The Application of Artificial Neural Networks to Interpret Acoustic Emissions from Submerged Arc Welding

A Thesis submitted for the degree of Doctor of Philosophy

by

John Richard McCardle

Faculty of Technology, Department of Design
Brunel University

June 1997

" Research is what I'm doing when I don't know what I'm doing."

Wernher von Braun

Acknowlegements

A debt of gratitude is expressed to the many people who, over the last five years, have helped me to submitt this work (eventually!). Particular thanks, however, to the following :-

All the team at Brunel, Karen for the lack of pressure, Ray for many a midnight beer in a dodgy continental bar, Tom & Julie for being good neighbours : all the members of the old Neural Applications Group, Wilky, Chris, Lee, Mystic Malcolm : all the participants of the successful WINNER project, especially Dan for his wit and repartee, Stan for his rampant enthusiasm, Wolfy for his teutonic efficiency, John, Ian, Babs, Kawal, and Jose.

A very special thanks to Marius whose friendship, listening ear and a beer down the pub has helped no end.

My love goes to Shelley without whose unflagging patience, understanding and endless cups of tea, I would have been totally lost.

Dedicated to the memory of my father

Andrew Owen McCardle

28/4/22 - 3/11/93

SUMMARY.

Automated fusion welding processes play a fundamental role in modern manufacturing industries. The proliferation of joint geometries together with the large permutation of associated process variable configurations has given rise to research into complex system modelling and control strategies. Many of these techniques have involved monitoring of not only the electrical characteristics of the process but visual and acoustic information.

Acoustic information derived from certain welding processes is well documented as it is an established fact that skilled manual welders utilise such information as an aid to creating an optimum weld. The experimental investigation presented in this thesis is dedicated to the feasibility of monitoring airborne acoustic emissions of Submerged Arc Welding (SAW) for diagnostic and real time control purposes.

The experimental method adopted for this research takes a cybernetic approach to data processing and interpretation in an attempt to replicate the robustness of human biological functions.

A custom designed audio hardware system was used to analyse signals obtained from bead on mild steel plate fusion welds. Time and frequency domains were used in an attempt to establish salient characteristics or identify the signatures associated with changes of the process variables. The featured parameters were voltage / current and weld travel speed, due to their ease of validation. However, consideration has also been given to weld defect prediction due to process instabilities.

As the data proved to be highly correlated and erratic when subjected to off line statistical analysis, extensive investigation was given to the application of artificial neural networks to signal processing and real time control scenarios. As a consequence, a dedicated neural based software system was developed, utilising supervised and unsupervised neural techniques to monitor the process. The research was aimed at proving the feasibility of monitoring the electrical process parameters and stability of the welding process in real time. It was shown to be possible, by the exploitation of artificial neural networks, to generate a number of monitoring parameters indicative of the welding process state.

The limitations of the present neural method and proposed developments are discussed, together with an overview of applied neural network technology and its impact on artificial intelligence and robotic control. Further developments are considered together with recommendations for future areas of research.

Contents Page

Contents Page

Contents Page

Contents Page

Contents

9.0 APPENDICES

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACF	Auto Correlation Function
ADC	Analogue to Digital Converter
Adaline	Adaptive Linear Neuron
AI	Artificial Intelligence
ANN	Artificial Neural Network
ANSI	Americal National Standards Institute
ART	Adaptive Resonance Theory
ASCII	American Standard Code for Information Interchange
AWS	American Welding Society
BP	Backpropagation
BPN	Backpropagation Network
BSB	Brain-State-In-A-Box
CCD	Charged Coupled Device
CCF	Cross Correlation Function
CIM	Computer Integrated Manufacture
CPN	Counter Propagation Networks
CPU	Central Processing Unit
D	Derivative (control)
DAC	Digital to Analogue Converter
DCI	Dip Consistency Index
DFT	Discrete Fourier Transform
DOS	Disk Operating System
DSP	Digital Signal Processing
DWT	Discrete Wavelet Transform
ECM	Electret Condenser Microphone
FCAW	Flux Cored Arc Welding
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
GMAW	Gas Metal Arc Welding
GTAW	Gas Tungsten Arc Welding
HAZ	Heat Affected Zone
HMM	Hidden Markov Model
I	Integral (control)
i.c	Integrated Circuit
iDWT	Inverse Discrete Wavelet Transform
iFFT	Inverse Fast Fourier Transform
I/O	Input/Output

jnd	Just Noticeable Difference
k-NN	k- Nearest Neighbour
LMS	Least Mean Squared
LVQ	Linear Vector Quantisation
MAC	Multiply Accumulate Cycle
Madaline	Multi-Adaptive Linear Neuron
MAG	Metal Active Gas
MIAB	Magnetically Impelled Arc Butt
MIG	Metal Inert Gas
MLP	Muti-Layer Perceptron
NDT	Non Destructive Testing
P	Proportional (control)
PC	Personal Computer
p.d.f	Probability Density Function
PI	Proportional + Integral (control)
PID	Proportional + Integral +Derivative (control)
PR	Power Ratio
PR	Pattern Recognition
RBF	Radial Basis Function
RAM	Random Access Memory
r.m.s	Root Mean Squared
ROC	Receiver Operating Characteristics
ROM	Read Only Memory
SAW	Submerged Arc Welding
SOFM	Self Organising Feature Map
SOM	Self Organising Map
SPL	Sound Pressure Level
STN	Spatio Temporal Networks
TI	Transfer Index
TIG	Tungsten Inert Gas
TN	True Negative
TP	True Positive
TSI	Transfer Stability Index
VLSI	Very large Scale Integration

MATHEMATICAL TERMS

a_i	Value of neuron i
a_n	actual output value of neuron n
α_t	Learning rate at epoch t
α_0	Initial learning rate
c	Speed of Sound (m/s)
C_k	Fourier coefficient
cov(x,y)	Statistical covariance
d	Particle displacement (m)
d	Number of components in feature vector
d_n	Desired value of neuron n
|E|	Input vector
E	Energy (kJ/mm)
E_{mean}	Average error
E_{norm}	Normalised error
E_t	Total Error
γ	Specific heat ratio
η	learning coefficient
I	Sound intensity (W/m^2)
I	Current (A)
I_{min}	Minimum current (A)
I_{max}	Maximum current (A)
I_{mean}	Average current (A)
I_{bk}	Average current <= I_{mean} (A)
k	Harmonic number
N_{min}	Minimum number of samples
N_s	Number of samples in test set
O_n	Output of neuron n
O_{pl}	Measured value of neuron
θ_h	Hidden layer error
θ_n	Output layer error
p	Steady State pressure (N/m^2)
p	Probability
p(e)	Probability of error
ρ	Density (kg/m^3)
r_{xx}	Auto-correlation coefficient
r_{xy}	Cross-correlation coefficient
t	Present epoch
T	Total number of epochs
T_{pl}	Target value for neuron

Ts	Travel Speed (mm/s)
u	Particle velocity (m/s)
|U|	Weight vector
V	Voltage (V)
V_{min}	Minimum voltage (V)
V_{max}	Maximum Voltage (V)
V_{mean}	Average Voltage (V)
V_{bk}	Average Voltage <= V_{mean} (V)
var(x)	Statistical variance
W_{ni}	Synaptic weight value between neuron a_i and neuron O_n
ω_0	Fundamental frequency

1.0 INTRODUCTION

1.0 INTRODUCTION

Cybernetics, the development of machines based upon human replication, has inspired techniques now referred to as Artificial Intelligence (AI). Developments in this field have been limited by the technical and often commercial viability, hence availability of state machines on which to implement such systems.

Early attempts to mimic biological brain processes, believed in the 1940's to be discrete neural switching, inspired John Von Neumann to derive the foundations for what has become the prolific digital, serially operated computer of today.

The architecture and operation of these systems enables sequential processing and addressed data storage. As a consequence programming techniques have evolved which exploit this inflexible hardware configuration to the detriment of other methods including parallel processing. Many modern data interpretation problems involve stochastic processes, and heuristic methods are often used to solve them. This characteristic of modern computers has resulted in methods relying on explicit programmed rules, often termed Rule Based or Expert systems, to interpret these functions. These methods have generated great interest in many areas of research, however, high speed real time applications have proven problematic due to the speed limitations of the architecture as well as the method used to transmit the data.

Condition Monitoring for machine diagnostics and control is a well established discipline and has been the focus of much research. The developments in Artificial Intelligence and computer based control have proved a fundamental technique in this area, however, problems have been encountered when attempting to process and interpret highly correlated or erratic data monitored in real time.

Such characteristics are often found within vibration and airborne acoustic signals obtained from a targeted process. In particular, acoustic data from welding processes are notoriously transient. Attention has been given, within research programmes, to analyse the complex signals associated with the acoustics of welding. The reason for this focus has been the established observation that skilled manual welders are capable of making intrinsic decisions concerning electrode position based partly on audible process noise. This ability is an example of real time biological control, and is a result of repetitive experience and the intelligent characteristic of analysing 'cause and effect'. The fundamental understanding of how so called intelligent beings are capable of performing such feats is still the subject of great discussion. But most psychologists and neuroscientists are agreed upon the fact that it has something to do with the way the brain is "wired".

Utilising computer modelling and statistical analysis, correlations have been found between controllable weld parameters and acoustic emissions. The processing involved, however, in generating control vectors from vast amounts of highly erratic data, has proved non-viable in certain real time control applications. This is not only due to the aforementioned architectural problems, but also because of the often necessary data pre processing required to highlight salient features with which to perform further assessment.

If the biological attributes are to be artificially replicated, new methods in signal processing, pattern recognition and intelligent control have to be researched. The advent of Artificial Neural Networks (ANNs) offers such an alternative.

The techniques used in neural networks have their foundations set on the multiple, parallel processing architecture of the neuronal and dendritic connections found in biological brains. Their operation is based upon cognition and recognition through example rather than initial, discrete sets of programmed rules.

This research investigates the viability of using Artificial Neural Networks for replicating these biologically intelligent abilities for monitoring the acoustics during the Submerged Arc Welding (SAW) process. In such a process the arc is hidden beneath a layer of granular flux and hence prevents the use of visual information from the arc, which is highly descriptive of the process state. Audible information is readily available from the process and offers a possible alternative to the complex and expensive methods of thermal imaging and X-Ray monitoring.

The relatively new technology of neural network application presents limitations to the viable use of dedicated Very Large Scale Integrated (VLSI) and parallel computer devices mainly due to the high financial cost of developing such systems. This research is hence aimed at providing an emulated neural system within a more commercially acceptable PC environment. Proposals for the future development of systems incorporating dedicated hardware and processors are discussed. The principles are not necessarily based on parallel VLSI technology but feasible alternatives such as high speed Digital Signal Processing (DSP) platforms within a PC host.

2.0 INFORMATION SURVEY

2.0 INFORMATION SURVEY

In the field of metal fabrication, Fusion Welding is one of the most important techniques in manufacturing industry. Although various metallic tools and structural configurations have been employed for thousands of years, today's prevalent methods of fusion welding have been developed and perfected only within the last seventy years.

With this development has come that of automation. Industrial requirements for standardisation and quality control of large scale component manufacture, together with improvements in safety by removing manual operators from a hazardous working environment, have resulted in industry directing much effort towards the remote robotic control of welding processes.

With automation has come the necessary development of real time weld monitoring and hence a proliferation of transducer designs, methods of signal processing and AI programming, employed to interpret the acquired signals.

In all, significant benefits have resulted from all these areas of research, however, much effort is still required in the area of adaptive control if the quest for reliable, fully integrated automation is to be successful. The adopted welding process for a particular task is subjective. The choice of welding control parameters and definitions of desired weld quality is still the subject of much debate and has resulted in a growing trend in the development of expert systems to derive pre-weld parameter settings and predict possible weld defects.

The large permutation of possible control scenarios and weld set-ups has meant that to date, no single fully integrated closed loop weld control system exists for all possible applications of a particular welding process.

2.1 The Fusion Welding Process

The fundamentals of fusion welding are common to all modern processes within this category. Heat is used as an input to melt the base or parent metal to be joined, often together with a filler metal, in an attempt to fuse the molten materials into one body. As they cool, the coalescence of material, in an ideal weldment, allows the final body to behave mechanically as one. (Funk & Rieber, 1985, p. 2)

Fusion welding processes which lend themselves more readily to automation utilise an electric arc between an electrode and the parent metal to provide the heat input. The molten weld pool formed will readily combine with the constituents of the surrounding air, unless shielded by an inert gas provided in the form of a regulated gas envelope at the welding head or as a by-product of a molten flux. If un-shielded, the resulting contamination will damage and weaken the welds mechanical properties, with the formation of porosity through oxidation and cracks during cooling.

