rate; the next best result, 1×10^{-6}, was sufficiently lower to reduce the possibility of the weights becoming trapped in a local minimum and not too low to adversely affect the rate of convergence.

vii. It was ascertained that a logarithmic relationship existed between training set size and validation rms error. The best performing ANN was therefore that trained with the largest training set employed in this investigation, 5×10^4.

viii. The most suitable method of training data selection was determined to be extract subsets from the whole data set available. It was ascertained through Fourier analysis that information in the data set was lost during re-sampling, therefore this method was unsuitable, and that using a large data set over fewer training epochs reduced the modelling performance of the ANN.

8.1.3.ii Modified Elman

i. As with the multi layer perceptron, the optimum architecture for the modified Elman network was not the largest, but that which was most compatible with the loudspeaker system. This was determined to be 3 neurons in the hidden layer (and therefore context layer).

ii. The investigation into the optimum number of training epochs determined that significant improvements in the ANN performance can be achieved by increasing the number of training epochs up until approximately 1×10^2 epochs, however the gain in ANN performance beyond this point constantly decreases. Therefore, there is little benefit in extending the training session beyond 5×10^3 epochs, especially when the significantly prolonged training period is considered.

iii. The default number of training epochs was set at 500. This was due to time restraints resulting from the extremely slow execution rate of the original modified Elman algorithm C++ code. The C++ code was altered to execute significantly more efficiently during the investigation discussed in section 6.2.

iv. A context layer self-feedback gain (α) value of 0.3 was found to produce optimum ANN performance.

v. The investigation ascertained that the momentum value does not significantly affect the minimum validation rms error values until it is above 1×10^{-1}. The optimum value was 7.5×10^{-1}.

vi. The stability of the network can be improved by employing a lower learning rate for the weights connecting the context layer to the hidden layer, thus reducing the rate of change of the weights that control the feedback. Experiments were conducted investigating a large number of learning rate combinations. It was determined that the stability of the training process was highly dependent upon the learning rate values and only one combination of those investigated resulted in a constantly decreasing training and validation rms error over the 500 epoch training period. The value for the context layer learning rate was 1×10^{-6} and for the rest of the network 1×10^{-5}.

vii. The investigation into training set size determined that the relationship between training set size and validation rms error value is best approximated by a power series in this case.

8.1.3.iii Optimum Configurations

i. Networks were trained with the optimum parameter values determined during the investigation. The optimum values for training set size and number of epochs were determined to be those where the gain in increasing the value further was outweighed by the increase in the training period.

ii. The validation rms error for the modified Elman network was 0.0591, the lowest achieved thus far, as would be expected with the use of an optimum parameter configuration, however, in the case of the multi layer perceptron, lower validation rms error values were achieved with alternative configurations. The lowest rms error achieved for the multi layer perceptron was 0.0129. This substantiates Miller's (1999) assertion that there exists an interaction between ANN parameters, therefore determining the optimum parameter configuration is not a simple case of identifying suitable parameters individually.

iii. Further experiments to investigate the interaction of ANN parameters may result in an improvement in achievable ANN performance.

iv. Chang et al (1994) utilised 100 units in the tapped delay line preceding the neural network input layer. In conjunction with 30 neurons in each of the two hidden layers, when modelling the combined transfer function of loudspeaker and room acoustics, a training rms error value of 0.0031 was achieved. However, no evidence was presented of the model's performance with previously unseen data and therefore the ANN weights being overfitted cannot be discounted, also large numbers of neurons in the hidden layers were used together with significantly more previous inputs and outputs in the input layer compared to that used in this investigation

8.1.3.iv Nonlinearity identification capability of ANN model

i. The validation rms error implied that there was good correlation between ANN model output and actual loudspeaker output, however, to confirm that the ANN model had identified the nonlinearities in the loudspeaker transfer function, an analysis of the data was performed using Pearson's Product Moment Correlation Coefficient (R^2). However, it was determined that the vast majority of the nonlinearity in the training and validation data sets occurred in the low amplitude regions, which was considered more likely to be caused by measurement noise rather than actual nonlinear behaviour of the loudspeaker. Therefore this investigation gave little insight into the performance of the ANN model in identifying nonlinearities.

8.1.3.v Frequency response of ANN model

i. In order to evaluate the ANN model's frequency response curve, the output of the best performing multi layer perceptron and modified Elman ANN models was analysed using Mathworks MathCAD 11 Fourier transform function.

ii. The frequency response of the best performing multi layer perceptron model resembles a low pass filter. However, the response of the actual loudspeaker more closely resembles a band-pass filter, with the response increasing significantly at the lower frequencies and slowly diminishing at higher frequencies. Also, the frequency response generated by the multi layer perceptron model is considerably flatter than the actual loudspeaker response.

iii. The frequency response curve discussed above was generated using a 5v input. Further frequency response curves were generated to determine the response of the models to input amplitudes not present in the training data. The maximum input amplitude in the training data was approximately 25v, therefore the model's response to a 20v input, just below the threshold and to a 50v input, well above the threshold was tested. The response of the model at higher input amplitudes is clearly significantly different to that at lower input amplitudes.

8.1.3.vi Distortion measurements from ANN model

i. The distortion curves generated from the best performing ANN models showed significantly lower levels of distortion than were measured from the actual loudspeaker.

8.1.3.vii ANN Training with Nonlinear Data

ii. In the case of the multi layer perceptron, significantly improved modelling performance of the loudspeaker frequency response and harmonic distortion was achieved when the most nonlinear data were selected as training data, however the modified Elman network showed no improvement. Implementing nonlinear data selection in practice may be complicated, however, it is likely that the loudspeaker will present more nonlinear behaviour as the operating period progresses, thus making it unnecessary to actively select the most nonlinear data.

8.1.4 Algorithm development

8.1.4.i Multi layer perceptron with time delay dependent input weightings

i. It is logical that the most recent input and output values will have a greater influence on the loudspeaker's behaviour, and also that inputs will have greater significance than outputs. It is therefore proposed that the multi layer perceptron ANN input pattern should be weighted in order to reflect this. The inputs and outputs were multiplied by an exponentially decreasing coefficient between 0 and 1, and the outputs multiplied by a further constant coefficient between 0 and 1. This would provide some rating of importance to the data. The validation rms achieved using the best performing parameter configuration determined for the multi layer perceptron in section 5.3 was 0.034. It was therefore concluded that time delay dependent weightings did not improve the multi layer perceptron's performance.

8.1.4.ii ANN training duration

i. The project aim was to realise a model update rate of 5 minutes. This necessitated that each subsequent ANN model be trained within those 5 minutes.

ii. The optimum network architectures described in section 5.3 were employed during this investigation and tests were conducted to ascertain a suitable compromise between the number of training epochs and the training set size in order to optimise the network performance over a training period of five minutes.

iii. Two approaches to weight initialisation were investigated:

- Randomly initiating the ANN weights.

- Initiating the ANN weights to previously trained values.

The previously trained weights were those resulting in the best performing ANN model for the multi layer perceptron and modified Elman network architectures, as discussed in section 5.3.

iv. It was observed that the performance of both architectures was significantly improved relative to the performance of randomly initiated weights when the weights were initiated to previously trained values.

v. However, the data used in this experiment was collected over a relatively small time period, which may limit the conclusions that may be drawn from the results. It is possible that over longer time periods the response of the loudspeaker will alter more radically and thus substantially larger alterations to the weights will be required, which may be more difficult to achieve over the target training period.

vi. Despite this it is anticipated that there would be greater advantage in utilising previously trained weight values, as they are likely to be closer to an effectively performing configuration than weights that are randomly initiated.

vii. The optimal ANN parameter configuration for the original training session may be different to that for a subsequent training session, for example it may be beneficial to employ a higher learning rate so as to increase the influence of each training sample upon the ANN weights when the training set is significantly smaller, as required for a 5 minute training period. Alternatively, it may be beneficial to decrease the learning rate in order to prevent the overfitting of the ANN weight to the new data set, which may result in the loss of good generalisation performance of the ANN model.

viii. Several of the multi layer perceptron ANN parameters were adjusted, including the learning rate and momentum term, in order to determine their effect upon the training results with the aim of improving the ANN model performance after the second training session. However, no improvement in ANN performance was achieved.

ix. Training times are derived from ANN's trained using a PC with a 1.5GHz Pentium 4 processor. The training times using a Digital Signal Processing (DSP) chip may vary significantly.

x. Improved performance may be achieved if the default model to which the ANN weights are initiated to is trained with data derived from sine sweeps.

xi. The data used to train the ANN models in these experiments were measured over a 4 second period of loudspeaker operation. It is anticipated that subsequent training sessions with data measured over a longer time period, or at a time significantly removed from that at which the original training data were measured may alter the results discussed above considerably. The second training set above contained relatively similar data to that of the first training set therefore the second training session required relatively small alterations to the weights in order to accommodate the new data set. Data collected over a larger time scale may vary more significantly and therefore require appreciably larger alterations to the weights, which may be difficult to accommodate over the required training period.

xii. The effects of time dependent nonlinear parameters will develop progressively over time as discussed in section 2.4.2. In this case the model will incorporate these changes in the transfer function if trained online or at regular intervals. However, in the case of a sudden, significant change in the loudspeaker output, such as that caused by a change in volume level, (input voltage is proportional to volume level), the ANN model may not be able to adjust to an acceptable accuracy in a short period of time, resulting in significant distortion in the loudspeaker output. The nonlinearities in the loudspeaker output will be more pronounced at higher volume levels as the required cone excursion will be greater. Tests were run to determine the ANN model's reaction to a substantial change in input signal amplitude. It was determined that the frequency response of the ANN model was significantly altered, however, it is likely that this would also be the case for the frequency response of the actual loudspeaker. Several options were considered to overcome the possible distortion resulting from a significant change in loudspeaker level, including increasing the model update rate in order to reduce the period where increased distortion levels are experienced, or triggering a model retrain when a change of level is detected. However, at the current network training speed, there would be a significant period of time where the loudspeaker could be distorted as the active model is erroneous in respect to the current level. It is likely that the continuous training algorithm (see section 6.3) will be the most able to overcome a change in level in a reasonable time scale, assuming the learning rate is large enough to incorporate these changes in loudspeaker behaviour in the model.

8.1.4.iii Online training

i. As mentioned in section 3.4.6, the modelling of the loudspeaker transfer function using ANN's has already been accomplished; the aim of this project was to develop a method of training the ANN to a satisfactory level of performance over a short time period so as to accommodate the time variant parameters of the loudspeaker transfer function in the model. A linearisation scheme based upon such a model would theoretically reduce distortion more efficiently, as the model would be a closer approximation to the loudspeaker's current behaviour than that of a generalised model.

ii. The target training time period was 5 minutes. Two methods were considered to update the model active in the linearisation scheme during this period. The first method was to continuously update the ANN weights with errors generated from every sample measured real-time, with each sample used only once. The second method was to train the ANN with a selected training set over a number of epochs and therefore longer training periods, up to the full five minutes, with the new model becoming operative in the linearisation scheme at the end of the training period.

iii. It was considered necessary to include some degree of validation in the algorithm to reduce the possibility that the ANN model diverges from actual loudspeaker response, as this may result in the linearisation scheme introducing additional distortion to the loudspeaker output. The method proposed performs a validation calculation using data measured concurrently with the training sample after every weight alteration to determine if the ANN model performance has been improved by the alteration. If the ANN performance is not improved the previous weight configuration remains active in the linearisation scheme.

iv. Superior results were achieved when several training epochs were employed before the ANN weights were updated, rather than continuously updating them.

v. The modified Elman network had a significant advantage of speed of training, therefore a substantially larger training set could be employed over the 5 minute training period, hence the larger reduction in the rms error. However, the multi layer perceptron had a significantly lower validation rms error overall, hence may be considered the more effective of the two ANN architectures.