The formation of an optimum solidified weld geometry is dependent upon the arrangement of certain pre-set conditions and the maintenance of process parameters during the operation. The pre-set parameters which are unable to be controlled during operation are :-

1. type and thickness of parent metal
2. type and form of filler material
3. shielding material
4. joint configuration and 'fit-up'

The derivation of these parameters are of high importance to the final quality of the weldment, and there is continuing development of expert systems dedicated to the correct selection of materials, consumables and joint configurations for certain structural applications. (Berger *et al.*, 1994, Palotas,1992)

The process parameters which are directly controllable during the performance of the weld are :-

1. Angle of electrode

The angle of the electrode is generally constant for a particular complete weld run (unless complex robotic control is used), but it can change between weld runs, and so it plays an important role in the overall weld bead geometry. Concentrations of heat caused by a non-optimum angle are detrimental to the welds structural integrity. Most automated welding, especially submerged arc welding, utilise a constant down hand electrode angle. (Funk & Rieber, 1985, p. 94)

2. Weld position (seam tracking)

Weld position or seam tracking provides automatic directional alignment of the arc relative to the joint. Although the 'fit-up' of the joint configuration is pre-determined, the accuracy of this parameter can change due to heat distortion of the parent metal during welding. Compensation is therefore required in positioning the welding head, as well as possible changes in material deposition rates. Mis-alignment of the weld pool causes insufficient penetration and consequently a weak weld. Modern, sophisticated methods of seam tracking include laser guidance with video imaging (Larson, 1993, p. 44) and, in the case of submerged arc welding, ultrasonic echoes (Stroud *et al.*, 1991: Stroud & Swallow, 1992, pp. 9-12: Harris, 1992).

3. Weld speed (travel speed), Current and Voltage

Voltage, current and weld or travel speed are united in determining the energy or heat input to the weld per unit length, and can be expressed as :-

$$E\ (kJ/mm) = \frac{\text{voltage x current}}{1000 \times \text{weld speed (mm/s)}} \qquad \text{Eqn. 2.1.1}$$

From the equation 2.1.1, it can be seen that the thermal efficiency increases with an increase in travel speed. However, weld speed is inversely proportional to weld bead width, and excessive speeds result in undercuts and irregular welds. Low speeds increase thermal input and can cause burnthrough. (Houldcroft, 1989, p. 100)

Current and voltage can be utilised in either a.c or d.c format. Direct current provides a generally more stable arc and steady metal transfer due to the constant polarity, however, it can suffer from 'arc blow' caused by the magnetic field or 'Pinch Effect' surrounding the arc (Houldcroft, 1977, p. 29). When welding magnetic materials, a magnetic flux is produced ahead and around the arc. If the lines of magnetic force become tightly packed, for instance at the ends of a weld joint, the cumulative flux can force the arc from the centre of the weld pool causing a defective weld. Compensation includes the careful positioning of the earthing terminals, reduction of arc length or ultimately reverting to an a.c power source. (Funk & Rieber, 1985, p. 67)

Current varies with arc length, and in automatic and semi-automatic processes is controlled via the wire filler feed speed. Increasing the current also increases the material deposition rate which, in turn, provides a relationship between the consumable filler material or wire diameter and wire feed speed. Increased current density, due to a smaller wire diameter, determines the depth of penetration. On line penetration measurement is difficult, especially in the case of submerged arc welding. On line

measurements have been studied with the application of ultrasonics with validation by post weld inspection. (Stroud, 1982)

Non-optimum settings of current, weld speed and arc length creates 'spatter'. This occurs when the energy provided to form the arc is enough to force the molten filler material from the weld pool resulting in damage to the base metal. (Funk & Rieber, 1985, p. 47)

The voltage determines the weld bead profile. Although the change in profile can seemingly affect the penetration, it has less influence than current density and it is mainly the dilution or coalescence of the weldment that is affected. Raising the voltage provides a wider, flatter bead with reduced side wall fusion, whilst a low voltage results in a more unstable, raised and rounded bead which is more susceptible to forming stress raisers at the edge of the weldment where the bead meets the parent metal (Houldcroft, 1989, pp. 28-29).

Higher voltages provide the capability of longer arc lengths. Increasing the arc length results in a reduced current flow and the overall effect is the provision of less thermal energy to the weld.

4. Wire feed speed

The wire feed speed or electrode feed rate is generally optimised depending upon electrode size and material. When constant voltage power sources are used the feed rate is used to control the welding current.

5. Electrode 'stick-out'

Stick-out or electrode extension is the distance from the contact tube (the point of electrical contact with the electrode) and the tip of the electrode (or more accurately the root of the arc). Increased stick-out reduces the current level necessary to melt the electrode for a given feed rate due to increased resistance (resistance heating). Excessive stick-out will cause increased deposition at low heat resulting in a poor bead shape.

6. Stand off

The stand off is the distance from the weld nozzle and the work piece and can therefore vary if a long electrode stick-out is required. Increased stand off can reduce the effectiveness of the shielding material and in the case of gas shielding will require increased gas flow and specialised nozzles to reduce turbulence. (Crichton *et al.*, 1978, pp. 137-143)

A typical weld set up is illustrated in figure 2.1 and figure 2.2 showing typical changes in weld bead profile due to changes in weld parameters.

The optimization of these parameters provides a stable arc and a constant, steady transfer of material to the weld pool. Many researchers have dedicated work towards categorizing the transfer mechanism phenomena observed in the arc (Norrish & Richardson, 1988 : Becken, 1969 : Lancaster, 1979, pp. 204-267).

Table 2.1 illustrates the classification of material transfer mechanisms as compiled by the International Institute of Welding (IIW). Although specific characteristics have been identified, there are four basic categories. (see figures 2.3a - d)

1. Spray transfer
2. Globular transfer
3. Short circuit or Dip transfer
4. Slag protected or Flux wall guided

(Lancaster, 1984, p. 210)

Spray transfer is inherently more stable with small droplets being transferred regularly at relatively high frequency. Globular transfer involves the formation of larger droplets whose momentum within the arc plasma can cause excess spatter and a lower frequency of transfer creating an unstable process. Short circuiting or Dip occurs when the wire feed speed is at a rate that exceeds the time necessary to melt it. When the wire touches the molten weld pool, the increase in current increases the heat input until the wire melts forming a bridge between the solid wire and the weld pool. Slag protected or Flux wall guided is found in submerged arc welding, where the arc is covered by molten flux. The droplets swing with a pendulum motion from the end of the wire and may come in contact with the wall of the surrounding solidifying flux. The droplet then detaches from the wire and runs down the flux wall to meet the weld pool. (Lancaster, 1984, p. 209)

The theories behind transfer phenomena are complex and include magnetic or Pinch effects, polarity and wire composition as well as controllable process parameters. In general the theoretical voltage/current relationship with associated transfer mechanisms is given in figure 2.4 (Chawla, 1993, p. 103).

The droplet deposition rate is dependent upon current density. Larger diameter wires give a lower transfer frequency for the same current (Lancaster, 1984, pp. 227-233). As increased current density also increases penetration it may be deduced that the observance of transfer rate is an indirect indication of penetration.

The process arc undergoes a highly transient initiation, the control of which is still in debate as weld quality is often depreciated by run-in and run-out conditions. However,

the steady state arc properties have to be maintained after initiation and through to the completion of the weld run, and this maintenance is of prime importance to weld quality.

The main forms of industrial arc welding retain the common physics of operation and differ in only two main ways :-

1. the manner of weld shielding
2. the method of filler material addition.

2.1.1 Weld Quality Assessment

The quality of any particular weld is subjective. In a recent Brite EuRam project aimed at applying Artificial Neural Networks to Metal Inert Gas welding, the technical specification states :-

> *"Weld quality is a difficult measure to quantify. Welds themselves are neither good or bad, just acceptable or unacceptable for the intended application.....A weld passed as acceptable for the construction of a storage tank may not be acceptable in the construction of a nuclear reactor."*

and lists the general quality criteria for welds as :-

1. Mechanical properties - strength and resistance to fatigue
2. Resistance to corrosion
3. Stiffness
4. Appearance

(Ogunbiyi, 1993, p. 6)

The appearance of the final weld bead is not purely cosmetic. The size and shape of the weld can have detrimental effects on the mechanical behaviour in terms of forming stress raisers as well as yielding a metric to the degree of penetration and side wall fusion.

In general an optimum weld is formed by selecting the correct material and parameter specifications prior to commencement and the uninterrupted control of the appropriate parameters during the process. Discontinuities, which are often encountered, can be responsible for the following main categories of defects :-

1. Porosity

The result of inefficient deoxidation of the weld area during formation, which yields gas pockets or voids within the weld bead.

2. Slag Inclusions

The entrapment of non-metallic solids formed from the complex metallurgical reactions during welding. These solids are often deposited at the base of the bead at the boundary with the parent material.

3. Incomplete Fusion

Caused mainly by the incorrect setting of process parameters, whereby the parent material is not heated to a high enough temperature to allow complete coalescence. Further causes include oxidation and unclean or badly prepared joints.

4. Undercut

Undercuts are formed at the boundary between the weld bead and the parent material. Sharp recesses can occur either at the side wall (during a multi-pass weld) or at the surface. Undercuts have a detrimental affect on the structural integrity of the joint, especially if it is subject to fatigue. General causes include high current, long arc lengths and incorrect selection of electrodes.

5. Hot Cracking

As the weld metal cools inherent constituents in either the parent or filler material, which have a lower melting temperature, accumulate at the grain boundaries during solidification. Further cooling causes added stress at the boundaries due to shrinkage which in turn pulls the grains apart and intergranular cracks propagate. Main causes are material impurity and incorrect weld parameter specification.

6. Cold Cracking

Cold cracking is the result of inadequate ductility of the parent or filler material. Induced stresses after cooling will promote cracks, especially in large grained materials. (Fallick *et al*, 1978, pp. 74-76)

Various levels in the severity of these conditions maybe permissible in certain welded applications, and due to the very nature of the process they are almost impossible to completely eliminate. However, the pressing need for high quality and cost effectiveness necessitates the minimisation of these defects.

2.1.2 Weld Control Objectives

From a monitoring and control point of view, the understanding of the cause of certain defects is essential. Once the cause has been established an algorithm has to be created to generate the appropriate control vectors to compensate for any monitored deviation.

Comprehensive study of the Metal Inert Gas process, carried out by the WINNER consortium (Stroud *et al*, 1996), yielded the parameters needed to be controlled to enable a consistent defect free weld for butt and fillet weld configurations. As well as the use of visual information the emphasis is on the monitoring of voltage, current, travel speed, wire feed speed, stand-off and acoustic sound pressure levels from the process. The corrective strategy is obtained through the direct control of all except the consequential parameters of sound and vision. Table. 2.2 shows an abridged version of the approach.

The inter-dependency of the monitored parameters can create problems in establishing a corrective control strategy. As a consequence a priority schedule has to be constructed with the most catastrophic defects being considered first.

Of special interest is Arc Stability. An unstable arc can be responsible for such defects as inconsistent fusion, slag inclusions, blowholes and porosity. Fallick *et al* (1978, pp. 72-73), attribute poor arc stability to :-

1. Open circuit voltage of the power source
2. Transient voltage recovery rates of the power source
3. Droplet size
4. Ionisation of the arc
5. Bad manipulation of the electrode

and concludes,

> " *Thus, the electrode* (type and size etc.), *the power source, and the welder, all contribute to arc stability."*

Existing methods of assessing arc stability for the MIG welding process have been reviewed by Ogunbiyi & Norrish (1996). The summary describes the wide and established use of descriptive statistical methods utilising the standard deviation of electrical process parameters to derive stability but, however, suggests a more adaptive method by means of ratios or indices.

The ratios are based upon analysing the transfer mode during the process, where spray

transfer is considered naturally stable, globular transfer unstable and dip transfer either stable or unstable depending on consistency. The indices are calculated as follows :-

$$\textbf{Transfer Index (TI)} = 1 - (I_{min} / I_{mean}) \qquad \text{Eqn. 2.1.2.1}$$

$$\textbf{Transfer Stability Index (TSI)} = I_{max} / I_{mean} \qquad \text{Eqn. 2.1.2.2}$$

$$\textbf{Dip Consistency Index (DCI)} = 1 - (V_{bk} / V_{mean}) \qquad \text{Eqn. 2.1.2.3}$$

where,

I_{min} = absolute minimum current value in window
I_{mean} = average of transient current of applied window size
I_{max} = absolute maximum current value in window
V_{mean} = average of transient voltage of applied window size
V_{bk} = average of transient voltage $<= V_{mean}$

Although the results were validated by visual assessment, it was found that anomalies occurred when equations 2.1.2.1 - 3 were tested individually on different MIG power sources. These equations were then combined to form a Power Ratio which could be used for "*a quick identification of the metal transfer mode and arc stability.*", being defined by :-

$$\textbf{Power Ratio (PR)} = I_{bk} \cdot V_{bk} / I_{mean} \cdot V_{mean} \qquad \text{Eqn. 2.1.2.4}$$

where,

I_{bk} = average of transient current $<= I_{mean}$

The calculated ratios are categorised as illustrated in figure 2.5. Further examination of the Power Ratio Index also provided an assessment for degrees of spatter encountered during the process. Spatter is generally associated with globular transfer and is the main visible indicator of instability (Ogunbiyi, 1995, p. 3).

Another control objective, which maybe further responsible for spatter, is arc length. Optimum arc length control is of great importance in arc initiation or strike to ensure that the electrode does not fuse to the weld piece (in the case of too short an arc length). Excessive arc length causes a lack in intensity and chaotic direction which tends to scatter molten material as it travels. These disturbances can in turn disrupt the shielding medium resulting in contamination and porosity. (Fallick, 1978, p. 68). In general, arc length is controlled by the correct settings for voltage and current and in manual welding Fallick (1978, p. 68) states that it is,

> "*.....largely a matter of welder skill, involving the welder's knowledge, experience, visual perception, and manual dexterity.*"

2.2 Submerged Arc Welding

Submerged Arc Welding (SAW) is a specialised form of a fusion arc welding process developed independently in the USA and former USSR during the 1930's.

The consumable filler wire provides the electrode and is fed via a spool, at a controllable rate, to the welding head. The constituents of the filler wire can be alloyed to provide a desired metallurgical weld formation.

The shielding material is in the form of a gravity fed granular flux, which completely immerses the electrode tip and consequently the arc. The burning of the flux constituents during welding provides an inert gas bubble around the weld pool preventing oxidation. Flux / wire combinations can be used to determine the composition of the weld metal. However, due to economic reasons of manufacturing alloyed wires, it is generally preferred that plain wires be used with alloyed additions made via the flux (Houldcroft, 1989, p. 61).

The gravity fed flux means that only down hand welding is possible together with a pre-set electrode angle, with a vertical, trailing or leading electrode affecting penetration, re-inforcement and the tendency to form undercuts (Houldcroft, 1989, p. 30). A typical submerged arc welding set up is illustrated in figure 2.6.

The immersion in flux provides thermal insulation with an efficiency as high as 60% (Houldcroft, 1989, p. 19). The flux layer also entertains high current densities and consequently deep penetration is possible. It is therefore used in heavy engineering and structural applications such as shipbuilding, civil engineering and pressure vessel construction utilising steel plates from 6mm thickness to greater than 50mm with the application of multi-pass and multiple wire techniques (Houldcroft, 1977, p. 49).

Shorter arc lengths maybe maintained beneath the flux enabling lower weld voltages with the advantage of increased dilution and side wall fusion.

The overall effect is that of a smooth profiled high strength structural joint which is economical to produce in terms of both consumables and energy input.

The advantages of this method also include increased power with reduced spatter and the covered arc negates the necessity for precautions against arc flash. However, the concealed arc and weld pool means that the operator cannot visually gauge the weld head to assess the stability of the weld formant.

Studies of the arc condition and transfer mechanisms associated with submerged arc welding have been limited due to the concealment of the arc during the welding process (Houldcroft, 1977, p. 51). Infra-red thermal imaging and X-Ray imaging has given some indication of the arc, weld pool and material transfer properties. Lancaster (1984, p. 209) highlights the findings of Van Adrichem (1966), whose use of high speed X-Ray film found the various transfer modes shown in figures 2.7a - h.

Relative deposition rates have been studied and documented with corresponding current densities as well as weld pool formation and dynamics (Lancaster, 1984, pp. 227-235). Deposition rates shown against current density is illustrated in figure 2.8.

Work carried out by Kralj & Tušek (1988) generated some findings concerning material transfer rates and dynamics with multiple electrodes. This study concentrated on the relatively stable spray or streaming transfer mode and examined the droplet transfer rates for various numbers of electrodes, differing current densities and changes in polarity. The results confirmed findings reviewed by Lancaster (1984, pp. 227-228), as well as concluding that there was an increase in droplet transfer rate for negative electrode processes.

The operating variables for Submerged Arc Welding are, in principle, identical to any electric arc process. The prime difference is the shielding medium, where the depth of granular flux, grain size and its constituents affect weld quality. If the flux is too deep the gases generated cannot readily escape and together with an arc that is too confined will cause the surface of the molten metal to distort resulting in a poor finish. With a flux depth too shallow open arcing (due to incomplete coverage) is more likely. This generates excessive spatter, arc flash, porosity and surface craters due to inadequate de-oxidation. (Wamack, *et al*, 1978, pp. 205-206)

Providing parameter, material and consumable selection are correct, the excellent shielding properties of the molten flux produce welds which are generally free of defects associated with oxidation. Most defects are caused by the contamination of the joint, electrode or flux, however, excessive travel speeds can cause ineffective flux coverage with the aforementioned consequences.

Submerged arc welds are, however, susceptible to mechanical failures. The slow cooling rates of the relatively large weld pools beneath the insulating properties of the flux result in a coarse grain structure, segmentation and cracking. Furthermore, the weld bead profiles of deep penetration welds tend to be narrower at the surface than at the middle. Micro-shrinkage voids can occur forming cracks at the centre due to the uneven cooling rates. (Wamack, et al, 1978, p. 222).

2.3 Robotic and Automated Welding

The American Welding Society (AWS) has suggested that the average time a manual welder spends welding is 30% of the overall project time. The remaining 70% is spent in preparation, dressing the weld and taking food and rest breaks. The development of automated robotic welding has increased this 'arc time' to between 80% - 90%. For industry this represents a substantial increase in productivity and consequently, profit. (Funk & Rieber, 1985, p. 372)

The desire to develop automation is not totally profit driven. The need to conform to health and safety aspects by removing manual welders from obnoxious and hazardous conditions has proved an important incentive.

Robotic welding entails the combined control scenarios of kinetic and process control, kinetic control for positional alignment and travel speed, and process control of the electrical power variables. The repetitive positional accuracy and overall quality of the final weld, relies heavily on the use of appropriate transducer technology, signal processing and adaptive control techniques.

State of the art welding robotics utilise sophisticated techniques of monitoring and adaptive control. Nishar *et al* (1994, pp., 4-11), describe the use of Infra-Red Thermography with charged coupled device (CCD) imaging to measure centre line bead temperature and a self-tuning proportional-integral (PI) controller to regulate the heat input to the gas metal arc welding process.

Visual techniques for seam tracking and positional alignment with CCD video have attracted much attention (Kaneko *et al*, 1993, pp. 677-680, Boilot *et al*, 1994, Paton, 1994). Combined monitoring of arc current, voltage and arc sound has been developed by Chawla and Norrish (Chawla & Norrish, 1992) by applying statistical analysis techniques to a data acquisition system.

Computer based control has been exploited for the adaptability of software, and compatibility with digital process and kinetic actuators. This exploitation is further enhanced by the utilisation of computer aided design facilities to form computer integrated manufacturing (CIM) plants. In this way large scale production can be undertaken with reliable speed and high standards of quality assurance.

Submerged arc welding is a heavy industrial semi-automatic process used within such areas as civil engineering and ship building. A recent survey of weld automation in worldwide ship building by Quintino and Pecas (1993, pp 10-21) highlighted the

difficulties in applying robotics to large scale construction in low production numbers. The problems are further compounded by complicated joint configurations and often unclean working conditions. The work concludes that the world wide trend is towards increased automation, however, the need in this area of industrial production is for more compact, portable and robust systems with improved on-line monitoring and control capabilities.

In response to these findings an EC Brite EuRam sponsored project, coordinated by Brunel University, U.K, ('WINNER' consortium, 1993), has endeavoured to apply state of the art technology to this problem. By developing a hybrid approach utilising a combination of expert systems, fuzzy logic and, most innovatively, artificial neural networks, a consolidated adaptive system for the control of the Metal Inert Gas process has been created.

Data from video, acoustic and monitored electrical traces from the adopted power source are captured and preprocessed. The neural network system classifies the data into degrees of quality criteria. The assessment is passed to a fuzzy logic system which carries out a comparison to the desired welding strategy formed by a pre-weld expert system, and provides the appropriate correcting vector.

The reported results are proving encouraging, and although further development is required in the areas of response time optimisation and power source adaptability, it has provided the first real time closed loop control system of its kind for welding processes.

2.4 Condition and Process Monitoring

The three main maintenance strategies adopted within industry, as described by Javed *et al* (1992, pp. 357-362), are :

1. corrective
2. preventative
3. predictive or condition based

The general philosophy adopted in this area is the financially more advantageous "prevention is better than cure", making predictive or condition based maintenance the more attractive option.

Although principles of condition monitoring have been established over the last thirty years, rapid strides have been made in the last ten years. This is mainly due to the now acknowledged potential cost savings they offer in plant maintenance together with an acceleration in available supporting technology.

In a literature review on the use of artificial intelligence in condition monitoring, MacIntyre (1993) goes some way to defining this discipline as involving:-

> *"....the capture of data which describes various parameters of machinery, and the analysis of this data to determine the condition of a given machine at a given point in time."*

Constant or periodic monitoring of machine plant involves the application of state of the art transducers to acquire data via vibrational and airborne acoustics, temperature, electrical power supplies and even off-line chemical analysis (in the case of lubricating oil samples for instance) in an attempt to identify abnormalities in operating conditions.

The effectiveness of predictive maintenance hinges on the correct application of techniques to interpret the often transient signals associated with monitored parameters. Javed *et al* (1992, pp. 357-362) provided a survey of the main signal processing methods in diagnostic management of machinery and highlighted the three main areas which offer most success :

1. Expert and Rule based systems
2. Artificial Neural Networks
3. Fuzzy Logic

(The operation, advantages and disadvantages of these various techniques are described in a future section.)

The general aim of these methods is to assess and classify data, which is often highly correlated, to perform a predictive decision concerning the state of the process.

Condition monitoring often involves the isolation of changes in signal patterns over relatively long periods of time, with the changes associated with specific component faults to derive a diagnosis. The accuracy of such systems depends on the sensitivity to small changes in the often noisy and transient data set.

In concluding the review of artificial intelligence MacIntyre (1993) states :-

> *".....expert systems in general have difficulty in dealing with noisy, incomplete and contradictory data, which is exactly the type of data found in condition monitoring applications."*

and furthermore,

> *"The pattern recognition and learning generalisation qualities of neural networks would appear to lend themselves to this kind of application....."*

2.5 The Nature of Acoustics

The study of the phenomena associated with sound and the derivation of acoustic quantities is credited to physicists such as Stokes, Rayleigh, Doppler and Helmholtz who were prominent scientists during the 19th century.

Sound is generated by the compression and rarefaction of the medium surrounding an oscillating or vibrating object. The speed at which sound propagates through a medium is derived from :

$$c = \sqrt{\frac{\gamma p}{\rho}} \qquad \text{Eqn. 2.5.1}$$

where, γ = ratio of specific heat at constant pressure to that of constant volume

p = steady state pressure of medium (N/m^2)

ρ = density of medium (kg/m^3)

For air at 20 degrees centigrade and at sea level the velocity of sound is calculated at 344 m/s.

Sound propagates as a wave. From a point source with no obstacles, it can be considered as a spherical dissipation known as "*free-field*" radiation. In reality, however, waves meet obstacles or encounter interference from changes in surrounding air pressure, at which point absorption, reflection and diffraction can occur.

Absorption is responsible for dissipating acoustic energy. Porous or fibrous materials possess higher absorption coefficients, but the degree of absorption is also dependent upon the frequency of the acoustic wave.

Sound energy, which is neither absorbed nor transmitted through an impeding obstacle, is reflected off the surface. Although some energy is invariably dissipated, the reflected sound propagates radially from the point or surface of incidence until damped by further obstacles or surrounding air pressure influences.

Sound waves encountering an obstacle can bend around it. Parted waves bending around an object may re-unite in various combinations to produce changes in direction and diffraction patterns. Waves passing through an opening will spread radially from that opening, this diffraction is defined by Huygen's theory of wave propagation.

Reflected sound interacting with the incident sound causes abnormal sound pressure distributions. This interference causes changes in the original direction of propagation and consequently tends to break up the existing waves. Depending upon the phase of the interacting waves, the resultant can be magnified or dampened. These phenomena are respectively known as 'constructive' and 'destructive' interference.

The sound pressure level (SPL) is measured as a force per unit area for the r.m.s amplitude of the wave. The large numerical range of measurable pressures warrants a logarithmic scale in decibels (dB), to form a ratio against a constant reference pressure. The reference is calculated as the pressure recorded at the threshold of audibility for a sound frequency of 1000 Hz, and is considered to be 2×10^{-5} N/m^2 . SPL is therefore expressed as,

$$\text{SPL (dB)} = 20. \text{Log}_{10} \left(\frac{P_{r.m.s}}{2 \times 10^{-5}} \right) \qquad \text{Eqn. 2.5.2}$$

The sound pressure is the product of air particle displacement and velocity where,

$$\text{Particle velocity, u (m/s)} = \frac{P}{\rho c} \qquad \text{Eqn. 2.5.3}$$

and ρc is known as the characteristic impedance of the medium with units of acoustic ohms and,

$$\text{Particle displacement, d (m)} = \frac{u}{2 \pi f} \qquad \text{Eqn. 2.5.4}$$

The sound intensity is calculated as the acoustical power per unit area (W/m^2) and is expressed as,

$$I = \frac{P^2}{\rho c} \qquad \text{Eqn. 2.5.5}$$

The large scale range of measurable power justifies the use of a logarithmic scale as a ratio with a reference power at the threshold of audibility which is calculated to be 10^{-12} W/m^2 . Therefore,

$$\text{Sound intensity level (dB)} = 10. \log_{10} \left(\frac{I_{r.m.s}}{10^{-12}} \right) \qquad \text{Eqn. 2.5.6}$$

Because sound intensity and pressure are referenced at the same level of audibility, for any given sound wave the same numerical result in dB is valid for both parameters.

Table 2.3 indicates relative sound pressures and intensity levels.