8.1.5 Further comments

i. Validation rms error varied considerably depending upon the content of the validation set, however, the relative difference between the validation rms error for each training session remained reasonably consistent.

ii. It has been determined that ANN models result in good correlation between modelled loudspeaker output and actual measured output, particularly over longer training periods. However, the ANN model does not only comprise of the loudspeaker transfer function, it also incorporates the transfer functions of all the elements between the amplifier and the voltmeter measuring the resultant back emf, including wires and resistors. These elements may also not behave completely linearly, hence even the most accurately trained ANN model will include some nonlinearities not actually present in the loudspeaker, which will contaminate the signal and may result in the addition of distortion to the loudspeaker output. However, so long as there is a net reduction in output distortion, this could be tolerated.

iii. The application of techniques such as Digital Signal Processing (DSP) may significantly enhance the results obtained during this project and should be considered in any further work.

iv. The ANN's ability to learn has previously been established through the use of relatively small, specifically selected training sets. Liu, (1993), used 400 training samples to train modified Elman ANNs to model nonlinear functions, Kalayci, Özdamar, (1995) used 1200 training samples with specific characteristics when training a multi layer perceptron to recognise spikes in electroencephalogram (EEG) waveforms, (Watton, Xue, 1997) used 1000 training samples to model fluid power systems. The considerable amount of information contained within a larger training set, (especially if some of the data conflicts, such as two identical inputs resulting in two different output measurements), may be detrimental to ANN learning, as the nonlinear characteristics it would be desirable for the ANN to extract from the data may be masked by the large volume of data with linear characteristics. However, this is unavoidable in a fully automated system, and unless filtering of nonlinear data were employed as mentioned in section 4.5.2.b, the training set must be large enough to ensure the inclusion of some nonlinear data and small enough so that this data is distinct enough to be learnt by the ANN.

v. Chang et al, 1994, used 1 x 10^4 training samples derived from white noise when modelling the combined transfer function of loudspeaker and room acoustics, however, large numbers of neurons in the hidden layers were used together with significantly more previous inputs and outputs in the input layer compared to that used in this investigation, which attained good validation results.

vi. In their ANN model of a loudspeaker Low, Hawksford, (1991) used constant input amplitude, (specifically chosen to ensure maximum cone excursion), sine wave tones swept over the frequency range of the loudspeaker, thus ensuring fully representative training data, along with significant preprocessing, to attain the ANN training signal. It would not be possible to apply this method to a practical system as the only signals available would be music signals.

vii. Alternative ANN architectures may also result in improved modelling performance. Watton, Xue, (1997), ascertained that training times for multi layer perceptrons may be long when large training patterns are employed and system noise is present, and poor dynamic modelling performance was achieved, particularly with previously unseen validation data.

8.2 FAULT DETECTION

8.2.1 End of line test

i. Once production is complete the loudspeakers are tested and those that fail the test are discarded. Various parameters are measured, those of interest in this project were frequency response and audible harmonic distortion (rub and buzz).

ii. Loudspeaker faults caused during production can be generally categorised as mechanical, chemical or placement asymmetry. These faults are further sub-categorised into 82 possible diagnoses by Harman/Becker.

8.2.2 Initial data analysis

i. Initial analysis of the frequency response data patterns ascertained that a number of the patterns contained extreme values (minus infinity). Attempts to train an ANN with such data patterns resulted in an ANN that could not adequately classify them, furthermore, the classification accuracy of other data was also compromised. It was concluded that this was due to the corruption of the training process by the training patterns with extreme values. Longer training periods and a larger number of training patterns containing extreme values may produce an ANN that can classify these patterns more successfully. However, the classification of other data patterns may still be compromised. It was therefore decided to remove any data patterns with extreme values from the training set.

ii. It was decided during this project to focus training ANNs to emulate the rub and buzz test as indications from Harman/Becker suggested that improvements to the rub and buzz test were of more value than to the frequency response test.

8.2.3 ANN to emulate rub and buzz test

i. Initial experiments involved training ANNs with data from the end of line rub and buzz test in order to emulate the output of the rub and buzz test i.e. to provide an output that indicates whether the loudspeaker should be accepted or rejected.

ii. Each data pattern was assigned a desired response according to whether it had been accepted or rejected by SOUNDCHECK^{TM} 4.1 during the end of line test.

8.2.4 Data preprocessing

i. It is generally agreed that preprocessing the ANN training data can improve the ANN performance (Haykin, 1999).

ii. Rejected loudspeaker data patterns that contained random, high level noise (defined as gross rejects) were removed from the training set in order to improve the uniformity of data patterns in the training set. These loudspeakers are easy to identify as rejects, and their inclusion in the training set significantly limited the ANN's performance.

iii. The initial boundary used to categorise a gross reject was the $\mu + 6\sigma$.

iv. The method of preprocessing investigated was normalisation. The training results of two ANNs, one trained with a normalised data set and one with a data set that was not. The preprocessing made a marginal difference to the ANN output, which in fact was detrimental.

8.2.5 ANN optimisation

i. Investigations into the topology for optimum network classification accuracy revealed diminutive variation in classification accuracy between different network topologies and in several cases larger network topologies resulted in higher validation rms errors and lower classification accuracies than smaller networks. The increased training time required was therefore unjustified. The optimum topology was determined to be 48-60-5-1, as the optimum number of neurons in the first hidden layer was 60 and in the second 5, hence the optimum topology was taken to be a combination of these results.

ii. The training epochs investigation revealed that two optimum values existed – one that optimised the validation rms error and one that optimised the classification accuracy. It was concluded that the classification accuracy was of more significance and therefore the optimum number of training epochs was determined to be 2×10^4.

iii. A momentum value of 5×10^{-1} significantly improved the ANN classification accuracy when compared to all other values tested. All other values resulted in the second term in equation (3.35) being orders of magnitude lower than the first term, thus rendering the second term insignificant in the training process.

iv. The optimum learning rate was determined to be 1 x 10^{-5}.

v. It was ascertained that the optimum training set size was 1 x 10^{3}. Larger training sets did not improve the validation rms or classification accuracy as anticipated. This may be because the extra data patterns in the larger training sets were significantly different to those in the validation set, hence the ANN is trained away from the optimum solution for the validation data.

vi. These results identify a possible limitation of ANNs in this application; increasing the range of data patterns in the training set may reduce the ANN's ability to classify individual data patterns correctly i.e. improving the generalisation of the ANN may be at the cost of classification accuracy.

vii. The optimum training set size must therefore be that which provides the ANN with a suitable compromise between depth and breadth of knowledge. This may require the periodic retraining of the ANN in order to incorporate new production trends that develop over time and eliminate those that are no longer occurring.

viii. Rejected loudspeaker data patterns with extreme values were removed from the training set in order to improve the ANN's performance when classifying data patterns with values closer to the test limits. The optimum boundary at which data patterns were excluded from the data set was determined to be the limits + 10σ.

8.2.6 Training times

i. The training time for the ANN with default parameters during the optimisation investigation was 2 hours.

8.2.7 Training data selection criteria

i. An analysis of training data patterns was conducted to identify properties of the data that may be the cause of discrepancies between ANN and end of line test output. At this time during the project, the test limits were assumed to be in the region of the $\mu + 3\sigma$. However, the analysis revealed that there were a significant number of accepted and rejected loudspeaker data patterns with values that were above $\mu + 6\sigma$ or below $\mu + 3\sigma$ respectively.

ii. Examination of the test limits and the method used to derive them determined that the test limits were significantly higher than the $\mu + 3\sigma$ for most frequencies. The test limits are actually derived from 3 initial production runs, where they are set to a level such that loudspeakers with audible distortion are rejected. These limits may then be modified at any time when a production trend develops that may have distortion levels above the limits but which is inaudible [Anthony, 2003].

iii. Examination of the test limit history revealed nine alterations over the data collection period, all of which were to raise the limits, resulting in loudspeakers that would previously have been rejected under the former limits being accepted under the amended limits. Therefore, if the test limits were raised to a level where the previously rejected loudspeaker would be accepted, the rejected data pattern could be almost identical to, or even have lower values than, an accepted data pattern recorded under the subsequent test limit version. Attempting to train an ANN with this conflicting data would inevitably lead to discrepancies in ANN output when compared to the end of line test results.

iv. The training data selection criteria was therefore modified to include all accepted loudspeaker data patterns and all rejected loudspeaker data patterns with all values below the limits $+ 6\sigma$. ANN classification accuracy was significantly improved.

8.2.8 Summary of ANN development

i. Figure 8.1 and table 8.1 show the improvements in ANN classification accuracy that have been made over the course of the research.

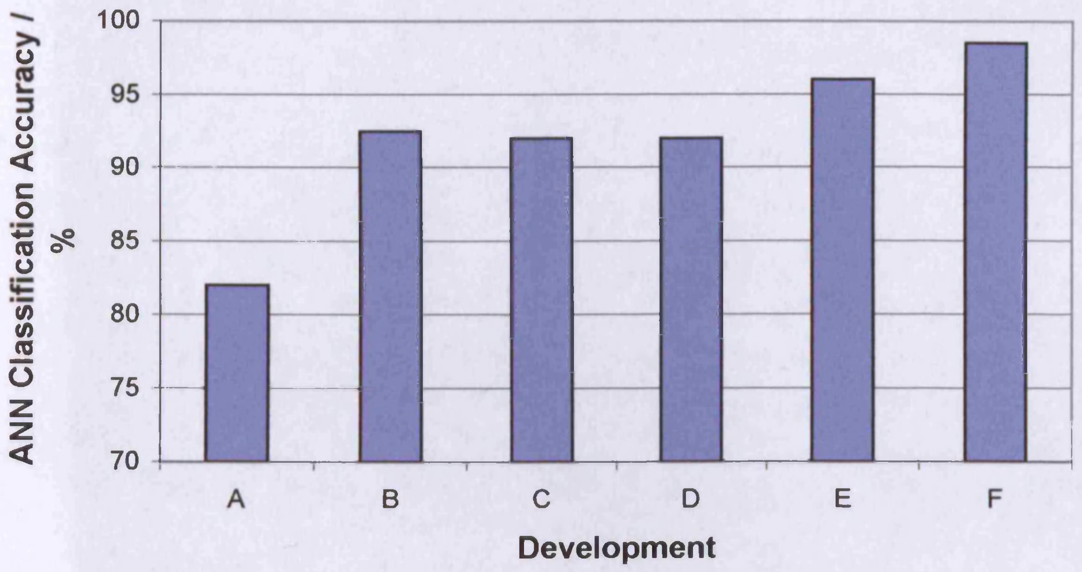


Figure 8.1 – Improvements in classification accuracy

Table 8.1 – Key to developments in figure 8.1

Label	Description
A	No Preprocessing, No Data Selection
B	No Preprocessing, Gross Rejects Removed
C	Preprocessing, Gross Rejects Removed
D	Data Selected from One Test Limit Version
E	Change of Data Selection Criteria to Limits + 6σ
F	ANN Parameter Optimisation

ii. The maximum correlation of results between the ANN and EOL test for the accept/reject ANN was 98.5 per cent.