The effect of distance attenuates the sound power in accordance with the inverse square law. Therefore, as,

$$\text{Power (I)} \quad \alpha \quad \frac{1}{d^2} \qquad \text{Eqn. 2.5.6}$$

and from equation 2.5.5,

$$\text{Pressure (P)} \quad \alpha \quad \sqrt{I} \qquad \text{Eqn. 2.5.7}$$

then :-

$$\text{Pressure (P)} \quad \alpha \quad \frac{1}{d} \qquad \text{Eqn. 2.5.8}$$

that is, sound pressure follows the inverse distance law. (Merken, 1989, pp. 141-168)

The interaction of emanating sound waves caused by various mechanisms provides complex waveforms whose phase, amplitude, intensity and frequency can appear sporadic or erratic.

The biological reception of sound, however complex, is generally understood in terms of mechanical transmission. The perception and interpretation of sound is far less clear but is discussed in a later section.

2.5.1 Acoustics and Welding

Acoustics are already known to be characteristic to many processes. In condition monitoring of rotating machinery, frequency components within acoustic emissions, are used to estimate bearing behaviour (Herraty, 1993, pp. 51-53). In cutting processes to estimate tool wear (Martin, 1994, pp. 527-551) and in the study of internal combustion engines to diagnose cylinder wear and injector problems (Vu *et al*, 1991, pp. 31-35).

An established observation is that a skilled manual welder is able to monitor an arc welding process with audible process noise. Subconscious decisions concerning the position of the electrode, effectively controlling the arc length and angle, are made in response to adverse changes in the audible sound.

Informal discussions with professional manual welders at Cranfield University, U.K, were aimed at identifying the features 'looked for' when instigating a weld. The general consensus was that the 'smoothness' and 'regularity' of the sound was a major key in establishing the stability of the weld. Further studies by Ogunbiyi (1996, p. 13) also highlighted a time dependency or temporal progression of the sound being indicative of changes in the process parameters. Relatively long intervals between the regular pulsating sound of the dipping indicated ignition problems. A 'harsh' sound was associated with excessive wire feed speed whilst a low frequency buzz was attributed to excessive voltage settings. In conclusion Ogunbiyi states :-

> *"The study of welding transient waveforms indicates that the welders are in effect looking out for low frequency or irregular sounds; the arc is expected to give a crisp regular noise..."*

The question is; How can these irregularities be measured and mathematically represented at an appropriate speed to yield them usable within a machine environment?.

Studies of acoustic emissions to characterise weld conditions, in various welding processes, have mainly concentrated on statistical correlations with known process parameters. The erratic nature of acoustic signals makes processing and interpretation a numerically intensive task. In the work to assess the performance of flux cored wires in the MAG welding process, Chawla (1993, p. 15), comments,

> *"....it has been recognised that it is a major task to evaluate this information* (airborne acoustic signals) *and to use it for process development and quality control"*

This difficulty was highlighted in the preliminary work where sound assessment was abandoned due to over sensitivity to noise. Off-line statistical analysis of acoustic emissions was then adopted together with spectral Fast Fourier Transform (FFT) analysis. By correlating and validating the analysis with acquired current and voltage signals, it was anticipated that numerical models of the welding process could be constructed. Investigations proved that by numerically integrating the sound data and plotting a spectrum of the result, close similarities occurred with the voltage and current data with a 1 ms delay. Further manipulation with moving averages, highlighted the pulse or dip modes of the process, however, limited success was found in assessing the spray current mode. The work concludes with a suggested direct correlation between arc current and integrated acoustic signals which therefore justifies its possible use for quality assessment of the welding process. However, a faster and more robust method needs to be found to utilise this parameter for on-line analysis and control.

Investigations into acoustic tuning of the MAG welding process have also been carried out by Blumschein (1994). A microphone was used to capture time/amplitude signals in an attempt to identify arc re-ignition when short circuiting. The short duration due to the explosive nature of this phase is difficult to monitor via the voltage trace. Blumschein claims that acoustics provide a longer duration of signal to analyse as well as providing electrode stick-out information. In conclusion, recommendations are made to utilise fuzzy logic principles to recognise the transient signals acquired.

In GTA welding Kaskinen *et al* (1986, pp. 763-765), have provided relationships in acoustic emissions and voltage / current parameters in pulse mode. In this mode, the pulsed power input to the weld creates a corresponding volume change to the arc plasma which, in turn, generates an alternating sound pressure level which is audible. The sound intensity proved to be proportional to the power input to the process, from which voltage and current values were derived. The voltage level provides information concerning the arc length and the research concludes with a definition for feasible arc length controller utilising acoustic information.

Schlebeck (1982), investigated acoustic emissions associated with MIAB (Magnetically Impelled Arc Butt) welding, utilising a microphone to monitor pipe welding. The system highlighted differences in time/amplitude acoustic signals corresponding to changes due to poor edge preparation and contamination. Further success showed the statistically validated identification of stable and unstable welds from the same signals.

Erdmann-Jesnitzer *et al* (1966), attempted to attribute audible noise from the arc welding process to various causes,

1. power source noise
2. gas discharge through particular nozzles
3. vibration through the system or acoustic resonance
4. variability of the anode and cathode spots
5. material transfer and mechanisms.

The work further found correlations between SPL and arc length but failed to find relationships with frequency spectra.

Fast Fourier Transforms were applied, with limited success, to CO_2 laser welding by Mao *et al* (1993, pp. 17-22). Acoustic data was acquired via a microphone suitably amplified and processed in real time with a PC hosted digital signal processing (DSP) card to generate an FFT between 4 Hz and 100 kHz. Further processing reduced the frequency range to isolate salient bandwidths. The acoustic emission energy was derived from integrating the FFT spectrum over all frequencies and reproducible correlations found with laser power, welding speed and lens focal position.

The evidence for research into the use of acoustic emission analysis in submerged arc welding is limited. This can be attributed to two main reasons, firstly the lack of visual information on which to validate the results. For example, expensive and often complex apparatus such as X-Ray and infra-red thermography imaging is required to examine weld pool dynamics and material transfer. Secondly, sound pressures are suppressed by the shielding flux which, at the same time, is responsible for creating sporadic transients as gases escape from the layer of molten slag.

Work undertaken by Pilous *et al* (1987, pp. 468-481), attempts to use acoustic emissions in submerged arc welding to isolate propagating weld defects such as cracking within the heat affected zone (HAZ) as they occur. Piezoelectric transducers were subjected to sound within the audible range via a suitable waveguide. It was found that the formation of cracks in the slag and scale obscured most data relating to the formation of defects with only very large or cold cracks possibly being discernable.

The effort evident in the area of acoustic emission analysis for welding processes, focuses on the correlations with the fundamental process parameters. The use of acoustics as a valid aid to weld monitoring is debatable in some instances as the methods used to interpret them have revealed already known metrics. However, the general conclusions call for more robust methods of signal processing and interpretation to yield more consistent and reliable measures of the process variables.

2.6 Concepts of Human Hearing

Human hearing is by no means the most acute to be found in the animal kingdom. With normal hearing the frequency range is between 20 Hz and 20 kHz, which is generally accepted as the bandwidth which constitutes audible sound. The intensity range is from 0 dB to 140 dB which corresponds to Sound Pressure Levels (SPL) of 2×10^{-5} N/m^2 - 200 N/m^2 and sound power levels of 10^{-12} W/m^2 - 1000 W/m^2 .

The threshold of feeling is around 130 dB and it is clinically accepted that no exposure over 115 dB SPL should be encountered if permanent damage to the hearing mechanism is to be avoided.

The ear is more sensitive to certain frequencies. Figure 2.9 shows a typical audiogram illustrating human hearing sensitivity. It can be seen from the bottom boundary that hearing is most susceptible to frequencies in the 1 kHz - 2 kHz range, with lower frequencies requiring more intensity to be heard. Sharp decreases in sensitivity are observed at 30 Hz and 12 kHz. The upper boundary illustrates the pain threshold levels with associated sound frequencies and intensities (Massaro, 1989, p. 185).

The hearing mechanisms utilise complex biological mechanics to direct, amplify and register the sound in the inner ear. The shape of the outer ear assists in directing the sound through the auditory canal to the ear drum or tympanic membrane, which vibrates in sympathy with the SPL and intensity of the sound wave. The middle ear consists of three bones, the ossicles, which connect the tympanic membrane to the oval window of the inner ear. The action of the ossicles mechanically amplify the movement of the ear drum, which can be as little as 10^{-8} mm. By means of the principle of levers and the difference in surface area between the ear drum and oval window of the inner ear, the gain is around 1000. (Fig. 2.10)

The inner ear consists of a liquid filled hollow spiral bone known as the cochlea, which reduces in diameter along its length. The basilar membrane partitions the cochlea and is anchored to the tectorial membrane by microscopic hair cells known as cilia. (Fig. 2.11)

Due to the incompressibility of the fluid, the amplified vibrations are transmitted through it from the oval window and propagated through the cochlea. The basilar membrane oscillates in sympathy with the fluid and, because its width and flexibility varies along its length (which is approximately 35mm), different regions will vibrate according to the frequencies present in the propagating wave. High frequencies excite that part nearest the oval window, middle frequencies toward the centre of the cochlea and low frequencies affect the tip as shown schematically in figure 2.12. The cilia register the area

of excitation and by means of synaptic connection transmit the information through the neurons of the auditory nerve. (Carlson, 1984, pp. 198-204)

The auditory system is able to simultaneously discriminate between different sounds, for example registering conversational speech within background music. Sound received at each ear is divided into twenty four discrete channels known as *critical bands*. Therefore the human auditory mechanism can be assimilated to a forty eight channel audio system (Rolan-Mieszkowski, 1993, pp. 27-28).

It is this biological setup which goes some way in explaining the sound pattern differentiation which enables complex auditory scene analysis.

2.6.1 Auditory Scene Analysis

Auditory scene analysis is a science in its own right. Its study is focused upon how biological systems process environmental sound in order to develop a mental representation of the world that surrounds it (Welch, 1996).

In order to create this mental image it is necessary to characterise audible sound to provide a scientific description of what we hear. Audible acoustic signals are generally described by the following rather subjective criteria :-

Loudness

The unit of loudness is the *phon* and it is related to sound pressure levels at 1kHz. There exists some controversy over the perceived loudness of sound. The general opinion is that for an increase or decrease of 10dB SPL the perceived loudness is double or half respectively. A subjective measure of loudness, the *sone,* equates to 40 phons which is equivalent to 40dB SPL, the level of a quiet room. Two sones is double the perceived loudness and 0.5 sones half.

Loudness is also dependent upon the frequency and the bandwidth of the signal. Figure 2.13 illustrates the relationship between frequency, SPL (dB) and loudness (phons). It shows for example that for a pure tone of 1kHz at 10 dB SPL an increase to 80 dB SPL is required for a pure tone at 20 Hz to be perceived as equally loud. (Everest, 1981, pp. 36-42)

The effect of bandwidth is due to the critical bands exhibited by the cochlea. The critical bands vary within the cochlea depending upon the frequency being analysed as illustrated by figure. 2.14. Any complex sound comprising of frequencies greater than the critical band will be perceived as louder than a sound whose component frequencies lie totally within it (Lindsay & Norman, 1977, p.176).

Pitch

The perception of pitch is related, not linearly but logarithmically, to frequency. The subjective unit of pitch is the *mel* and its relationship to frequency is illustrated in figure 2.15. As a point of reference a pitch of 1000 mels is defined as the pitch of a 1kHz tone at 60dB SPL (Everest, 1981, p. 44).

Other adjectives exist for human sound perception including timbre, dissonance, consonance, density, volume and musicality. These areas are more a measure of quality than mathematical interpretation and although of psychological interest, need not be discussed within the context of this thesis.

Of further interest within a sound filled environment is that of *masking*. Masking occurs as a result of other pure or complex tones being present within a particular active critical band of the basilar membrane. Depending on the frequency and intensity of the tones, one will mask the effects of the other therefore rendering it inaudible or distorted (Lindsay & Norman, 1977, pp. 157-160). Figure 2.16 illustrates the frequency and sound levels of a tone in the presence of a noise attempting to mask it. Notice how the intensity and upper or lower frequency of the tone affects its position within the active critical band rendering it undetectable.

Detection of minimum perceptible changes in sound is very subjective, and also depends upon frequency and intensity. For a pure tone of between 50 Hz and 10 kHz and an SPL of 50 dB a change of 1 dB can be detected with normal hearing. As sensitivity drops with SPL, a 40 dB pure tone requires a 3 dB change to be discernable. This is a consequence of the density of ganglion neurons present along the basilar membrane. For perceptible changes in pitch, experimental evidence suggests that on average there are 1150 neurons per millimetre of membrane and a unit change on the mel scale will activate 12 neurons along its length. However, human experiments into the audibility of changes in pitch have shown that for a just noticeable difference (jnd) an average of 0.3% change in frequency is required. When related to neural activation this equates to a spacing of 52 neurons on the basilar membrane or 4.3 mels (Lindsay & Norman, 1977, pp. 164-166).

Investigations into the detection of signals of short durations have been carried out by Dai & Wright (1995, pp. 798-805). The work is aimed at providing knowledge on how the human auditory system detects these signals including the time duration limits of both signal and lapse time between signals. Current theory suggests two methods :-

1. The listener takes a single observation of approximately 200ms. The decision as to the presence of a signal is based upon the energy of that signal over this period.

2. The listener takes multiple 4ms observations and bases the decision on information statistically combined over these periods.

The question posed is; Are these *integration-periods* fixed ?. A combination of sound frequencies (250Hz, 1kHz, 4kHz and spectral noise up to 6kHz) and durations (4ms - 299ms) were tested on human subjects. Dai & Wright concluded that their results,

> *"....demonstrate remarkable flexibility in listening strategy; listeners adjust the temporal-integration interval according to the demand of the specific task."*

The perceived duration time of a sound was found to be dependent upon the spectral bandwidth of the signal as well as the expected duration of the signal.

The temporal progression of signals is known to be of extreme importance to the understanding of environmental sounds. Wang & Shamma (1995, pp. 186-194) have provided mathematical models for the filter and spectral operations of the cochlea and attempted to replicate the mapping of auditory spectrum within the primary cortex.

Further work on the discrimination of auditory temporal patterns has been undertaken by Ross & Houtsma (1994, pp. 19-26). Here, fifty sets of an intermittent series of 'clicks' within a total time period of around 4s and of interclick intervals of *n* x 250ms, where *n* is a random integer, were generated. Fourteen human recipients were subjected to these patterns in an anechoic chamber. The two conducted experiments analysed firstly the discrimination between the semantic structure of the pattern groups and secondly the effect of inducing an isochronous click along with the pattern structures. In 70% of the cases a distinct improvement in discrimination was noted for changes in click/pause order and if a relatively strong rhythmic click was present rather than timing differences (i.e. length of the pause). Ross & Houtsma (1994, p. 25) conclude with the observation that:-

> *"For sequences of 10 events or more and spanning several seconds, subjects appear to do more than compute single correlation statistics. They seem to be able to isolate, concentrate on, and operate on details of elements or groups of events in the sequence."*

and,

> *"....a very powerful holistic percept that allows perceptual discrimination between pulse sequences was seen to be the metric strength or "clock sense"."*

The localisation of sound is generally achieved through the binaural set up of mammalian audition. The detection of a time delay, albeit small, enables accurate sound location. For a complex sound at eye level and in front of a subject, an accuracy of 1-2 degrees is achievable (Everest, 1981, p. 43).

In all, the human auditory system is not linear. These non linearities are often difficult to understand let alone mathematically model. However, research continues in an attempt to establish an understanding with the ultimate aim of full replication.

2.7 Auditory Perception, Understanding and Thought Processes

The human brain is universally accepted as the most complex biological organ to understand. Theories of consciousness, thought processes and human behaviour have plagued philosophers, psychologists, mathematicians and physicists alike for decades and seem likely to continue to do so for many more. To attempt to discuss the full implications and current theories of consciousness is beyond the scope of this thesis.

Present studies in neurophysiology accept the single cell neuron as the basic element or building block of the biological brain. As many as 10^{11} neurons, closely woven and heavily interconnected, constitute the average human brain.

The neuron itself consists of a body or soma attached to which are dendrites which receive information from other connected neurons. The axon transmits information from the soma by means of electro-chemical potential at a speed of around 100 m/s. Axons and dendrites terminate at synapses, through which complex chemical exchanges take place, to facilitate communication. The maximum known firing rate for a neuron is 1000/s which includes a required recovery or reset period. Interconnection between as many as 1000 neurons is common. (Jubak, 1992, pp. 1-20)

Neurons differ in their configuration of dendritic structure and inter-connectivity. Figure 2.17 shows a typical biological neural cell and some of its many categories. Examples of this are found in all areas of biological brains. (Green, *et al*, 1986, pp. 244-247).

Studies have revealed a mapping of the brain, where different parts are attributed to various bodily and mental functions. The auditory system utilises the right and left temporal lobes as well as the primary auditory cortex situated close to the centre of the brain. (Carlson, 1984, p. 201)

Approximately 30,000 nerve fibres connect the cilia of each inner ear to the primary auditory cortex, all of which are capable of communicating simultaneously with the multiple dendritic connections of the cortex and temporal lobes. It is this massively parallel configuration which enables complex sound to be analysed and acted upon. This architecture provides a robust generalising capability when processing sensory patterns. It also has the benefit of 'graceful degradation', the ability to function with virtually unnoticeable drop in performance when neuronal elements fail to operate correctly or die completely.

The perception and understanding of sound is still theoretical with deep seated hypotheses in the realms of psychology. The majority of studies associated with the

auditory system have focused on the understanding of speech and the interpretation of sequential phonemes to construct semantic structures. These functions are associated with the left temporal lobe. The right temporal lobe is responsible for processing pure tones and rhythms associated with music, as well as the directional location of sound (Lindsay & Norman, 1977, pp. 240-248). It is possible that both temporal lobes would be required to de-code the transient sound associated with the welding process, especially if the sound signals acquired contain a degree of phraseology, pure tone and rhythmic synchronisation.

The recognition of music and rhythmic sounds, as well as the syntax and semantics of speech, are associated with Gestalt psychology. This theory was originally proposed by Wetheimer, Koffka and Kohler in 1929. They suggested that the brain organises acquired sensory information into a contextual whole (Massaro, 1989, pp. 38-41). Although originally developed to explain the concepts of visual stimuli, in many cases it is as equally relevant for audible processes. Figures 2.18a-b illustrate a typical visual example of Gestalt. The excerpt of a scene shown in figure 2.18.a is meaningless until seen within the contextual whole in figure 2.18.b. The concept implies that a reasonably representative amount of information needs to be present, as in the notes of a melody, to perceive the meaning. Individual or discrete components acquired in a random order or offered in non-experienced unstructured order, fail to be recognised or register a semantic error (Jackendoff, 1993, pp. 165-171). The philosophical question is what can be considered 'a reasonably representative amount' of information ?. Although the answer is obviously subjective, theoretical rules have been argued in a study by Bregman (1990). This work is based on the auditory equivalents of Gestalt in which certain features within an *auditory stream* are mentally grouped and given internal associations, where an auditory stream is defined as :-

> *"...our perceptual groupings of the parts of the neural spectrogram that go together..."*

The biological mechanism which enables welding sound to be recognised and acted upon is based on *'intelligent prognosis'*, a term used by Durlacher (1994) in comparing the ancient art of flint napping and the modern process of manual welding. A particular action (a blow to the rock face or the angle of the welding torch) is known or predicted to yield a certain result. In both these cases the operator has partaken of experiential learning and is aware of the affect of a particular action and of the tactile, visual and audible information associated with it.

How are these sensory features localised and recognised within the brain ? As already stated earlier in this section the neural transmission being limited to approximately 1kHz

for each neuron is indicative of the lack of relationship between rate of firing and direct sound frequency measurement. Present theory suggests that the characteristics of sound are encoded as a frequency modulated activation and/or a multi neuronal action which defines an associative mapping of the perceived signal.

Wang & Shamma (1995, p. 189) provide the most concise description of the currently accepted theory for the interpretation of auditory spectra. This is based on recently discovered '*feature maps*' within the primary auditory cortex:-

> *"Neurons in the brain usually exhibit strong selectivities to certain features in the stimulus. Those neurons having similar selectivities are also often systematically and topographically segregated. It thus appears that features are represented and mapped into response areas in the brain. One can tell what features are there by examining which response areas are triggered by the acoustic stimulus. The early auditory processing described above can also be interpreted as mapping the spectral contents (feature) of the acoustic signal on a spatial dimension (tonotopic axis)"*

What is evident is that it is to these theories that the technologist has to turn if artificial replication of human intelligence is to be pursued for the exploitation of its advantages.

2.8 The Cybernetic Approach

The origins and inspiration for the creation of cybernetics cannot be attributed to any one instance in scientific history. Many names and trains of inventive thought have contributed to its development.

In the mechanical sense the main inspiration has been that of feedback mechanism with the governor, as discussed by Clerk Maxwell in 1868, being possibly the most significant.

During the nineteen twenties, the hey-day of telemetry, theories of information transmission were suggested and discussed by such prominent scientists as Nyquist, Hartley and Shannon. 'The Hartley' subsequently became a unit defining a packet of telemetric information (now considered to be equal to 3.219 bits!) (Le Lionnais, 1966, p. 61).

In 1938 the cardiologist Rosenblueth and mathematician Wiener cooperated for the first time in an attempt to establish a relationship between these theories and the processing and transmission of information within biological systems (Le Lionnais, 1966, pp. 60-62).

In 1948, Norbert Wiener first adopted the term "Cybernetics" from the Greek κυβερνήτης meaning 'steersman' and published a book of the same name in 1961 (Wiener, 1961). He consequently defined the science of *"control and communications in the animal and the machine"*. The discussion and theories of this work concluded in four main themes,

1. In the area of information processing, there is a level at which animals and machines can be compared with validity.

2. The essential rules of human information processing and actuation can be derived from statistical rules of probability and Boolean Logic.

3. The principles of feedback control are valid for biological systems.

4. The structure, architecture and characteristics of biological brains needs to be artificially replicated to achieve an understanding of its processing capabilities.

Although these points were speculative and theoretical, and still the subject of much debate, they laid the foundations for the development of automated control and robotics.

The second theme can be seen reflected in the programming of expert and rule based systems which presently constitutes the majority of artificial intelligent systems applications.

The proliferation of the, now familiar, modern personal computer has hampered the development of processing architectures based on Wieners fourth theme. Only recently, with the urge to develop faster more robust systems, has attention been given to parallel processing techniques instead of the Von Neumann derived serial approach.

The disciplines involved in the study of cybernetics are broad. George (1977, p. 13) discusses the amalgamation of *"parent"* sciences which constitute cybernetic reasoning and highlights the six main areas as Psychology, Physiology, Physics, Logic, Mathematics and Engineering.

The engineering aspects involved in biological replication are mainly understood. The neuropsychological methods of information processing and interpretation are the main thrusts of interest and George (1977, p. 10) succinctly states:-

> *"In cybernetics we are primarily interested in a special class of automata, namely those which learn from experience."*

It is this approach, to replicate experiential learning, that forms the basis for such areas in artificial intelligence as neural network technology.

2.8.1 Control Theory, Actuators and Transducers

The principles adopted in automatic closed loop or feedback control are well documented and have become an established technology. It would be impossible to cover all aspects of control theory and the reader is advised to refer to given texts for specific information.

In general, control techniques utilise open loop methods whereby a direct command is give to the process with no means of monitoring, or closed loop using a feedback reading to compute an error and form a further corrective signal.

The four fundamental elements within closed loop control are,

1. Comparator element
2. Actuator element
3. Process element
4. Transducer or feedback element

In simple terms the desired output signal is registered by the comparator through which the signal is passed to the actuator. The actuator applies the desired value to the kinetic or process parameter via any suitable feed forward element. The transducer monitors the process variable and the signal measured is passed back to the comparator. The comparator assesses the difference between the desired and measured value to form an error. This error is used as a correcting signal and is passed to the actuator for appropriate adjustment. The process element yields its own inherent characteristics depending upon its generic type, i.e. fluid flow, thermal, mechanical or electrical etc.. The method of error adjustment is therefore subjective and selected so as to yield optimum response time, accuracy or offset and stability criteria. The optimum control solution is obtained through the derivation of a mathematical model or transfer function of the overall system and analysing its response to forced input functions.

In classical analogue control theory the adjustment signal can be a proportional (P), integral (I) or derivative (D) function of the error. Combinations of these functions, PI, PD or PID are possible (*compound action*) and they are derived from the error by the comparator or feed forward element. The effect of using these various modes of control influences the offset, response times and the degree of oscillation experienced about the desired set point.

The method of signal transmission can be either analogue or digital, depending on the method of comparator assessment of the signal and type of actuator and transducer.

Within digital control systems the comparator element is fulfilled by the digital computer or simple logic systems, utilising programmed statistical rules to analyse signals acquired by monitoring transducers via an analogue to digital converter (ADC). The correction signals can be derived and delivered to the actuators via a digital to analogue converter (DAC). The techniques associated with signal conversion are described in section 2.8.2.

Transducer technology to monitor parameters offers a plethora of devices for measuring process variables and kinetics.

Control Systems for Submerged Arc Welding

The control of submerged arc welding entails, primarily, the control of kinetic actuators and power source adjustment.

For position and seam tracking, stepper motors are utilised and for travel speed, variable speed d.c. motors.

Stepper motors operate on pulse input signals, which can be generated digitally and transmitted via a serial port and suitably amplified to drive the motor. The stepper motors, unless monitored , typically by rotary shaft encoders, constitute an open loop system where no feedback is used. Travel speed utilises a pre-calibrated power/speed ratio motor and unless monitored by a tacho-generator also constitutes a pre-set open loop system.

Apart from the general kinetic control, welding systems depend upon the optimum control of the applied power source (current and voltage). The control of current is also with a variable speed d.c. motor to change the rate of feed of the sacrificial electrode, although not strictly the same thing, the close correlation between current demand and wire feed speed make it a possible option. Varying types of power sources (including manufacturers versions of the same specifications) tend to exhibit different characteristics in terms of accuracy and in response and reset times.

The response time necessary to control a welding process varies with welding speed and/ or the size of the weld pool. The faster the weld or the smaller the weld pool the need for accuracy increases, which necessitates a faster response time. Houldcroft (1977, p. 291) makes the "*not unreasonable assumption*" that:-

> *".....a weld pool can accommodate a disturbance lasting no longer than the time taken to advance by, say, one-tenth of its width."*

Transducers for Sound Sources

The monitoring of sound sources utilises microphones, the correct selection of which depends upon,

1. Frequency response
2. Signal to noise ratio
3. Sound direction
4. Sound intensity (microphone sensitivity)
5. Atmospheric conditions

The characteristics of microphones are defined by their operation. Although most rely , at the first stage, on the sympathetic resonance of a diaphragm there are four principle types:-

Dynamic

The diaphragm is connected to a coil which passes through a magnetic field. The induced voltage corresponds to the frequency and amplitude of the movement of the diaphragm. These microphones exhibit a very uniform frequency response but, however, are very susceptible to external magnetic fields.