iii. An investigation was conducted to ascertain if the ANN could be more specific and give an output relating to the frequency at which the loudspeaker was being rejected, as this could be indicative of the loudspeaker's defect.

iv. This resulted in a reasonable correlation between desired output and actual ANN output when the frequency range was split into three and five bands, and rejected data patterns with values above the limits in a single frequency band were used in the training and validation sets.

v. Experiments were conducted to ascertain optimum data configuration, frequency band allocation and desired output values. The results suggested that grouping one data pattern from each rejection band together, followed by one accepted data pattern with output values of equal increments between 2 and 12, where the frequency regions had been allocated by assigning an equal number of the available data patterns to each frequency band gave the lowest validation rms error values.

vi. Improving the sophistication of the ANN discussed above to incorporate loudspeaker data patterns with values above the limits in more than one frequency band resulted in a significant loss of accuracy.

vii. Employing multiple ANNs to produce a vector describing the frequency bands in which the loudspeaker was rejected resulted in a significant improvement in accuracy in comparison to the multiple output ANN. The output was also significantly more sophisticated as loudspeaker patterns with values above the limits in any combination of the frequency bands could be analysed by the multiple ANN system. This system therefore may be considered to be an improvement upon SOUNDCHECK^{TM} 4.1.

viii. This method could be developed further to incorporate a larger number of smaller frequency bands which could target frequency ranges where specific faults are known to exhibit.

ix. However, a significant increase in precision was considered unachievable while using only total harmonic distortion data, due to the inseparability of signals relating to specific faults. A possible improvement would be to train ANNs to analyse distortion data from individual harmonics, instead of the total harmonic distortion that has been used here. There may be a greater possibility of diagnosing specific loudspeaker faults with this data.

x. Previously documented fault detection systems (Kalayci and Özdamar, (1995), Foo et al, (2002)) generally do not employ large training sets; small training sets of fault targeted data are used to train the ANN. In the case of loudspeaker fault detection, there were a multitude of possible fault diagnoses, which did not necessarily produce separable characteristics in the total harmonic distortion curve which was used to train the ANN. Therefore, additional, or more fault specific data would be required in order to achieve this.

xi. However, it may be argued that employing ANNs may not be the optimum method; the technique of splitting the data pattern into frequency bands in order to obtain an output vector could be used in conjunction with a system such as SOUNDCHECK^{TM} 4.1, which uses definite numerical limits, to better effect.

xii. It is questionable as to whether the degree of accuracy obtained during this investigation could be significantly improved upon. It has become evident that an ANN cannot be trained to decipher between a loudspeaker with just one value slightly above the audible distortion level and one that has no audible distortion. When training ANNs with this marginally dissimilar data, the ANN generalises to a degree such that the output for both cases is extremely similar. In this scenario the EOL test should have no difficulties discerning between the two loudspeakers, hence it is concluded that, although an unsatisfactory outcome, this research has determined that ANNs do not provide a superior method of determining whether a loudspeaker should be accepted or rejected than the current EOL test, SOUNDCHECK^{TM} 4.1.

8.2.9 Analysis of Misclassified Data Patterns

i. Statistical analysis to determine what differentiated misclassified loudspeaker data patterns from those that gave ANN output values close to the desired output did not result in any definite conclusions.

ii. The data patterns were 49 point vectors. With such a large input vector it is difficult to accurately ascertain the reason for discrepancies between actual and desired ANN output. The data patterns could vary significantly in a manner that was not picked up by the statistical analysis.

iii. Particular configurations of the data pattern may result in ANN misclassification. The ANN requires previous exposure to similar patterns in order to learn successfully and therefore produce correct outputs for previously unseen data patterns. If this was not the case and the particular configuration of data pattern was sufficiently different to any data pattern in the training set, or even that there were not enough examples of the data pattern in the training set, the ANN may not have learnt the desired output for that data pattern.

iv. Similarly, a data pattern may be misclassified if the ANN training set contained a majority of similar data patterns with the opposite desired response.

v. An investigation into ANN training data determined that when 16 training data patterns identified as having a similar degree of variation were presented to the previously trained ANN, the ANN outputs for all the data patterns were extremely similar, despite the EOL test classifying 4 as accepted loudspeakers and 12 as rejects. It was proposed that this result was obtained because the majority of the training data patterns were rejects, hence the ANN was trained to assign the most common output value to data patterns of this configuration.

vi. Therefore it is unlikely that the ANN can be trained to decipher EOL rejects with only marginal audible distortion from accepted loudspeakers. Also, in the case discussed in section 7.10.3, the ANN had been trained to recognise the pattern configuration in question as a reject, which may not be the most suitable outcome.

8.2.10 Validation of ANN with data verified by an expert listener

i. A validation data set was compiled with data from loudspeakers that had been listened to by an expert listener from Harman/Becker. The 24 rejected loudspeaker data patterns were predominantly gross rejects, as this is the most common mode of failure experienced on the production line.

ii. The ANN was not trained with gross rejects, however the network classified 22 of the data patterns (92 per cent) correctly. The magnitude of the majority of the data pattern's outputs were significantly larger than that generated by rejected data patterns with values close to the limits, as the magnitude of the values in the data patterns were significantly larger. The configuration of the ANN weights was such that it could still generate the appropriate negative output to classify the rejected data pattern correctly, despite the increased magnitude of values in the data pattern.

8.2.11 Further comments

i. The application of techniques such as Digital Signal Processing (DSP) and alternative ANN training algorithms may significantly enhance the results obtained during this project and should be considered in any further work.

ii. Throughout the course of this research it has been necessary to assume that the end of line test classifies correctly, as only limited data have been available where the end of line test result has been verified by a trained human listener. Without physical testing and comparison loudspeaker by loudspeaker it is not possible to make definitive conclusions on the accuracy of the ANN, as these calculations are based upon the correlation with end of line test output.

iii. If greater confidence is considered necessary, data compiled specifically for this purpose is required as the accuracy of the network is directly related to the accuracy of the data used in the training process.

iv. The ANN will have to be retrained at regular intervals in order to incorporate any new production trends and to exclude old trends that are no longer occurring.

v. Jack, Nandi (2000) used a Genetic Algorithm to dramatically reduce the number of inputs required by the ANN in order to classify bearing faults. The Genetic Algorithm identified the most significant features in the input data and these alone were used to train the neural network. This method not only reduced the size of the ANN required, but also significantly improved classification accuracy. A similar method may be beneficial in this application.

9 CONCLUSION

A practical approach to loudspeaker transfer function modelling using musical excitation signals, continuously updated to incorporate time dependent nonlinearities has been developed. The approach is a genuinely black box technique that does not require prior knowledge of any parameters and can therefore be applied universally. The acquisition of training data through excitation with a music signal was necessary in order to facilitate the adaptation of the ANN model to alterations in the loudspeaker transfer function during operation. The incorporation of time dependent nonlinearities will reduce distortion more efficiently, as the model is a closer approximation to the loudspeaker's current behaviour than that of a generalised stationary model. The model output showed good correlation to actual loudspeaker output. The modelling performance of the loudspeaker frequency response and harmonic distortion was also good when the most nonlinear data available were selected as training data for the model. However, the data used in this experiment was collected over a relatively short time period. It is possible that over longer time periods the response of the loudspeaker will alter more radically and thus substantially larger alterations to the weights will be required, which may be more difficult to achieve over the target training period of 5 minutes.

ANNs were also applied to loudspeaker fault detection. Initial experiments involved training ANNs with data from the end of line rub and buzz test in order to emulate the output of the test i.e. to provide an output that indicates whether the loudspeaker should be accepted or rejected. In order to develop the fault detection scheme an investigation was conducted to ascertain if the ANN could be more specific and give an output relating to the frequency at which the loudspeaker was being rejected, as this could be indicative of the loudspeaker's defect. This resulted in a reasonable correlation between desired output and actual ANN output when the frequency range was split into three and five bands. Employing multiple ANNs to produce a vector describing the frequency bands in which the loudspeaker was rejected resulted in a significant improvement in accuracy in comparison to the multiple output ANN. The output was also significantly more sophisticated as loudspeaker patterns with values above the limits in any combination of the frequency bands could be analysed by the multiple ANN system. This system therefore may be considered to be an improvement upon the current end of line test.

APPENDIX 1
NOMENCLATURE

Term	Definition
A	arbitrary amplitude of cone displacement
B	magnetic flux density
C_{AB}	compliance of the air in the enclosure
C_M	compliance of the suspension system
C_{Mp}	compliance of combined centre and edge passive radiator suspensions
C_{MS}	compliance of combined centre and edge driver suspensions
CCD	charge coupled device
$d_j(n)$	desired output of neuron j
e_g,	source voltage
e_{rms}	root mean square error
$e_j(n)$.	error between desired and actual output of neuron j
F_m	suspension force factor
H_n	nth order Volterra operator.
i	voice coil current
k	stiffness of the loudspeaker suspension
l	length of voice coil
L_e	electrical inductivity of voice coil
m	total number of inputs (excluding the bias) applied to neuron j
m	mass of the cone
M_{AP}	acoustic mass of port or passive radiator including air load
M_{MP}	mass of the passive radiator
M_{MS}	combined mass of the driver diaphragm and wire on the voice coil, and are the and the compliance and mechanical responsiveness of the combined centre and edge suspensions of the passive radiator respectively
n	nth training data pattern
N	number of samples in the training set
p_g	acoustic driving pressure
r_{AB}	acoustic responsiveness due to internal energy absorption within the enclosure
R_{AB}	acoustic resistance due to internal energy absorption within the enclosure
r_{AL}	acoustic responsiveness due to losses caused by leakage
R_{AL}	acoustic resistance due to losses caused by leakage
R_{AP}	acoustic resistance of port or passive radiator losses

R_E	dc resistance of driver voice coil
R_g	output resistance of source
r_M	mechanical resistance due to dissipation in the air load
R_M	mechanical damping of loudspeaker system
r_{MP}	mechanical responsiveness of combined centre and edge passive radiator suspensions
r_{MS}	mechanical responsiveness of combined centre and edge driver suspensions
S_D	effective surface area of loudspeaker diaphragm
SPL	sound pressure level
t	time
$v_j(n)$	induced local field
$w_{ji}(n)$	synaptic weight connecting the output of neuron i to the input of neuron j
$w_{jc}(n)$	synaptic weight connecting the output of context unit c to neuron j.
w_{j0}	bias applied to neuron j
w^*	weight vector of an optimal solution
x	displacement
$y_c(k)$	context unit activation
$y_i(n)$	signal at the output of neuron i
$Z_{AS}(s)$	impedance of the driver branch
$Z_{AB}(s)$	impedance of the branch representing the enclosure interior
Z_{AA}	impedance of branch representing any enclosure apertures excluding that of the driver
α	feedback gain of the self-connections
$\delta_j(n)$	local gradient
ε_{av}	average error energy
$\varepsilon(n)$	total error energy
$\varphi_j(\cdot)$	activation function
μ	momentum coefficient
η	learning rate coefficient
ω	angular velocity
$\Delta w_{ji}(n)$	weight change from iteration n
∇	gradient operator
$\nabla\varepsilon_{av}(w)$	gradient operator of the average error energy function