Piezoelectric / Crystal

In this case the mechanical coupling of the diaphragm invokes a stress on a piezoelectric material thereby producing changes in electrical charge. Due to the nature of the piezo crystal or ceramic, these microphones are easily damaged by high temperature or humidity.

Carbon

The vibration of the diaphragm compresses or relaxes on a pocket of fine carbon granules. This action decreases or increases the carbon module's electrical resistance respectively. Although providing high efficiency in terms of amplified voltage output, these microphones provide a very poor signal to noise ratio.

Condenser / Electret

The diaphragm constitutes the conductive plate of a capacitor with a voltage applied to a high resistance fixed backplate. The movement of the diaphragm causes a change in capacitance and consequently a change in the measured charge. The result is a system yielding an excellent uniform frequency response. The low output signals delivered, however, necessitate the use of high grade amplification in close proximity.
(Prout & Bienvenue,1990, pp. 127-133)

The direction of sound can be compensated for by selecting uni-directional or "free-

field" types which discriminate against potential masking noises. Otherwise, pressure type microphones can be used which offer a good frequency response to sound arriving at random incidences to the diaphragm. (Beranek, 1986, p. 165)

In general, manufacturers produce sound transducers of varying specifications, frequency responses / sensitivity are often tailored for specific applications.

2.8.2 Digital Data Acquisition

Continuous or analogue signals need to be converted to a discrete time series for compatibility with digital systems.

When digitising continuous signals, a sampling theorem has to be followed to ensure a valid representation of the original signal. The most commonly accepted theorem is that of Shannon in (1949, pp. 10-21), however, some texts credit Whittaker in 1915 for its resolvement (see Stearns & Hush, 1990, p. 49). Whichever stance you take, in essence, the theorem states that for a signal of maximum frequency X Hz the sampling frequency must be at least 2X Hz to be accurately representative in its discrete form. The frequency 2X Hz is known as the Nyquist frequency, and a lowpass filter with a cut-off frequency of X Hz needs to be instigated before digitising. If the input signal has frequency components above half the Nyquist, *aliasing* occurs where these unwanted frequencies are mirrored into the original signal resulting in a distorted, fake representation.

The accuracy of the converted signal is also an appreciative problem. Depending upon the time period between each sample and the resolution of the discrete binary scale, *quantisation errors* will occur as illustrated in figure 2.19 (Marvin & Ewers, 1994, p.48).

Hardware criteria for analogue to digital (ADC) data conversion depends on the required speed and resolution. The most common methods include successive approximation, dual slope, flash converters and sigma delta ADCs. All these methods are well described in most digital signal processing (DSP) texts and need not be elaborated here.

Following the digitising technique the discrete samples can be readily stored and mathematically analysed by high speed computer methods, as described in the next section.

2.8.3 Signal Processing

Signals acquired by monitoring transducers have to be conditioned to be made compatible with comparator hardware, and processed to enable valid analysis. The four main targets of signal processing are,

1. statistical analysis
2. recognition
3. prediction
4. modelling (Martin, 1991, p. 21)

The prime aim is to deliver a mathematical description of a signal which enables replication and a consequent level of understanding of a particular phenomenon.

The analysis of continuous or analogue waveforms, as obtained from microphones, can be subject to active or passive filtering to eliminate unwanted frequencies within the signal. Typical arrangements use highpass filters to eliminate low frequency interference such as mains noise; bandpass, stopband and notch filters to isolate individual bandwidths for further study; and lowpass filters to eliminate high frequency interference and to act as an 'anti-aliasing' filter prior to digitising. Once digitised, the discrete time series signal can be subjected to further filtering with mathematically modelled filters via software processing.

Signal Filtering

The principle objective of filtration is that of *signal recovery*. Unwanted frequency components or transient noise can be removed to reveal the fundamental signal shape and duration. Methods of filtering are many and well documented but all techniques have inherent characteristics of stability, frequency response and degrees of inflicted phase shift. Filters are designed for specific applications and chosen for optimum performance in the above mentioned areas.

Generally signal filters, either analogue or digital, are time invariant where the system coefficients are fixed. However, adaptive filters have been developed in which the coefficients can be controlled and changed with respect to the incoming signal. These are often used in conjunction with very high precision sensors to monitor interfering signals which can then be subtracted from the original signal. In this way changes in the frequency of the noise is compensated for. (Marvin & Ewers, 1994, pp. 119-120)

Further manipulation can include statistical analysis, correlation with other signals or transformation into other domains to enable further study.

Correlation

The correlation of two signals is considered as a measure of association. Two methods are adopted, the *auto-correlation function* (ACF) and the *cross-correlation function* (CCF). The ACF is especially relevant in the comparison of random signals in the time domain and can be defined as a measure of the average product of a signal and a time displaced version of itself. The continuous ACF follows the form :-

$$r_{xx}(k) = \lim_{N \to \infty} \frac{1}{(2N+1)} \sum_{m=-N}^{N} x_m \cdot x_{m+k} \qquad \text{Eqn. 2.8.3.1}$$

where, x_m & x_{m+k} are sample values separated by kT seconds for $(2N + 1)$ samples and the m parameter taken between +N and -N. The expression r_{xx} is standard notation for the ACF (Lynn, 1977, p. 85).

For a finite time series the expression can be annotated as:-

$$Y(n) = \sum_{k=0}^{K-1} x(k) \cdot x(n+k) \qquad \text{Eqn. 2.8.3.2}$$

where,

K = the finite length of the sequence x(k)

n = the current time sample index

The cross-correlation function (CCF) is very similar apart from the fact that it compares two different signals instead of a time shifted version of the same signal. In its discrete finite form it is expressed as:-

$$r_{xy}(\tau) = \sum_{-\infty}^{\infty} f_1(t) \cdot f_2(t+\tau) \qquad \text{Eqn. 2.8.3.3}$$

where, f_1 and f_2 are the two signals displaced by τ seconds. Note also the change in notation, r_{xy} as opposed to r_{xx} for the standard notation for the CCF.

Convolution

The convolution integral is a means to determine the behaviour of linear filter systems. The technique is similar to the finite cross-correlation process except that the comparison is between the input signal and the time reversed version of the impulse response characteristic of the filter system. Convolution can be expressed as :-

$$f_2(t) = \sum_{-\infty}^{\infty} f_1(T) \,.\, I(t - T) \qquad \text{Eqn. 2.8.3.4}$$

where, I is the impulse response characteristic (Lynn, 1977, pp. 145-147).

Transforms

Domain transforms are designed to provide a mathematical model of the signal for manipulation or to produce another descriptive state to isolate or highlight salient characteristics.

In sound analysis, the emphasis has been towards the study of the frequency domain and consequently the optimisation of the Fast Fourier Transform (FFT) to examine the frequency content of a complex time series signal. This technique is described in section 2.8.3.1.

Further processing and analysis within the frequency domain can be meaningful in certain instances. A technique very common in speech analysis is the use of *cepstral coefficients*. In effect the FFT is considered a time domain waveform with a temporal progression which can be subjected to a further FFT process as given in section 2.8.3.2.

In the realms of condition monitoring, however, a recent interest has developed in *Wavelet transforms*. Originally devised to aid seismic signal analysis, the time series signal is able to be decomposed into a series of descriptive components. A recent comparison by O'Brien and Macintyre (1994, pp. 75-86), compared wavelets and the more common and proven FFT in the condition monitoring of boiler feed pumps via vibration data. The wavelets are tested as an alternative to a windowed FFT on the basis that localised information is discarded during the more global transform of Fourier analysis. However, wavelets have the disadvantage that cueing has to occur. That is, a comparison of signals is only valid if they start at the same position on the time base. This technique is therefore more suitable for constant and cyclic time series patterns where thresholds or other techniques can be used to trigger the analysis.

The transients associated with arc welding acoustics make pattern recognition in the

time domain more complex. Cueing methods become severely problematic due to the sporadic nature of the signals.

Wavelet transforms have, however, been used as a method of data compression. As such the technique could be used within the frequency domain, where the spectral coefficients are relatively scaled, cued in terms of bandwidth and are generally more stable. The application of the wavelet transform is described in section 2.8.3.3.

Other transforms regularly encountered are Z and LaPlace transforms, both of which transform the time series into a mathematical description in terms of Poles and Zeros. The Z transform can be considered a generalisation of the Discrete Fourier Transform (DFT), as La Place can be to the continuous Fourier Transform (Stearns & Hush, 1990, p. 124). The Z transform is used to avoid convolution in the time domain and LaPlace transforms mirror this action in the complex annotated frequency domain. (Martin, 1991, p. 399)

2.8.3.1 The Fast Fourier Transform

The biological interpretation of sound involves the sympathetic resonance of the basilar membrane and thereby the extraction and interpretation of salient frequency bandwidths. The analysis of complex time / amplitude signals in the frequency domain can be achieved with the use of the Fast Fourier Transform (FFT) which can be considered a limited artificial replication of the biological signal processing system, as exhibited by the cochlea.

The FFT is based on the Fourier *series* originally developed by the French mathematician and Physicist Jean Baptiste Fourier. The principle is that a complex periodic signal can be considered the sum of a fundamental frequency and a number of harmonically related frequencies of differing amplitudes. Therefore the complex waveform can be expressed as a series of sine and cosine waves by the formula,

$$x_{(t)} = \frac{a_o}{2} + \sum_{k=1}^{k} (a_k \cos k\,\omega_o t + b_k \sin k\,\omega_o t) \qquad \text{Eqn. 2.8.3.1.1}$$

or can be expressed as a complex such that,

$$x_{(t)} = \sum_{k=-k}^{k} C_k \cdot e^{j(k\omega_o t)} \qquad \text{Eqn. 2.8.3.1.2}$$

and hence,

$$C_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} x_{(t)} \cdot e^{j(-k\omega_o t)} \cdot d(\omega_o t) \qquad \text{Eqn. 2.8.3.1.3}$$

where,

ω_o = fundamental frequency = $2\pi f_o$

k = harmonic number

a, b = least square coefficients of $x_{(t)}$

C_k = Fourier coefficient

The series describes a continuous periodic signal, however, real world signals have changing phase and are essentially aperiodic. For aperiodic signals of finite energy the Fourier Transform is derived. As the frequency content of the signal tends to infinity so the number of descriptive phasors also tends to infinity and the spacing between them , ω_o , tends to zero. Consequently the Fourier coefficient, C_k , also tends to zero. Defining a frequency function,

$$\mathbf{X}_{(f)} = \frac{C_k}{f_o} \qquad \text{Eqn. 2.8.3.1.4}$$

then,

$$x_{(t)} = \int_{-\infty}^{\infty} \mathbf{X}_{(f)} \cdot e^{j(\omega t)} \cdot df \qquad \text{Eqn. 2.8.3.1.5}$$

and hence,

$$\mathbf{X}_{(f)} = \int_{-\infty}^{\infty} x_{(t)} \cdot e^{j(-\omega t)} \cdot dt \qquad \text{Eqn. 2.8.3.1.6}$$

The digitising of a continuous signal over time results in a discrete time series acquired in accordance with the sampling theorem. By replacing the continuous time value, t, with a discrete series of N steps then the Discrete Fourier Transform (DFT) can be defined and expressed as,

$$\mathbf{X}_{(f)} = \sum_{n=0}^{N-1} x_{(n)} \cdot e^{\frac{j(-\omega n)}{N}} \qquad \text{Eqn. 2.8.3.1.7}$$

Where $x_{(n)}$ consists of discrete steps or points. The exponential term is often referred to as the 'twiddle factor', and is given the term W_N^{nk}, so that the DFT is expressed as,

$$\mathbf{X}_{(f)} = \sum_{n=0}^{N-1} x_{(n)} \cdot W_N^{nk} \qquad \text{Eqn. 2.8.3.1.8}$$

It can be seen that as N increases the computational requirements also increase which can have a detrimental effect on the real time application. For this reason an optimised technique for the derivation of the DFT has been developed known as the Fast Fourier

Transform (FFT). The mathematical description remains the same and only the method of computation is altered.

The efficiency of the FFT relies on the composite value of N which is generally a power of 2, where the power is known as the 'order' of the FFT and the value is indicative of the resolution of the resulting spectral plot. This composite approach enables the algorithm to be applied as a repeated computation of an elementary element known as a 'butterfly' (Stearns & Hush, 1990, pp., 102-124 : Papamichalis, 1990, pp., 53-69). It is not necessary to explain the intricacies of these techniques as they are well documented, varied and sometimes hardware specific. The general result is an optimised version of the DFT which provides the possibility of real time application.

The resulting coefficients provide a spectral plot of the component frequencies and relative amplitudes. With appropriate normalisation and logarithmic scaling, amplitudes are obtainable as a dB scale.