APPENDIX 2
LOUDSPEAKER FAULTS

Identification Number	Fault Description
1	Solder on coil
2	Adhesive on coil
3	Loose turns
4	Coil damage
5	Overwinding
6	Damaged former
7	Unstuck pucara paper
8	Oval coil
9	Unstuck coil to O C D
10	Mounting up incorrect
11	Butt solder joint faults
12	Centre termination faults
13	Tag / tagging
14	Damaged yoke
15	Front plate burred
16	Pole of centre to front plate
17	Unstuck suspension
18	Twisted O C D
19	Sunken O C D
20	Coil off centre
21	Dymax in gap
22	Moyen in gap
23	Filings – plating
24	Filings – magnet particles
25	Staking faults
26	Unstuck cone / surround
27	Unstuck surround / chassis
28	Damaged chassis
29	Chassis tizz
30	Tag tizz
31	P.C.I. (tilted or sunk)
32	Cone off centre
33	Damaged cone
34	Excess adhesive under cone
35	Adhesive in wrong position under cone
36	Two cones fitted
37	Two O C D's fitted
38	Edge buzz
39	Excessive adhesive on speaker

Identification Number	Fault Description
80	Scrim faults
81	Bits under scrim
82	Supplier related reject speakers

APPENDIX 3
MULTI LAYER PERCEPTRON
C++ CODE

```
/******************************************************
      Backpropagation NN for identification of MIMO system
      2 hidden layer, hyperbolic tangent active function
                              BPMLP1
******************************************************/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

#define RMS     1000
#define RN      5
#define RC      0         //1--recurrent,  else dependent
#define TN      49996     //Training number
#define itera   5000
#define IN      9                     //input number to buffer layer
#define ON      1                     //output
#define LR      0.00001
#define MM      0.001
#define Rlr     0.9                   //Dynamic rate of learning rate
#define DegN    10                    //degradation number
#define Nv      10              //Interval to show validation error
#define SC      (2.4/IN)
#define TF      "T 4IO NL training data 29-07-05 TN49996.txt"
                                                //training data file
#define WGT     "Bpwt1 T NL 50-20-1 05-08-05.txt"
                                                //trained weights
#define OWT     "Bpwt1 T NL 50-20-1 05-08-05.txt"      //old weights
#define RMSF    "Bprms1 T NL 50-20-1 05-08-05.txt"
                                                //rms in training
#define n0       IN                         //inputs to buffer layer
#define n1       20                       //1st hidden layer neurons
#define n2       5                          //2ed hidden layer neuron
#define n3       ON                         //output layer neuron
#define rmin     (1.0e-09)                  //target
#define RM       RAND_MAX

static double w1[n1][n0+1],w2[n2][n1+1],w3[n3][n2+1];
static double ot0[n0],ot1[n1],ot2[n2],ot3[n3],lr=LR,mm=MM,
ym[n3],er[n3];
static double pdw1[n1][n0+1],pdw2[n2][n1+1],pdw3[n3][n2+1];
static double dw1[n1][n0+1],dw2[n2][n1+1],dw3[n3][n2+1],x;

void main()
{
      char type;
      int i,j;
      void learn();

      FILE *fp;
```

```
    printf("Continue to train (c) or\n start to train (s)?\n");
    type=getch();
    if(type=='s')
      {
         for(i=0;i<n1;i++)
           for(j=0;j<n0+1;j++)
                {
                w1[i][j]=SC*2.0*((double)rand()/RM-0.5);
                pdw1[i][j]=0.0;
                }

         for(i=0;i<n2;i++)
           for(j=0;j<n1+1;j++)
           {
             w2[i][j]=SC*2.0*((double)rand()/RM-0.5);
             pdw2[i][j]=0.0;
           }

         for(i=0;i<n3;i++)
           for(j=0;j<n2+1;j++)
           {
             w3[i][j]=SC*2.0*((double)rand()/RM-0.5);
             pdw3[i][j]=0.0;
           }
         printf("Start training now!\n");
      }
    else if(type=='c')
           {
           fp=fopen(OWT,"r");
           for(i=0;i<n1;i++)
             for(j=0;j<n0+1;j++)
                {
                fscanf(fp,"%le\t",&x);
                w1[i][j]=x;
                pdw1[i][j]=0.0;
                }

           for(i=0;i<n2;i++)
             for(j=0;j<n1+1;j++)
                {
                fscanf(fp,"%le\t",&x);
                w2[i][j]=x;
                pdw2[i][j]=0.0;
                }

           for(i=0;i<n3;i++)
             for(j=0;j<n2+1;j++)
                {
                fscanf(fp,"%le\t",&x);
                w3[i][j]=x;
                pdw3[i][j]=0.0;
                }
```

```
            fclose(fp);
            printf("Continue training now!\n");
            }
          else { printf("Stop!\n"); exit(0);}
     learn();
}

/*************************/
/*****  NET_OUT     ******/
void net_out()
{
     int i,j,k,l;
     double sa,sb;

     for(j=0;j<n1;j++)
       {
          ot1[j]=0.0;
          for(i=0;i<n0+1;i++)
            if(i<n0)   ot1[j]+=w1[j][i]*ot0[i];
            else       ot1[j]+=w1[j][i];
          sa=exp(ot1[j]);
          sb= 1.0/sa;
          ot1[j]=(sa-sb)/(sa+sb);
       }

     for(k=0;k<n2;k++)
       {
         ot2[k]=0.0;
         for(j=0;j<n1+1;j++)
           if(j<n1)   ot2[k]+=w2[k][j]*ot1[j];
           else       ot2[k]+=w2[k][j];
         sa=exp(ot2[k]);
         sb= 1.0/sa;
         ot2[k]=(sa-sb)/(sa+sb);
       }

     for(l=0;l<n3;l++)
       {
         ym[l]=0.0;
         for(k=0;k<n2+1;k++)
           if(k<n2) ym[l]+=w3[l][k]*ot2[k];
           else     ym[l]+=w3[l][k];
       }
}
/***************************/
/*******     TRAIN    *******/
void train()
{
     int i,j,k,l;
     double sum1[n1],dt1[n1],sum2[n2],dt2[n2];

     for(l=0;l<n3;l++)
```

```
        for(k=0;k<n2+1;k++)
          {
            if(k==n2) dw3[l][k]=er[l]*lr+mm*pdw3[l][k];
            else      dw3[l][k]=er[l]*lr*ot2[k]+mm*pdw3[l][k];
          }

      for(k=0;k<n2;k++)
        {
          sum2[k]=0.0;
          for(l=0;l<n3;l++) sum2[k]+=er[l]*w3[l][k];
          dt2[k]=(1.0+ot2[k])*(1.0-ot2[k])*sum2[k];
          for(j=0;j<n1+1;j++)
            {
            if(j==n1)
              dw2[k][j]=lr*dt2[k]+mm*pdw2[k][j];
            else
              dw2[k][j]= lr*dt2[k]*ot1[j]+mm*pdw2[k][j];
            }
        }


      for(j=0;j<n1;j++)
        {
           sum1[j]=0.0;
           for(k=0;k<n2;k++) sum1[j]+=dt2[k]*w2[k][j];
           dt1[j]=(1.0+ot1[j])*(1.0-ot1[j])*sum1[j];
           for(i=0;i<n0+1;i++)
             {
             if(i==n0)
               dw1[j][i]=lr*dt1[j]+mm*pdw1[j][i];
             else
               dw1[j][i]=lr*dt1[j]*ot0[i]+mm*pdw1[j][i];
             }
        }
}


//*****************************************
       //SAVE WEIGHT
//*****************************************
void save_w()
{
  int i,j;
      for(i=0;i<n3;i++)
        for(j=0;j<n2+1;j++)
          {
             w3[i][j]+=dw3[i][j];
             pdw3[i][j]=dw3[i][j];
          }

      for(i=0;i<n2;i++)
        for(j=0;j<n1+1;j++)
```

```
            {
              w2[i][j]+=dw2[i][j];
              pdw2[i][j]=dw2[i][j];
            }

      for(j=0;j<n1;j++)
        for(i=0;i<n0+1;i++)
          {
             w1[j][i]+=dw1[j][i];
             pdw1[j][i]=dw1[j][i];
          }
}

/*****************************/
/******     LEARN      *******/
void learn()
{
      unsigned int i;
      int j, flaga=0, flagb=0;
      double r,y[ON], errorN, errorD,yt;
      double yn[RN],e,ey,rms=RMS;
      unsigned long int k;
      FILE *fi,*fp,*fb;

      fi=fopen(TF,"r");
      mm=MM;
      for(k=0;k<itera; k++){
          e=ey=0.0;
          rewind(fi);
            for(i=0;i<TN;i++) {
                  for(j=0;j<IN+ON;j++) {
                        fscanf(fi,"%le\t",&x);
                        if(j<ON) y[j]=x;
                        else ot0[j-ON]=x;
                  }
                  if(RC==1)
                        if(i>=RN)
                              for(j=0;j<RN;j++)
                                    ot0[j]=yn[j];
                  net_out();

                  if(RC==1) {
                        for(j=RN-1;j>0;j--)
                              yn[j]=yn[j-1];
                        yn[0]=ym[0];
                  }
                  for(j=0;j<ON;j++)
                        er[j]=y[j]-ym[j];
                  train();
                  if(i<TN-1)  save_w();

                  for(j=0;j<ON;j++) {
                        e+=(y[j]-ym[j])*(y[j]-ym[j]);
                        ey+=y[j]*y[j];
                }
        }

        r=sqrt(e/ey);
          fp=fopen(RMSF,"a");
          fprintf(fp,"%lu\t%le\t%le\t%le\n",k,r,lr,mm);
          fclose(fp);

          if(k%Nv==0) {
                rewind(fi);
                errorN=0.0; errorD=0.0;
                for(i=0;i<TN;i++) {
                        for(j=0;j<IN+ON;j++) {
                                fscanf(fi,"%le\t",&x);
                                if(j<ON) yt=x;
                                else ot0[j-ON]=x;
                        }
                        net_out();

                        errorN+=(yt-ym[0])*(yt-ym[0]);
                        errorD+=yt*yt;
                }
                errorN=sqrt(errorN/errorD);
          }

          if(r>rms) flaga+=1;
          if(flaga==DegN){
                lr=Rlr*lr;
                flaga=0;
          }

/*        if(r<rms) flagb+=1;
          if(flagb==3){
                lr=lr/Rlr;
                flagb=0;
          }*/

        if(r<rms){
                mm=MM;
                rms=r;

                fb=fopen(WGT,"w");

                fprintf(fb,"rms=%le\n",rms);
                for(i=0;i<n1;i++)
                        for(j=0;j<n0+1;j++){
                                fprintf(fb,"%le\t",w1[i][j]);
                                if(j==n0) fprintf(fb,"\n");
                        }
```

```
            for(i=0;i<n2;i++)
                  for(j=0;j<n1+1;j++){
                        fprintf(fb,"%le\t",w2[i][j]);
                        if(j==n1) fprintf(fb,"\n");
                  }

            for(i=0;i<n3;i++)
                  for(j=0;j<n2+1;j++){
                        fprintf(fb,"%le\t",w3[i][j]);
                        if(j==n2) fprintf(fb,"\n");
                  }
            fclose(fb);
      }
        else mm=0.0;

      printf("k=%lu\trms=%le\ttrms=%le\tlr=%le\n",k,rms,r,lr);
        if(k%Nv==0) printf("Error=%le\n",errorN);
      if(rms<rmin) k=itera;
   }
}
```