Due to the discrete nature of the data acquisition, errors are naturally present between the true signal spectrum and the derived spectrum because of the finite number of points considered. This error, often termed '*spectral leakage*' can be alleviated by 'shaping' the analysis window which is applied to the time / amplitude signal. The effect of windowing algorithms is a spectral plot with a highlighted fundamental frequency and an exponential (or similar function) roll-off towards the extremities of the spectrum. Windowing techniques such as , Hanning, Hamming and Blackman reduce the spectral leakage at the expense of bandwidth selectivity. That is, the roll-off shaping can mask salient frequencies if they occur close together within the spectrum. (Martin, 1991,pp. 272-279)

By applying subsequent time windows to the time / amplitude signal and calculating the associated FFT spectrum a series of time variant FFTs are produced. The result is a three dimensional contour mapping with axes of amplitude, frequency and time.

Generating and analysing such information is possible in real time with the application of dedicated digital signal processing (DSP) hardware. Systems such as the Texas Instruments TMS320C** series utilise application specific high level language programming to perform adaptive DSP techniques (Papamichalis, 1990, pp. 31-49).

The fundamentals of hardware DSP platforms are documented in a future section.

2.8.3.2 Cepstral Analysis

The term *cepstrum* is a derivation of the word *spectrum* (with the first four letters reversed). It was first described by Bogart *et al* (1963, pp. 209-243) with a definition given as the power spectrum function of the logarithmic power spectrum of a given time amplitude signal. This basically amounts to generating two frequency domain conversions in succession. Although logically this appears (and theoretically is) an FFT^2 function, the secondary FFT is often referred to as an inverse FFT (iFFT) which can lead to some confusion.

In keeping with the adoption of confusing terms, further analysis and processing within the cepstral domain are also described by syllabically re-arranged words:-

frequency	becomes	queferency
phase	becomes	saphe
amplitude	becomes	gamnitude
filtering	becomes	liftering
harmonic	becomes	rahmonic
period	becomes	repiod

Although others are mentioned these are the main areas where further analysis takes place.

The reasons for using such a domain is that it provides a method of de-convoluting a complex signal. Such signals are often found in the areas of speech recognition, seismology and hydroacoustics (Childers et al, 1977, p. 1437). Perhaps the most graphic illustration is its use in speech recognition.

Vocalised speech is considered to be the three fold convolution of signals generated by a vocal impulse train (of short duration), a glottal impulse response and a vocal tract impulse response.

If the time amplitude signal of the speech is suitably windowed and subjected to an FFT of suitable order with a log amplitude scaling, a typical trace is obtained as shown in figure 2.20. The responses of the vocal system are present as amplitude modulations within the frequency spectrum. By treating the spectrum as a 'time amplitude' signal and subjecting it to a further FFT function, these excitations can be isolated as shown in figure 2.21 (Marvin & Ewers, 1994, pp. 172-174).

2.8.3.3 The Wavelet Transform

Discrete Wavelet Transforms (DWT) have already been mentioned in section 2.8.2 as a method of signal transform. It is similar to the FFT in that it is a linear operation on a data vector whose length is an integer power of two. The transform yields a numerically different vector of the same length (Press, *et al*, 1992, p. 591). Within the FFT the length of input vector dictates the number of frequency components that can be represented. In the DWT the length determines the number of wavelet levels, as the DWT process follows a hierarchical format (Newland, 1993, p. 298).

In the FFT the fundamental operators are functions of sines and cosines where the input data vector is iteratively convolved with these functions to determine the frequency content. With the DWT the input signal is convolved with a matrix of orthogonal wavelet coefficients which are chosen so as to reflect the desired conversion scaling and accuracy.

The wavelet coefficient values have been calculated by Daubechies (1990) and therefore already available for practical applications. The proofs associated with the derivation of the coefficients are well documented in Newland (1993, pp. 295-396) and need not be discussed here.

The matrix of coefficients can be considered as a 'smoothing filter'. When applied to an input vector the action of the orthogonal transform matrix performs two related convolutions, decimates each of them by half and combines the remaining halves by interleaving the data. The interleaved data can be rearranged by grouping every other data point (thereby relieving the interleaving) and then the resulting vector is subjected to the matrix again. Figure 2.22 shows the general technique for applying the matrix with the resulting transformed vector.

The final vector always results in two 'smoothed' coefficients, denoted by S_1 & S_2 and a hierarchy of 'detailed' coefficients denoted as D_n, D_n and d_n etc.. In this format the S components are known as the *mother-function coefficients* and the D - d components the *wavelet coefficients* (Press, *et al*,1992, pp. 592-595).

The original signal can be reconstructed simply by applying the transpose of the matrix to the final wavelet transform vector the appropriate number of times. The function known simply as the inverse DWT (iDWT).

This alternative way of breaking down a signal into its constituent parts provides some advantages over the FFT in some instances. The integral action of the FFT takes into

account the duration of the whole signal as it acts from $+\infty$ to $-\infty$, therefore any local oscillation in the time domain, which may represent an important feature, will be averaged out during transformation. The wavelet transform can retain this local information as a scaled wavelet coefficient (Newland, 1993, p. 295).

Perhaps the most useful application of wavelet transforms is that of data compression. The transform produces a hierarchy of wavelets often in terms of amplitude. The amplitude of the wavelets can vary to the extent of making some negligible in value in comparison to the largest coefficient values. By applying a threshold to the vector and zeroing the most negligible, a sparse data set is produced. Subsequent iDWT action with nulled coefficients can reproduce the original signal with minimal disturbance.

It is important, however, to note that the sparse vector is not compressed in length but in amplitude only. That is, when compressing a function with wavelets it is necessary to record not only the amplitude of the non-zero components but also their position within the vector if the iDWT is to remain valid.

2.8.4 Artificial Intelligence (AI)

The problems associated with replicating human intelligence are compounded by its unknown and still much discussed psychological and neurophysiological definition.

As well as the hypothesis proposed by Wiener in 'Cybernetics' (Wiener, 1961), the concept of artificial intelligence and how to achieve it was discussed by two prominent mathematicians, John von Neumann and Alan Turing during the 1950s.

In his book " The Computer and the Brain" (Neumann, 1958, pp. 478-491), von Neumann discusses the mathematical theories of analogue and digital computing in comparison with known neuronal action. The comparison continued into the area of engineered hardware componentry and biological systems. It was concluded that with the known maximum speeds of neuronal switching being 1000/second, artificial hardware had a 10^4 - 10^5 advantage of speed. However, the shear volume of biological elements and parallel interconnections heavily counteracted this advantage in terms of physical space and computing power. Today this has been relieved to a certain degree by the development of VLSI devices, but the development of parallel architecture still remains.

Alan Turing proposed what has become known as "The Turing Test", as a method of deciding if a computer can be considered intelligent (Turing, 1950, pp. 492-519). The test involved the human interrogation of the machine and another human via printed messages. The interrogator has to decide on the different sexes of the entities he is questioning and it is the object of the machine to lead the interrogator to the wrong decision.

This level of intelligence testing is still under debate, however, protagonists of artificial intelligence strive to replicate a so called '*conscious*' level of operation with the ability to generate innovative and imaginative thought.

There are many theories as to how this may be achieved, that is to copy the creative dexterity of human thought. Creativity has been seen to be based on the combination of existing ideas to formulate a new, innovative solution to a problem. Other creative attributes are, the ability to plan a sequence of actions without prompting (especially so in intelligent prognosis), learning from experience and the ability to generalise. The generalisation allows the formulation of a specific answer in response to imprecise or inferred information. (Taylor, 1988, pp. 39-40)

What is evident is that the mind does not consist of explicit, pre-programmed rules.Many applications of artificially intelligent techniques fail to replicate this aspect of biological

systems.

In the area of robotics and control, 'intelligent' action has to be the result of a judgmental assessment of incoming data from external sensors or transducers. Such signals are often noisy or corrupt and require flexible or highly adaptive analysis. When monitoring a process the emphasis is on the recognition of signals indicative of a process state and ultimately, with signal enhancement, pattern recognition.

Statistical correlations and the application of template matching, for example, can be utilised for relatively stable, non-transient patterns. Highly correlated data presents a classification problem where clearly defined class boundaries are unobtainable due to noise corruption or simply the inherent characteristics of the acquired data set. The anomalies or 'outliers', which cannot be locally classified, could be dealt with by statistical methods such as multi-pass linear regression or piecewise-linear discriminants where mathematical rules are added to the algorithm. Some of the more widely adopted methods are discussed in section 2.8.5.

Time dependency is a fundamental issue in the field of AI. The term 'real-time operation' is naturally subjective to the application. In an article entitled "The Challenges of Real-Time AI", Musliner *et al* (1995, pp. 58-66) states that real-time AI systems must be capable of:-

1. working continuously over extended periods of time
2. interfacing to the external environment via sensors and actuators
3. dealing with uncertain or missing data
4. focusing resources on the most critical events
5. handling both sychronous and asynchronous events in a predictable fashion with guaranteed response times
6. degrading gracefully

It is apparent that all these areas have been tackled to some degree. However, the AI fraternity is still in a state of flux in its attempts to satisfy these requirements at a speed acceptable for automated control and robotics.

There are three general approaches adopted within the field of monitoring and control, which attempt to provide an artificial decision making process, *knowledge engineering* or *expert systems*, *fuzzy logic* and *artificial neural networks*.

Knowledge Engineering or Expert Systems

As the names imply, the application of knowledge engineering techniques utilises the combined experience and knowledge of human experts within a particular field. Taking into account the initial conditions for a process or problem, the expertise is encoded as a set of programmed rules. The input parameters are correlated with the knowledge base in anticipation of formulating an optimised solution.

Problems arise with input parameters which do not conform to the programmed rules, that is an occurrence not accounted for and previously unseen. Various techniques have been employed in programming expert systems to achieve the flexibility of operation required in dealing with such anomalies. Statistical inference engines using Bayesian probabilities add a degree of flexibility but still require *a priori* knowledge derived from a large data base to formulate a reliable solution. In general these systems operate on a hierarchical program structure of nested loops and iterative searches, therefore, anomalies are dealt with by expanding the knowledge base. As a consequence the more complicated the problem the more computationally intensive the process becomes, reducing the feasibility of real time application. Expert systems have limited use outside a narrow field of expertise and are often incapable of dealing with novel data.

Expert systems have been used prolifically within the welding industry. However, a recent survey by Pecas & Quintino (1993, pp. 179-186), indicates that these systems are mainly utilised in off-line situations such as data-base management roles where procedural or defect assessments are required. The five areas listed are:-

1. Procedure generators
2. Processes, consumables and equipment selectors
3. Risk evaluation
4. Process/ equipment diagnostics
5. Defect assessment

The area of process/system diagnostics is the only area with a time dependency aspect, all the other areas are 'off-line' analytical and predictive situations. Further evaluation has revealed that approximately only 13% of expert system technology for welding is dedicated to possible 'on-line' applications, with a meagre 4% being commercially available.

At a more specific level, Doumanidis (1994, pp. 13-24) has reviewed methods of control for robotic welding. In what is termed a 'representative' survey it was found that 69% of the methods were classical analogue open or closed loop control of the power source, 25% were described as adaptive techniques utilising numerical/analytical models and only

the remaining 6% were based upon fuzzy logic and artificial neural networks.

Fuzzy Logic

Fuzzy logic is a continuation of *vague* or *multi-value* logic as discussed by Bertrand Russell in the early part of the 20th century. The associated mathematical theory and application was labelled 'fuzzy' by Zadeh (1965, pp. 338-353) . The proposal was based upon and explained with Venn diagram sets. A Gaussian relationship is applied to the class or set boundaries in an attempt to avoid unclassifiable outliers within novel data. This provided a limited generalising capability, however, fuzzy systems are essentially a form of expert system and as such comply to mathematical rules. As such, problems are apparent when dealing with highly correlated data which are not entirely linearly separable. (Kosko, 1994, p. 290)

Fuzzy logic has experienced a high level of commercial acceptability, especially in the Japanese consumer electronic market, due to its relatively easy implementation (Burley, 1991, pp. 25-28). In more complex control scenarios the trend is towards a hybrid approach instigating neural network techniques in conjunction with fuzzy math known as neuro-fuzzy controllers. In a welding context Ohshima *et al* (1993, pp. 188-193), have developed such a system for pulsed MIG welding. A CCD camera output is digitised and subjected to a neural network assessment in which the inputs are discrete measurements across the visual weld pool. The neural network categorises the data and provides an output as a penetration estimate. The information is passed to the fuzzy based comparator element to provide the necessary corrective control signal.

Artificial Neural Networks

The recent advent of Artificial Neural Networks has provided a relatively high speed adaptive technique with the necessary generalising capabilities for processing transient signals. As a result they have offered a new method of approach in real time signal processing and control and are discussed in more detail in section 2.9.

2.8.5 Statistical Pattern Recognition and Decision Making

The previous sections were mainly concerned with the pre-processing of 'raw' signals into a state compatible with digital manipulation. This section attempts to discuss existing methods of analysing these signals, so as to highlight salient features in order to group and classify them.

As already discussed in section 2.7, human perception of the of the surrounding environment is often based on the collection of discrete objects which can be segregated into 'recognisable' classes. The creation of the boundaries for these classes is known in pattern recognition (PR) as *segmentation.* The fundamental questions posed are; What are the features which enable the classification ?, What classes exist within the data ?, Is there a degree of redundancy in the information ?, and Where do we draw the boundary line to allow discrimination between one class and another ?