APPENDIX 4
MULTI LAYER PERCEPTRON
VALIDATION
C++ CODE

```
/*************************************
   Validation of BPMLP hyperbolic tangent
       2-hidden-layer MISO systems
 ************************************/
#include<stdlib.h>
#include<dos.h>
#include<stdio.h>
#include<math.h>

#define IN  9
#define ON  1
#define n0  IN
#define n1  50
#define n2  20
#define WGT "Bpwt1 T NL 50-20-1 05-08-05.txt"

#define INPUT  "V 4IO NL training data 29-07-05 TN15532.txt"
#define OUTPUT "RV Bpwt1 V NL 50-20-1 05-08-05.txt"
#define VN      15532 /*sample number*/

main ()
{
  int i,j,m,l;
  static double yt,xt,xd,y,yn;
  static double ot0[IN],ot1[n1],ot2[n2],ym,sa,sb,e,ey;
  static double x,wt1[n1][IN+1],wt2[n2][n1+1],wt3[n2+1];
  FILE *fp,*fb;

  fp=fopen(WGT,"r");
  if (fp==NULL)
    {
      printf("No weight file!\n");
      exit(0);
    }

  for(i=0;i<n1;i++)
    for(j=0;j<IN+1;j++)
      {
        fscanf(fp,"%le\t",&x);
        wt1[i][j]=x;
      }
  for(i=0;i<n2;i++)
    for(j=0;j<n1+1;j++)
      {
        fscanf(fp,"%le\t",&x);
        wt2[i][j]=x;
      }
  for(i=0;i<n2+1;i++)
    {
      fscanf(fp,"%le\t",&x);
      wt3[i]=x;
    }
```

```
fclose(fp);

   fp=fopen(INPUT,"r");
   if (fp==NULL)
     {
     printf("No data file!\n");
     exit(0);
     }

   fb=fopen(OUTPUT,"w");
   if (fb==NULL)
     {
     printf("No output file!\n");
     exit(0);
     }

   e=0.0; ey=0.0;
    for(i=0;i<VN;i++)
     {
          for(j=0;j<IN+ON;j++)
                {
                fscanf(fp,"%le\t",&x);
                if(j<ON) yt=x;
                else ot0[j-ON]=x;
                }
         for(l=0;l<n1;l++)
           {
              ot1[l]=0.0;
              for(m=0;m<IN+1;m++)
                      {
                      if(m<IN)  ot1[l]+=ot0[m]*wt1[l][m];
                      else      ot1[l]+=wt1[l][m];
                      }
                sa=exp(ot1[l]);
              sb=1.0/sa;
              ot1[l]=(sa-sb)/(sa+sb);
           }

         for(l=0;l<n2;l++)
           {
           ot2[l]=0.0;
           for(m=0;m<n1+1;m++)
                {
                if(m<n1)  ot2[l]+=ot1[m]*wt2[l][m];
              else      ot2[l]+=wt2[l][m];
                }
           sa=exp(ot2[l]);
           sb=1.0/sa;
           ot2[l]=(sa-sb)/(sa+sb);
           }
         ym=0.0;
         for(l=0;l<n2+1;l++)
```

```
            {
            if(l<n2)   ym+=ot2[l]*wt3[l];
            else       ym+=wt3[l];
            }
          e+=(yt-ym)*(yt-ym);
            ey+=yt*yt;

      fprintf(fb,"%lf\t%lf\n",ym,yt);
     }
    e=sqrt(e/ey);
    printf("rms=%le\n",e);
    fclose(fp);
    fclose(fb);
    getchar();
    return(0);
}
```

APPENDIX 5
MODIFIED ELMAN
C++ CODE

```
/*****************************************************

      Modified Elman Recurrent Backpropagation NN
      for identification of MISO system 1 hidden layer,
      hyperbolic tangent active function
      Integral Validation Code
                  BPME1
*******************************************************/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>

#define ALFA   0.3
#define MRMS   1000
#define RMS    1000
#define ERR    1000                      /*Initial error value*/
#define TN     51100                     /*Training number*/
#define VN     25600                     /*Validation  number*/
#define EPOCH  375
#define IN     1                  /*input number to buffer layer*/
#define ON     1                         /*output*/
#define LR     0.00001       /*normal unit weight learning rate*/
#define LRC    0.000001      /*context unit weight learning rate*/
#define MM     0.75
#define Rlr    0.9                /*Dynamic rate of learning rate*/
#define DegN   10                        /*degradation number*/
#define SC     (2.4/IN)
#define TF     "TDATA P TN 51100 01-10-04 setn100 divn500b.txt"
                                         /*training data file*/
#define VALID  "VDATA P TN25600  01-10-04 setn100 divn1000.txt"
                                         /*validation data file*/
#define WGT    "Bpmeqwt1 5min OW TN51100 EP375.txt"
                                         /*trained weights*/
#define OWT    "Bpmewt1 5min RW TN51100 EP375.txt"
                                               /*old weights*/
#define VRMS   "Bpmeq1 5min OW TN51100 EP375.txt"
#define TRMS   "Bpmeq1 5min OW TN51100 EP375.txt"
#define VF     "Vq 5min OW TN51100 EP375.txt"
#define n2      ON                /*output layer neurons*/
#define n1      3                 /*1st hidden layer neurons*/
#define CON       n1              /*Number of context units*/
#define n0      IN+CON                   /*input layer*/
#define rmin    (1.0e-09)                /*target*/
#define RM      RAND_MAX

static double w1[n1][n0+1],w2[n2][n1+1];
static double ot0[n0],ot1[n1],ynet,ydes,lr=LR,lrc=LRC,mm=MM,er,r,rt,
minrms=MRMS,rtlow=ERR;
static double pdw1[n1][n0+1],pdw2[n2][n1+1];
static double dw1[n1][n0+1],dw2[n2][n1+1],x;
static double cont[CON];

void main()
{
      char type;
      int i,j;
      void learn();

      FILE *fp;

      printf("Continue to train (c) or\n start to train (s)?\n");
      type=getch();
      if(type=='s')
        {
           for(i=0;i<n1;i++)
             for(j=0;j<n0+1;j++)
                 {
                 w1[i][j]=SC*2.0*((double)rand()/RM-0.5);
                                         /*initialise weights*/
                 pdw1[i][j]=0.0;
                 }
           for(i=0;i<n2;i++)
             for(j=0;j<n1+1;j++)
                 {
               w2[i][j]=SC*2.0*((double)rand()/RM-0.5);
               pdw2[i][j]=0.0;
                 }

            for(i=0;i<CON;i++)
                             /*Initialise context units to 0.0*/
                 {
                 cont[i]=0.0;
                 }
           printf("Start training now!\n");
        }
      else if(type=='c')
             {
             fp=fopen(OWT,"r");
             for(i=0;i<n1;i++)
               for(j=0;j<n0+1;j++)
                  {
                  fscanf(fp,"%le\t",&x);
                  w1[i][j]=x;
                  pdw1[i][j]=0.0;
                  }

             for(i=0;i<n2;i++)
             for(j=0;j<n1+1;j++)
                  {
                  fscanf(fp,"%le\t",&x);
                  w2[i][j]=x;
                  pdw2[i][j]=0.0;
```

```
                        }
            for(i=0;i<CON;i++)
                                    /*Initialise context units to 0.0*/
                  {
                  cont[i]=0.0;
                  }

             fclose(fp);
             printf("Continue training now!\n");
             }
           else { printf("Stop!\n"); exit(0);}
      learn();
}

/*************************/
/*****  NET_OUT     ******/
void net_out()
{
int i,j,k,l;
double sa,sb;

for(i=0;i<n1;i++)
{
      ot1[i]=0.0;
      for(j=0;j<n0+1;j++)
      {
            if(j<n0)  ot1[i]+=w1[i][j]*ot0[j];
          else      ot1[i]+=w1[i][j];
      }
          sa=exp(ot1[i]);
          sb= 1.0/sa;
          ot1[i]=(sa-sb)/(sa+sb);
}

for(k=0;k<n2;k++)
{
      ynet=0.0;
      for(l=0;l<n1+1;l++)
      {
            if(l<n1) ynet+=w2[k][l]*ot1[l];
          else     ynet+=w2[k][l];
      }
}

}

/****************************/
/*******     TRAIN    *******/
void train()
{
int i,j,k,l;
double sum1[n1],dt1[n1];
```

```
for(l=0;l<n2;l++)
{
      for(k=0;k<n1+1;k++)
/*The formula for weight change is learning rate x local gradient x
neuron output plus momentum term*/

            if(k==n1) dw2[l][k]=er*lr+mm*pdw2[l][k];
/*Local gradient in output layer is error x derivative of activation
function, linear activation function so fi diff=1 therefore local
gradient = error*/
            else      dw2[l][k]=er*lr*ot1[k]+mm*pdw2[l][k];

      }

}
for(j=0;j<n1;j++)
{
      sum1[j]=0.0;
      for(k=0;k<n2;k++)
/*Local gradient for hidden neuron is sum of the product of local
gradients for proceeding layer and weight values, multiplied by the
derivative of the activation function*/
/*The local gradient in the proceeding layer is determined from sum
of errors and derivative of linear activation function (=1) therefore
just utilise error values*/
      {
      sum1[j]+=er*w2[k][j];
/*This is only valid for linear activation function in output layer*/

      }

      dt1[j]=(1.0+ot1[j])*(1.0-ot1[j])*sum1[j];
/*Derivative of hyberbolic tangent function multiplied by local
gradient*/

      for(i=0;i<n0+1;i++)
      {
            if(i==n0) dw1[j][i]=lr*dt1[j]+mm*pdw1[j][i];
                                                  /*bias calculation*/
            else if(i==0) dw1[j][i]=lr*dt1[j]*ot0[i]+mm*pdw1[j][i];
            /*input neuron weight adjustment - assumes only 1 input*/
            else dw1[j][i]=lrc*dt1[j]*ot0[i]+mm*pdw1[j][i];
                                  /*context unit weight adjustment*/
      }
}
}


/************************/
/*    ADJUST WEIGHT     */
/************************/
void adjust_w()
{
  int i,j;
```

```
      for(i=0;i<n2;i++)
            {
            for(j=0;j<n1+1;j++)
                  {
/*The weights are updated and previous change in weight saved for use
in the momentum term of the next forward activation*/
                  w2[i][j]+=dw2[i][j];
                  pdw2[i][j]=dw2[i][j];
                  }
            }

      for(j=0;j<n1;j++)
        for(i=0;i<n0+1;i++)
          {
             w1[j][i]+=dw1[j][i];
             pdw1[j][i]=dw1[j][i];
          }
}
/********************************/
/*          SAVE STATE          */
/********************************/
void save_s()
{
int i;
/*Saves current activations of hidden layer for use in next
iteration*/

for(i=0;i<CON;i++)
{
      cont[i]=ALFA*cont[i]+ot1[i];
}
}

/*******************************/
/*     ERROR CALCULATION       */
/*******************************/
void errorcalc()
{
int i,j,k,m;
double e,ey,input,vout[VN][3],contv[CON];

FILE *fa,*fb;

fa=fopen(VALID,"r");
if(fa==NULL)
{
      printf("No validation data file\n");
      exit(0);
}

e=0.0;                              /*Initialise error values to 0*/
ey=0.0;
```

```
rewind(fa);
for(i=0;i<VN;i++)
{
      for(j=0;j<IN+ON;j++)
      {
            fscanf(fa,"%le\t",&x);
/*Scan from validation data file the inputs and desired output*/
            if(j<ON) ydes=x;              /*assumes only 1 output*/
            else     input=x;             /*assumes inly 1 input*/
      }
      if(i==0)
      {
            for(j=0;j<CON;j++)
                  {
                        contv[j]=cont[j];
/*validation context units take on value of training context units -
as lr is small they should be acceptable*/
                  }
      }
      else
      {
            for(j=0;j<CON;j++)
/*Saves previous activation of hidden layer in context units for all
other iterations*/
            {
                  contv[j]=ALFA*contv[j]+ot1[j];
            }
      }
      for(k=0;k<IN+CON;k++)
/*Combines input units with saved context units*/
            {
                  if(k<IN) ot0[k]=input;
                  else    ot0[k]=contv[k-IN];
            }
      net_out();

      for(m=0;m<3;m++)
/*assuming 1 input and output*/
      {
            if(m==0) vout[i][m]=input;
            if(m==1) vout[i][m]=ydes;
            if(m==2) vout[i][m]=ynet;
      }

      for(j=0;j<ON;j++)
/*Calculate error for validation data*/
      {
            e+=(ydes-ynet)*(ydes-ynet);
            ey+=ydes*ydes;
      }
}
fclose(fa);
```

```
r=sqrt(e/ey);

if(r<minrms)                          /*Stores validation results*/
{
      fb=fopen(VF,"w");
      minrms=r;
      for(i=0;i<VN;i++)
      {

      fprintf(fb,"%le\t%le\t%le\n",vout[i][0],vout[i][1],vout[i][2]);
      }
      fclose(fb);
}
}
/*****************************/
/*****    TRAIN RMS     *****/
void trainrms()
{
int k,j,i;
double et,eyt,xin[IN],contr[CON];

FILE *fe,*fh;
/*Calculate rms error for training data at the end of each epoch*/
fh=fopen(TF,"r");
rewind(fh);
et=0.0;
eyt=0.0;


for(k=0;k<TN;k++)
{
      for(i=0;i<IN+ON;i++)                  /*scan input*/
      {
            fscanf(fh,"%le\t",&x);
            if(i<ON) ydes=x;                /*assumes 1 output*/
            else   xin[i-ON]=x;
      }

      if(k==0)
      {
            for(j=0;j<CON;j++)
                  {
                        contr[j]=cont[j];
/*validation context units take on value of training context units -
as lr is small they should be acceptable*/
                  }
      }
      else
      {
            for(j=0;j<CON;j++)
/*Saves previous activation of hidden layer in context units for all
other iterations*/
```

```
                {
                        contr[j]=ALFA*contr[j]+ot1[j];
                }
        }
        for(i=0;i<IN+CON;i++)
/*combine input units with context units*/
        {
                if(i<IN) ot0[i]=xin[i];
                else   ot0[i]=contr[i-IN];
/*takes last value of context units - should this be rest to 0?*/
        }

        net_out();

        for(j=0;j<ON;j++)
        {
                et+=(ydes-ynet)*(ydes-ynet);
                eyt+=ydes*ydes;
        }
}
rt=sqrt(et/eyt);

if(rt<rtlow)
/*Stores weights resulting in lowest rms*/
{
        mm=MM;
        rtlow=rt;
        fe=fopen(WGT,"w");
        fprintf(fe,"trms=%le\n",rtlow);
        for(i=0;i<n1;i++)
        {
                for(j=0;j<n0+1;j++)
                {
                        fprintf(fe,"%le\t",w1[i][j]);
                        if(j==n0) fprintf(fe,"\n");
                }
        }
        for(i=0;i<n2;i++)
        {
                for(j=0;j<n1+1;j++)
                {
                        fprintf(fe,"%le\t",w2[i][j]);
                        if(j==n1) fprintf(fe,"\n");
                }
        }

        fclose(fe);
}


fclose(fh);
}
```

```
/******************************/
/******     LEARN      *******/
void learn()
{
int i,j,flagb=0,m;
double xin[IN],rms;
unsigned long int k,flaga=0;        /*so k can be extremely large*/

FILE *fc,*fd,*ff,*fg;

fc=fopen(TF,"r");
mm=MM;                                    /*Sets momentum rate*/
rms=RMS;                            /*Sets initial error value*/
for(m=0;m<EPOCH;m++)
{
      rewind(fc);
      for(k=0;k<TN;k++)
      {
            for(i=0;i<IN+ON;i++)                /*scan input*/
            {
                  fscanf(fc,"%le\t",&x);
                  if(i<ON) ydes=x;              /*assumes 1 output*/
                  else   xin[i-ON]=x;
            }     -

            for(i=0;i<IN+CON;i++)
/*combine input units with context units*/
            {
                  if(i<IN) ot0[i]=xin[i];
                  else   ot0[i]=cont[i-IN];
            }

            net_out();                    /*Forward activation*/

            for(j=0;j<ON;j++)
            {
                  er=ydes-ynet;     /*Calculates current error*/
            }

            train();          /*Calculates neurons local gradients*/

            save_s();
/*saves hidden neuron values in context units*/

            adjust_w();                         /*Adjusts weights */



/*          printf("epoch%d\tk=%lu\ter(k)=%le\tlr=%le\n",m,k,er,lr);
                                              /*Prints data to screen*/
      }
```

```
        trainrms();

        fg=fopen(TRMS,"a");
        fprintf(fg,"%d\t%lu\t%le\n",m,k,rt);
        fclose(fg);

        printf("epoch%lu\ttrms=%le\n",m,rt);

        if(m==EPOCH-1)
        {
                errorcalc();

                ff=fopen(VRMS,"a");
                                        /*Writes validation results to file*/
                fprintf(ff,"%d\t%lu\t%le\n",m,k,r);
                fclose(ff);

                printf("epoch%lu\ttrms=%le\tvrms=%le\n",m,rt,r);
        }
}
fclose(fc);
printf("min trms=%le\tmin vrms=%le\n",rtlow,minrms);
}
```

APPENDIX 6
DATA RESAMPLING
C++ CODE

```
/*******************************************************************/
/*Data Resampling Program    */
/****************************************************************/


#include<stdlib.h>
#include<dos.h>
#include<stdio.h>
#include<math.h>

#define IN  1
#define ON  1
#define TN  52203
#define DESRATE   2205        /*desired sample rate*/
#define ACTRATE   44100       /*present sample rate of data*/
#define INPUT  "lspk data 4a.txt"
#define SAMPLEDOUT "sampled 2205Hz lspk data 4a.txt"
#define REJECTOUT "reject lspk data 1a.txt"

static double data[TN][IN+ON];

void main ()
{
  int i,j;
  double x;
  void sample();

  FILE *fp;


     fp=fopen(INPUT,"r");

       if (fp==NULL)
       {
            printf("No data file!\n");
            exit(0);
       }

       for(i=0;i<TN;i++)
            {
            for(j=0;j<IN+ON;j++)
                  {
                  fscanf(fp,"%le\t",&x);
                  data[i][j]=x;
                  }
            }
       fclose(fp);

       sample();
}
```

```
void sample()
{
      int i,SR;
      FILE *fp,*fq;

      fp=fopen(SAMPLEDOUT,"a");
      fq=fopen(REJECTOUT,"a");

      SR=ACTRATE/DESRATE;

      for(i=0;i<TN;i++)
                        {
                        if(i%SR==0)

      fprintf(fp,"%le\t%le\n",data[i][0],data[i][1]);
                        else

      fprintf(fq,"%le\t%le\n",data[i][0],data[i][1]);
                        }

      fclose(fp);
      fclose(fq);

}
```

APPENDIX 7
SINEWAVE FOR
FREQUENCY ANALYSIS
C++ CODE

```
/****************************************************************/
/*Sine wave generator for frequency analysis of MISO BPMLP models*/
/****************************************************************/


#include <stdio.h>
#include <dos.h>
#include <stdlib.h>
#include <math.h>

#define FK  2000                       /*wave frequency Hz*/
#define FS  44100                      /*sample rate in Hz*/
#define N   88300                      /*sample length*/
#define MAG 5                          /*wave magnitude volts*/
#define M_PI       3.14159265358979    /*value of pi*/
#define tstep      1/FS
#define PO  4                          /*Previous outputs*/
#define     PI     4                   /*Previous inputs*/
#define IN  1
#define     ON     1
#define nOs IN                         /*input neurons SISO network*/
#define     n1     10                  /*hidden layer neurons SISO
network*/
#define     n2     5
#define     n0m    IN+PO+PI            /*input neurons MISO network*/
#define     n3     50         /*hidden layer neurons MISO network*/
#define     n4     20

#define SISOWGT   "Bpwt1 TF TDATA.txt"          /*SISO weight file*/
#define MISOWGT "Bpwt1 T NL 50-20-1.txt"        /*MISO weight file*/
#define OUT "R TF 2000Hz Bpwt1 T NL 50-20-1 05-08-05.txt"

static double     wave[N+PO],time[N+PO],pout[PO],ymout[N];
static double     swt1[n1][IN+1],swt2[n2][n1+1],swt3[n2+1];
static double     mwt1[n3][IN+PO+PI+1],mwt2[n4][n3+1],mwt3[n4+1];
static double     pot1[n1],pot2[n2],ot0[N][IN+PO+PI],ot1[n3],ot2[n4];
static double     sa,sb,x,y;

void main()
{
void timestep();
void sinwav();
void prevout();
void calcout();
void output();

timestep();
sinwav();
prevout();
calcout();
output();
```

```
}

/****Integer array****/
void timestep()
{
int i;
static double x[N];

for(i=0;i<N+PO;i++)
{
      x[i]=i;
      time[i]=x[i]*tstep;
}

}

/****Sine wave****/

void sinwav()
{
int i;

/*FILE *fa;

fa=fopen(SINWAVE,"w");*/

for(i=0;i<N+PO;i++)
{
      wave[i]=MAG*sin(time[i]*2*M_PI*FK);
      /*fprintf(fa,"%le\t%le\n",time[i],wave[i]);*/
}

/*fclose(fa);*/
}

/****Previous output generator****/

void prevout()
{
int i,j,k;

FILE *fa;

fa=fopen(SISOWGT,"r");                    /*Scan in weights*/
  if (fa==NULL)
    {
      printf("No weight file!\n");
      exit(0);
    }

  for(i=0;i<n1;i++)
    for(j=0;j<IN+1;j++)
```

```
        {
          fscanf(fa,"%le\t",&x);
          swt1[i][j]=x;
        }
    for(i=0;i<n2;i++)
      for(j=0;j<n1+1;j++)
        {
          fscanf(fa,"%le\t",&x);
          swt2[i][j]=x;
        }
    for(i=0;i<n2+1;i++)
      {
        fscanf(fa,"%le\t",&x);
        swt3[i]=x;
      }
    fclose(fa);

for(i=0;i<PO;i++)
{
      for(j=0;j<n1;j++)
      {
            pot1[j]=0.0;
            for(k=0;k<IN+1;k++)
            {
                  if(k<IN)  pot1[j]+=wave[i]*swt1[j][k];
                else      pot1[j]+=swt1[j][k];
            }
            sa=exp(pot1[j]);
            sb=1.0/sa;
            pot1[j]=(sa-sb)/(sa+sb);
      }
      for(j=0;j<n2;j++)
      {
            pot2[j]=0.0;
            for(k=0;k<n1+1;k++)
            {
                  if(k<n1)  pot2[j]+=pot1[k]*swt2[j][k];
                  else      pot2[j]+=swt2[j][k];
            }
            sa=exp(pot2[j]);
            sb=1.0/sa;
            pot2[j]=(sa-sb)/(sa+sb);
      }

      y=0.0;
      for(j=0;j<n2+1;j++)
      {
            if(j<n2)  y+=pot2[j]*swt3[j];
            else      y+=swt3[j];
      }

      pout[i]=y;
      printf("%le\t",pout[i]);
}
}

/****Lspk Output Calculation****/
void calcout()
{
int i,j,m,l,k;
double ym;

FILE *fa/*,*fp*/;

fa=fopen(MISOWGT,"r");                        /*Scan in weights*/
  if (fa==NULL)
    {
      printf("No weight file!\n");
      exit(0);
    }

  for(i=0;i<n3;i++)
    for(j=0;j<IN+PO+PI+1;j++)
      {
        fscanf(fa,"%le\t",&x);
        mwt1[i][j]=x;
      }
  for(i=0;i<n4;i++)
    for(j=0;j<n3+1;j++)
      {
        fscanf(fa,"%le\t",&x);
        mwt2[i][j]=x;
      }
  for(i=0;i<n4+1;i++)
    {
      fscanf(fa,"%le\t",&x);
      mwt3[i]=x;
    }
fclose(fa);

for(i=0;i<N;i++)
{
      for(k=0;k<IN+PO+PI;k++)
      {
      if(k%2==0)   ot0[i][k]=wave[i+(PI-(k/2))];
      else  ot0[i][k]=pout[PO-((k+1)/2)];
      }

      for(l=0;l<n3;l++)
      {
          ot1[l]=0.0;
          for(m=0;m<IN+PO+PI+1;m++)
            {
                  if(m<IN+PO+PI)     ot1[l]+=ot0[i][m]*mwt1[l][m];
```

```
                else            ot1[l]+=mwt1[l][m];
            }
            sa=exp(ot1[l]);
            sb=1.0/sa;
            ot1[l]=(sa-sb)/(sa+sb);
        }

        for(l=0;l<n4;l++)
        {
            ot2[l]=0.0;
            for(m=0;m<n3+1;m++)
            {
                if(m<n3)  ot2[l]+=ot1[m]*mwt2[l][m];
                else      ot2[l]+=mwt2[l][m];
            }
            sa=exp(ot2[l]);
            sb=1.0/sa;
            ot2[l]=(sa-sb)/(sa+sb);
        }
        ym=0.0;
        for(l=0;l<n4+1;l++)
        {
            if(l<n4)  ym+=ot2[l]*mwt3[l];
            else      ym+=mwt3[l];
        }

        ymout[i]=ym;

        for(j=0;j<PO;j++)
        {
            if(j<PO-1)  pout[j]=pout[j+1];
            else        pout[j]=ym;
        }
}


}

/****Generate output file****/
void output()
{
int i,j;

FILE *fa;

fa=fopen(OUT,"w");

for(i=0;i<N;i++)
{
        if(i>99)
            /*Skips first samples where convergence is occuring*/
        {
```

```
      for(j=0;j<IN+1;j++)
            {
            if(j<IN) fprintf(fa,"%le\t",ymout[i]);
            else  fprintf(fa,"%le\n",wave[i+PO]);
            }
      }
}
}
```

APPENDIX 8
DATA PATTERN
FAIL FREQUENCY LOCATOR

```
/************************************************/
/*************Training Data Selector*************/
/*************and fail frequency locator  *******/
/************************************************/

#include<stdlib.h>
#include<dos.h>
#include<stdio.h>
#include<math.h>
#include<conio.h>

#define IN  48                    /*Number of frequency points*/
#define ON  1                     /*Number of outputs*/
#define k   6
                  /*Number of standard deviations from test limits*/
#define TN  5973                        /*Number of data patterns*/
#define PN  0        /*Number of classified passes in sample*/
#define INPUT     "R&B REJ.txt"         /*File containing data*/
#define FAIL      "FAIL L+6s R&B REJ.txt"
                            /*File for data outside test limits*/
#define PASS      "PASS L+6s R&B REJ.txt"
                            /*File for data inside test limits*/
#define     MEAN  "MEAN R&B 99-100bm16.txt"
                                        /*File to record mean*/
#define     SD          "SDEV R&B 99-100bm16.txt"
                            /*File to record standard deviation*/
#define LIMITS    "LIMIT 17-09-02 to 11-10-02.txt"
                            /*File containing soundcheck limits*/

static float data[TN][IN+ON];
static float rms[IN+ON];
static float sdev[IN+ON];
static float limit[IN];
static float limithigh[IN+ON], limitlow[IN+ON];

void main()
{
      int i,j,m;
      float x,y;
      void standevcalc();
      void passfail();

      FILE *fp,*fq;

      fp=fopen(INPUT,"r");

      if(fp==NULL)
            /*Incase there is something wrong with input file */
            {
            printf("No input file!");
            exit(0);
            }
      for(i=0;i<TN;i++)
                  /*Reads data file  and enters them into an array */
            {
            for(j=0;j<IN+ON;j++)
                  {
                  fscanf(fp,"%e\t",&x);
                  data[i][j]=x;
                  }
            }

      fclose(fp);

      fq=fopen(LIMITS,"r");

     if (fq==NULL)
       {
            printf("No limits file!\n");
            exit(0);
       }

       for(m=0;m<IN;m++)
            {
            fscanf(fq,"%e\t",&y);
            limit[m]=y;
            }
       fclose(fq);

      standevcalc();
                        /*Calculates RMS and standard deviation*/

      passfail();
/*Calculates test limits and checks data to determine whether or not
the loudspeaker is within limits */

}

void standevcalc()
{
      int i,j;
      char type;
      float m,n,g,h,x,y;

      FILE *fp,*fq,/**fr,*/*fs;

      printf("Calculate new mean (n) or use previous (p)?");
      type=getch();

      if(type=='n')

      {

      fp=fopen(MEAN,"w");
      fq=fopen(SD,"w");
      for(j=1;j<IN+ON;j++)
/*Calculates the square of the value then sums the square values in
each column */
            {
            n=0;
/*Each column contains all the data samples for one frequency */
            for(i=0;i<TN;i++)
            {
                  if(data[i][0]==1)
/*Ensures only data classified as a pass is included in calculation
*/
                        {
                        m=data[i][j];
                        m=m*m;
                        n=n+m;
                        }
                  }
            rms[j]=sqrt(n/PN);
                                    /*Calculates the RMS for each column*/
            fprintf(fp,"%e\t",rms[j]);
            }

      for(j=1;j<IN+ON;j++)
            {
            y=0;
            for(i=0;i<TN;i++)
                  {
                  if(data[i][0]==1)
                        {
                        x=data[i][j]-rms[j];
                  /*Deducts the corresponding rms from each value*/
                        x=x*x;
                        y=y+x;
                        }
                  }
            sdev[j]=sqrt(y/PN);
                                          /*Calculates standard deviation*/
            fprintf(fq,"%e\t",sdev[j]);
            }
      fclose(fp);
      fclose(fq);
      }


      if(type=='p')
      {
/*    fr=fopen(MEAN,"r");*/
      fs=fopen(SD,"r");
      for(j=0;j<IN;j++)
/*Reads previously calculated mean and standard deviation from file
*/
```

```
            {
            /*fscanf(fr,"%e\t",&g);
            rms[j]=g;*/
            fscanf(fs,"%e\t",&h);
            sdev[j]=h;
            }
      /*fclose(fr);*/
      fclose(fs);
      }

}


void passfail()
{
      int i,j,fail;

      FILE *fp,*fq;

      for(j=1;j<IN+ON;j++)
      /*Calculates boundary*/
            {
            limithigh[j]=limit[j-1]+k*sdev[j];
            /*limitlow[j]=rms[j]-k*sdev[j];*/
            }
      fp=fopen(FAIL,"a");
      fq=fopen(PASS,"a");

      for(i=0;i<TN;i++)
/*Checks data against test limits and assigns value 1 to 'fail' if
any value in the row is outside limits */
            {
            fail=0;
            for(j=1;j<IN+ON;j++)
                  {
                  if(data[i][j]>limithigh[j]) fail=1;
                  /*if(data[i][j]<limitlow[j]) fail=1;*/
                  }

            if(fail==1)
                        /*If one or more values are outside test
limits the whole row will be sent to 'FAIL' file */
                  {
                  for(j=0;j<IN+ON;j++)
                        {
                        fprintf(fp,"%e\t",data[i][j]);
                        if(j==(IN+ON-1)) fprintf(fp,"\n");
                        }
                  }
            if(fail==0)
                  /*If not the row will be sent to the 'PASS' file */

                  {
                  for(j=0;j<IN+ON;j++)
```

```
                              {
                              fprintf(fq,"%e\t",data[i][j]);
                              if(j==(IN+ON-1)) fprintf(fq,"\n");
                              }
                        }
                  }
      fclose(fp);
      fclose(fq);
}
```

APPENDIX 9
REFERENCES

Adams, C.J., 1983, Adaptive Control of Loudspeaker Frequency Response at Low Frequencies, Presented at the 73rd Convention of the Audio Engineering Society, Eindhoven, The Netherlands.

Anthony, S., 2002. Project update. [Email] Personal email to Fox, C. (July 30th 2002)

Anthony, S. 2003. RE:Current Work. [Email] Personal email to Fox, C. (June 25th 2003)

Bailey, S., Watton, J., 2002, A Neural Network Approach to Transmission Line Modelling for fault Diagnosis of a Hydraulic Pressure Control System, Proceedings IMechE Part I Journal of Systems and Control Engineering, Vol. 216, Part I, p. 357.

Benson, J.E., 1968, Theory and Design of Loudspeaker Enclosures, Part 1 – Electro-Acoustical Relations and Generalised Analysis, A.W.A. Tech. Rev., Vol. 14, pp 1-57.

Benson, J.E., 1972, Theory and Design of Loudspeaker Enclosures, Part 3 – Introduction to Synthesis of Vented Systems, A.W.A Tech. Rev., Vol. 14, pp 369.

Beranek, L.L., 1954, Acoustics, McGraw-Hill, New York.

Birt, D.R., 1991, Nonlinearities in Moving-Coil Loudspeakers with Overhung Voice Coils, J. Audio Eng. Soc., Vol. 39, No. 4.

de Boer, E., 1959, Acoustic Interaction in Vented Loudspeaker Enclosures, J. Acoust. Soc. Amer. (Letter), Vol. 31, pp 246.

Bodenhausen, U., Waibel, A., 1991, Learning the Architecture of Neural Networks for Speech Recognition, IEEE*

Caulton, C.O., Dickry, E.T., Perry, S.V., 1936, The Magic Voice, Radio Engineering, Vol. 16, Pt 10, pp 8-10, 22.

Chang, P., Lin, C.G., Yeh, B., 1994, Inverse Filtering of a Loudspeaker and Room Acoustics using Time-Delay Neural Networks, J. Audio Eng. Soc., Vol. 95, No. 6.

Chernof, J., 1957, Principles of Loudspeaker Design and Operation, IRE Transactions on Audio.*

Elliot, R., 2003, Measuring Loudspeaker Parameters, [WWW] <URL: http://sound.westhost.com/tsp/.htm [Accessed 12 March 2004].

Elman, J.L., 1990, Finding Structure in Time, Cognitive Science 14.

Fielder, L.D., Benjamin, E.M., 1987, Subwoofer Performance for Accurate Reproduction of Music, Presented at the 83rd Convention of the Audio Engineering Society, New York.

Foo, S.Y., Stuart, G., Harvey, B., Meyer-Baese, A., 2002, Neural network-based EKG pattern recognition, Engineering Applications of Artificial Intelligence, Vol 15, pp 253 – 260.

Frank, W., Reger, R., Appel, U., 1992, Loudspeaker Nonlinearities – Analysis and Compensation, Signals, Systems and Computers, 1992 Conference Record of the Twenty-Sixth Asilomar Conference on 26-28 October 1992, Vol. 2.

Fraser, N., 1998, [WWW] <URL: http://vv.carleton.ca/~neil/neural/neuron-a.html [Accessed 10 August 2005]

Gao, X.Y., Snelgrove, W.M., 1991, Adaptive Linearization of a Loudspeaker, Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing.

Greenfield, R., Hawksford, M.O., 1991, Efficient Filter Design for Loudspeaker Equalization. J. Audio Eng. Soc., Vol. 39, No. 10.

Greiner, R.A., Sims, T.M., 1983, Loudspeaker Distortion reduction, Presented at the 74th Convention of the Audio Engineering Society, New York.

Griesinger, D., 2004, Conference Workshop: Perception of Sound, Presented at the 116th Convention of the Audio Engineering Society, Berlin.

Harwood, H.D., 1972, Loudspeaker Distortion Associated with Low Frequency Signals, Presented at the 2nd Central Europe Convention of the Audio Engineering Society, Munich, Germany.

Haykin, S., 1999, Neural Networks A Comprehensive Foundation, Second Edition, Prentice Hall.

Hutt, B., 2002, Evolving Artificial Neural Network Controllers for Robots using Species-Based Methods, Thesis (PhD) – University of Reading.

Jack, L., Nandi, A., 2000, Genetic Algorithms for Feature Selection in Machine Condition Monitoring with Vibration Signals, IEE Proceedings of Visual Image Signal Processing, Vol. 147, No. 3, p. 205.

Kaizer, A.J.M., 1987, Modelling of the Nonlinear Response of an Electrodynamic Loudspeaker by a Volterra Series Expansion, J. Audio Eng. Soc., Vol. 35, No. 6.

Kalayci, T., Özdamar, Ö., 1995, Wavelet Preprocessing for Automated Neural Network Detection of EEG Spikes, IEEE Engineering in Medicine and Biology.

Kantz, H., Screiber, T., 2004, Nonlinear Time Series Analysis, Second Edition, Cambridge University Press.

Klippel, W., 1992a, Nonlinear Large-Signal Behaviour of Electrodynamic Loudspeakers at Low Frequencies, J. Audio Eng. Soc., Vol. 40, No. 6.

Klippel, W., 1992b, The Mirror Filter – A New Basis for Reducing Nonlinear Distortion and Equalizing Response in Woofer Systems, J. Audio Eng. Soc., Vol. 40, No. 9.

Klipsch, P.W., 1968, Modulation Distortion in Loudspeakers, Presented at the 34th Convention of the Audio Engineering Society, Los Angeles.

Lippmann, R., 1987, An Introduction to Computing with Neural Nets, IEEE ASSP Magazine.

Liu, X., 1993, System Identification and Prediction Using Neural Networks, PhD Thesis, University of Wales Cardiff.

Locanthi, B.N., 1952, Application of Electric Circuit Analogies to Loudspeaker Design Problems, Presented at the 1952 IRE National Convention, New York.

Low, S., Hawksford, M.O.J., 1993, A Neural Network Approach to the Adaptive Correction of Loudspeaker Nonlinearities, Presented at the 95th Annual Convention of the Audio Engineering Society, New York.

Lurette, C., Lecoeuche, S., 2003, Unsupervised and Auto-Adaptive Neural Architecture for On-Line Monitoring. Application to a Hydraulic Process, Engineering Applications of Artificial Intelligence Vol. 16, p. 441.

Marshall Leach, W., 1990, A Generalized Active Equalizer for Closed-Box Loudspeaker Systems, J. Audio Eng. Soc., Vol. 38, No. 3.

Mason, R., Ford, N., Rumsey, F., de Bruyn, B., 2001, Verbal and Nonverbal Elicitation Techniques in the Subjective Assessment of Spatial Sound Reproduction, J. Audio Eng. Soc., Vol. 49, No. 5, pp 366 - 384.

Medsker, L., Jain, L., 2000, Recurrent Neural Networks Design and Applications, CRC Press LLC.

Miller, K. A., 1999, Deeper understanding and enhancement of the Elman recurrent neural network, with context layer self-feedback, for the modelling of linear and non-linear dynamic systems. Thesis (Ph.D.) – University of Wales, Cardiff.

Mills, P.G.L.,Hawksford, M.O.J., 1989, Distortion Reduction in Moving-Coil Loudspeaker Systems Using Current-Drive Technology, J. Audio Eng. Soc., Vol. 37, No. 3.

Narendra, K.S., Parthasarathy, K, 1990, Identification and Control of Dynamical Systems Using Neural Networks, IEEE Transactins on Neural Networks, Vol. 1 No.1. pp 4 – 27.

Newman, T., 2004, Motional Feedback Report, Harman Becker Automotive Systems.

Nguyen, D., Widrow, B., 1990, Neural Networks for Self-Learning Control Systems, IEEE Control Systems Magazine 10, Vol. 3, p. 18.

Novak, J.F., 1959, Performance of Enclosures for Low Resonance High Compliance Loudspeakers, J. Audio Eng. Soc., Vol. 7, p. 29.

Olson, H.F., 1944, Action of a Direct Radiator Loudspeaker with Non-Linear Con Suspension, J. Amer. Stand. Assoc. Vol 16.

Olson, H.F., Analysis of the Effects of Nonlinear Elements Upon the Performance of a Back-Enclosed, Direct Radiator Loudspeaker Mechanism, Presented at 1960 Fall Convention of the Audio Engineering Society in New York and at 1961 Spring Convention in Los Angeles.

Pham, D.T., Bayro-Corrochano, E.J., 1994, Neural Classifiers for Automated Visual Inspection, Proceedings of the Institution of Mechanical Engineers, Vol 208, p. 83.

Pham, D.T., Liu, X., 1993, Identification of Linear and Nonlinear Dynamic Systems using Recurrent Neural Networks, Artificial Intelligence in Engineering.

Pham, D.T., Liu, X., 1995, Neural Networks for Identification, Prediction and Control, Springer.

Picton, P., 2000, Neural Networks, Second Edition, Palgrave.*ref 2 ANN theory

Psaltis, D., Sideris, A., Yamamura, A., 1988, A Multilayered Neural Network Controller, IEEE Control Systems Magazine, 8 Vol. 2, p. 17.

Rice, C.W., Kellogg, E.W., 1925, Notes on the Development of a New Type of Hornless Loud Speaker, Presented at the Spring Convention of the A.I.E.E., St. Louis.

Roder, H., 1936, The Theory of the Loudspeaker, Radio Engineering, Vol. 16, Pt 7, P10-12, 22, Pt 8, P21-25, Pt 9, P24-26, 29, Pt 10, P19-22.

Rossing, T.D., 1989, The Science of Sound, Second Edition, Addision-Wesley Publishing Company.

Rumelhart, D.E., McClelland, J.L, Hinton, Williams, 1986, Learning Internal Representations by Error Propagation, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, MIT Press, Cambridge.

Saerens, M., Soquet, A., 1991, Neural Controller Based on Back-Propagation Algorithm, IEE Proceedings Control Theory Applications, Vol. 138, No 1, p55.

Schetzen, M., 1989, Volterra and Weiner Theories of Nonlinear Systems, 2nd Edition, Robert E. Krieger Publishing Company.

Small, R.,1972a, Direct-Radiator Loudspeaker System Analysis, J. Audio Eng. Soc. Vol.20, No. 5, pp. 383 - 395.

Small, R., 1972b, Closed-Box Loudspeaker Systems Part 1 – Analysis, J. Audio Eng. Soc. Vol.20, No. 10, pp. 798 - 808.

Small, R., 1973a, Closed-Box Loudspeaker Systems Part 2 – Synthesis, J. Audio Eng. Soc. Vol.21, No. 1, pp. 11 - 18.

Small, R., 1973b, Vented-Box Loudspeaker Systems Part 1 – Small Signal Analysis, J. Audio Eng. Soc. Vol.21, No. 5, pp. 363 - 372.

Small, R., 1973c, Vented-Box Loudspeaker Systems Part 2 – Large Signal Analysis, J. Audio Eng. Soc. Vol.21, No. 6, pp. 438 - 444.

Small, R., 1973c, Vented-Box Loudspeaker Systems Part 2 – Large Signal Analysis, J. Audio Eng. Soc. Vol.21, No. 6, pp. 438 - 444.

Small, R., 1973e, Vented-Box Loudspeaker Systems Part 4 – Appendices, J. Audio Eng. Soc. Vol.21, No. 8, pp. 635 - 639.

Smith, L., 2002 [WWW] <URL: http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html [Accessed 12 September 2002].

Tan, C., Moore, B.C.J., Zacharov, N., 2003, The Effect of Nonlinear Distortion on the Perceived Quality of Music and Speech Signals, J. Audio Eng. Soc. Vol. 51, No. 11.

Thiele, A.N., 1961, Loudspeakers in Vented Boxes: Part 1, Presented at the 1961 I.R.E. Radio and Electronic Engineering Convention, Sydney. Proceedings of the I.R.E. Australia, Vol. 22. Reprinted in J. Audio Eng. Soc. 1971, Vol. 19, No. 5, p. 382-392.

Thuras, A.L., 1930, Sound Translating Devive, U.S. Patent 1,869,178, application Aug. 15th 1930; patented July 26th 1932.

Unknown, 2002 [WWW] <URL: http://blizzard.gis.uiuc.edu/htmldocs/Neural/neural.html [Accessed 12 September 2002].

Villchur, E.M., 1956, Problems of Bass Reproduction in Loudspeakers, Presented at the 8th Annual Convention of the Audio Engineering Society, New York.

Wang, D., Bao, P., 2000, Enhancing the Estimation of Plant Jacobian for Adaptive Neural Inverse Control, Neurocomputing, Vol. 34, p. 99.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K., 1989, Phenome Recognition Using Time-Delay Neural Networks, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 37, No. 3, p. 328.

Watton, J., Xue, Y., 1997, Simulation of Fluid Power Circuits using Artificial Network Models Part 1: Selection of Component Models, Proc. Instn Mech. Engnrs, Vol. 211, Part 1, pp. 417 – 428.