The first two questions can be regarded as inter-dependent as there can be as many classes as there are features. However, what can be considered a feature is very subjective, and for any particular group of objects, the possibilities could be prolific. Nadler and Smith (1993, p. 223) have noted the purpose of features within a PR context and, in particular, state that:-

> *"They reduce the dimensionality of the space in which the actual classification is carried out, as compared with measurement space."*

and,

> *"They select useful measurements and summarize them, thereby enhancing the reliability of the recognition."*

The reduction in dimensionality is a critical requirement. The selection of just one feature could reduce the dimensionality of the object a million-fold but reduce the possibility of optimal classification by the same amount because of limiting the number of accounted features. In visual images the features extracted may be simply colour and/or shape, in time amplitude signals frequency and/or amplitude. Patterns may not always be categorised in terms of features which are a measurement of physical appearance or the result of an instantaneous impression. Patterns can change with time, and the temporal progression of a time series can itself become an inherent feature. In these cases patterns may be grouped in accordance with a common time variation. Whatever description is chosen, the ideal situation is to acquire a set of measurements unique to a particular class. In practical problems this is rarely possible, and the high correlation of complex

data patterns resulting in data sets which are linearly inseparable calls for an intuitive or statistically feasible decision process to classify them. To achieve this it is necessary to construct an analytical model or distribution function to mathematically describe the data.

Feature extraction is approached in either a structural or statistical way. Structural methods adopt a feature interpretation based upon semantics whereby some correspondence to human intuitive perception is practised. Statistical methods rely completely on number crunching, where the features are extracted and manipulated as numerical measurements (Nadler & Smith, 1993, p. 151).

Whatever approach is adopted, a decision has to be made as to the classification boundaries and the location of acquired patterns by means of feature grouping.

Decisions based upon statistical methods utilise either descriptive or inferential techniques to study the population distributions of acquired data patterns. The most common techniques used are:-

Sample and Population Variance & Co-variance Matrices.

Standard statistical measurements such as variance and co-variance matrices are directly related to the 'distances' between the mean of the population and the acquired sample variable. The variance is defined as:-

$$\text{var}(x) = S^2 = \frac{\sum(x - \bar{x})^2}{N} \quad \text{where} \quad S = \sqrt{\frac{\sum(x - \bar{x})^2}{N}} = \text{Standard Deviation} \qquad \text{Eqn. 2.8.5.1}$$

The co-variance is related in concept to the correlation coefficient in that it is a measurement of similarity and can be defined as:-

$$\text{cov}(x,y) = \frac{\sum(x - \bar{x})(y - \bar{y})}{N} \qquad \text{Eqn. 2.8.5.2}$$

Discriminants and Logistic Regression.

Any plane/hyperplane or surface/hypersurface which can be created to separate two or more clusters of data can be defined as a discriminant. The complexity of the function which defines the discriminant is entirely dependent upon the separability of the data set.

Figure 2.23.a shows a two dimensional data set which comprises of two known classes. The two classes exhibit a degree of correlation in that they overlap. A solution to the segmentation of the data would be to continually construct linear (or polynomial if

appropriate) discriminants as shown in figure 2.23.b. It can be seen, however, that the number of discriminants increases with the number of classes and the degree of correlation. In 'real' statistical problems the number of discriminants can be very large and in most instances there will be no reasonable set of discriminants that can separate the classes without error.

Clustering Techniques.

Nadler & Smith (1993, p. 294) define clustering as:-

> "*.....an attempt to find structure in a set of observations that we know very little about.*" (italics in original)

There are generally two approaches needed to deal with either known data types, where the data can be labelled and analysed, or a series of observations with unknown classes. Simply labelling the data can reveal the distribution of data classes and highlight 'compactness' and possible segmentations. Problems occur when the data is of an unknown type. Figure 2.24 illustrates this problem, the data shown is of unknown classification and the dilemma is; Is the data representing one cluster or two ? and, if two where does the segmentation boundary lie ?. Several methods exist for determining this information including 'minimising squared error', 'hierarchical' and 'graph-theoretical' clustering (Nadler & Smith, 1993, p. 294-330), but perhaps the most common is that of *k-Nearest Neighbour* (k-NN).

k- Nearest Neighbour Methods (k-NN).

This technique can deal with unknown boundary situations and is known as an '*unsupervised learning*' method as the classification of the objects is not pre-assigned. If a number of theoretical prototype classifications are created and assigned to an object at random, the probability of a 'near neighbour' of that object being of the same class is quite high. The 'k' refers to the integer number of neighbours taken into account for this inferred labelling process. What can be considered a near neighbour is based upon the minimum Euclidean distance or some other adopted distance metric within the feature space. By iteratively applying the labelling algorithm to the data set, a point will be reached where ambiguities occur between the classification of objects. These areas are considered the natural boundaries for the prototype classes. By applying a 'majority win' algorithm to these areas the segmentation can be further enhanced. If the distance between objects of separated classes is bisected, an optimised boundary line can be constructed as illustrated in figure 2.25.

The main problem with this technique is that the segmentation is based solely upon the distance metric used within the feature space. If the data is sparse and as a consequence

not truly representative of the pattern population, false clusters can be created.

Further methods can be adopted based upon the laws of probability.

Probability Density Functions and Distributions.

If the data consists of continuous, random variables (vectors) the distribution can be described as a *probability density function* (p.d.f) or simply a *probability distribution* if discrete variables are used. The probability measurement can be used as it is associated with the frequency of occurrence and, in this context, the number of patterns which occur within a given limit or interval. In this way it is possible to determine the probability of a random variable occurring within the interval.

An extension to this method is the use of *conditional probability* where a prior knowledge about the problem is used to determine the overall probability function. This forms the basis of *Bayes Theorem.*

Bayes Theorem (Bayesian Inference).

This theorem is basically a relationship between probabilities and can be expressed as:-

$$p(X|Y) = \frac{p(Y|X)\,p(X)}{p(Y)} \qquad \text{Eqn. 2.8.5.3}$$

where, $p(X)$ is the probability of X , $p(X|Y)$ is the probability of X given that Y has occurred, $p(Y|X)$ is the probability of Y given that X has occurred and $p(Y)$ is the probability of Y . Described in words this means :-

$$\text{posterior} = \frac{\text{likelihood x prior}}{\text{evidence}} \qquad \text{(Bishop, 1994)}$$

Because of the use of a prior probability, the initiation requires that a certain level of assumption be made. This can lead to a subjective and personalistic view of the problem based on the strength of an individual's expectation that a particular event will occur (Kirk, 1990, p. 234). It is, however, able to provide an optimal solution in most cases as a consequence of the constant update of probabilities as new information is received.

Markov Models.

Frequently, a set of observations is a time series in which a discrete sequence presents itself as a change in state from one observation to the next. If these transitions can be expressed in terms of probabilities then the process is called a *Markov* process or a *Markov Chain.*

This process depends on a finite number of previous states to be taken into consideration, with the number of previous states considered giving the 'order' of the Markov model (i.e. one past event 1st order, two events 2nd order...etc.). Further data is necessary concerning the probabilities of the process's initial state. From the anticipated sequence of events (commonly known as a *transition graph*) a probability matrix can be constructed. The rows on the matrix are considered the probability vectors for the time interval of events which, when matrix multiplied with the probabilities of previous and anticipated events, lead to a probability of the process final state (Nadler & Smith, 1993, pp. 398-403).

This method has found extensive use in the form of *Hidden Markov Models* (HMMs) for speech recognition. The generation of human speech consists of the projection of a time sequence of phonemes. The series of phonemes for a particular word can be built as a Markov model. The term 'hidden' is used here as it is generally impossible when using this method to analyse the state sequence that results in a particular observation (Wheddon & Linggard, 1990, p. 213).

The use of HMMs involves the phonetic analysis of a vocabulary of words (defining a training set), the construction of a suitable series of HMMs (training) and the performance testing on a series of unknown utterances (recognition). HMMs require large amounts of training data, especially in the case of speaker independent recognition, which leads to high computational demands (with consequential time constraints). However, accuracy as high as 98.1% has been recorded on a limited set of speaker independent spoken numbers (Wheddon & Linggard, 1990, p. 209-226).

Self-Organising Feature Maps (SOFM).

An SOFM is a type of artificial neural network, and as such is discussed in detail in section 2.9.2. The inherent properties of this method, utilising unsupervised learning, enables the automatic clustering of high dimensional data patterns. They are especially prominent in problems where no known classes exist within the data. Zhang & Li (1993, pp. 2448-2451) provide an insight into the use of such methods to segment and analyse the clustering of random data within the 'output layer' of the network. By analysing the output activations a 'density' of similar patterns can be noted where a lack of activation can be considered a natural boundary. The advantages and disadvantages of this method will be discussed in section 2.9.2.

2.8.6 Auditory Modelling

Substantial research has been dedicated to modelling the human auditory system. Most approaches have been developed around the replication of the basilar filtration and spectral analysis observed within the cochlea.

Software functions have been developed at the Loughborough University of Technology in the form of the LUTEar project (O'Mard et al, 1996). Utilising ANSI standard 'C' functions for UNIX and DOS platforms the core routines were designed as a modular approach to support investigations into auditory phenomena, speech processing and voice analysis. The comprehensive library of functions enables the modelling of multiple critical band filtration, basilar activation levels and neuronal conductivity, based upon the present understanding of biological systems.

A similar approach has been adopted by Slaney (1994). An 'Auditory Toolbox' for Matlab applications on a UNIX platform has been developed utilising various Cochlea, filtration and spectrographic models.

A VLSI implementation scheme has been proposed by Lazzaro & Mead (1996) to develop auditory models in hardware. These devices are being developed as voice recognition and auditory scene analysers for the manufacture of computer peripheral devices.

The approaches adopted here are very comprehensive studies into the specific areas of human auditory replication, where each biological stage is encoded as mathematical or transistor functions. The lack of commercial availability of VLSI devices and the complex software modules necessary to perform biologically inspired audition render them, at present, non-viable for real-time monitoring and control systems.

2.9 The Concept of Artificial Neural Networks

Artificial Neural Networks were conceived as an attempt to mathematically model and replicate the architecture and data transmission concepts of the biological brain. By modelling the action of a discrete neuron and using it as a basic building block, an architecture can be constructed with various configurations. The interconnections of the basic neuronal elements dictate the direction and path of communication.

Figure 2.26 illustrates the major parts of a biological neuron, with figure 2.27 showing a schematic of its mathematical, artificial counterpart. The communication of the synapses are represented by x1 - xn and constitute the input to the neuron. Mathematical weights w1 - wn are used to give a bias to the communication channels to the body or soma of the neuron, and hence control the significance of incoming information by means of a threshold. The soma has a mathematical transfer function F(I) to derive an overall output value, I, which is transmitted to another connected neuron.

Figure 2.28 shows an interconnected architecture of the basic neuronal elements. Information is presented at the input layer and transferred to neurons of each successive layer with an output layer providing an ultimate value.

By engineering the mathematical functions and method of interconnection an overall algorithm can be developed which, when presented with input information, can give an optimised value to the synaptic weights. It is this method which gives the neural network the effect of being able to 'learn' by example and recognise similar patterns. The architecture and mathematical functions are numerous and varied with the effect of providing different models to perform different tasks. The result is a highly adaptive mathematical method for processing information. The parallel nature of data transmission through the network promotes high speed operation once the 'training' or synaptic weight optimisation phase is complete.

The design, optimisation and performance assessment of neural networks has recently opened into a branch of artificial intelligence in its own right. The parameters for their successful use are still under heavy debate with new methods and algorithms becoming more prolific.

2.9.1 A History of Neural Network Development

Although practical applications of neural networks are relatively recent, its foundations precede that of digital computers and expert systems. A landmark paper by McCulloch and Pitts, "A logical calculus of the ideas immanent in nervous activity", was originally published in 1943 (McCulloch & Pitts, 1943, pp. 115-133) which, incidently, predates Wieners publication of 'Cybernetics'. Its influence was far reaching, providing inspiration for the development of three areas of practical artificial intelligence including John von Neumann and digital computers, Marvin Minsky and the development of expert systems and Frank Rosenblatt with 'microscopic' intelligence which later developed into the field of neural networks. This field has experienced a somewhat chequered history since then in comparison with its more highly developed counterparts. (Eberhart & Dobbins, 1990, p. 15).

In 1949, Donald Hebb made a primary contribution to neural network development by postulating that the information within neurons is held and controlled by the synaptic weights. That is repetitive patterns offered to the neuron increase the value of the weights consequently creating reinforcement. This reinforcement learning is also known as 'Hebbian' learning and constitutes a significant part of modern learning algorithms used today. (Eberhart & Dobbins, 1990, p. 17).

The Summer of 1951 saw the building of what can be considered the first simple 'neural computer'. Minsky and Edmonds constructed a device, from war surplus parts, and set it up to learn to run a maze. The conclusions drawn from the experiment, however, lead to the development of stored instructions to interpret data from external sensors to make decisions. As such, the system set the precedence for 'macroscopic' artificial intelligence or expert systems.

Frank Rosenblatt is considered to have developed the first 'learning machine' in 1958. The innovation was more impressive due to its simulation on an IBM 704 computer. The 'Perceptron' was inspired by Rosenblatt's studies of biological vision systems, in particular that of the fly. A retina of 400 photocells was randomly connected to a single layer of neuronal or 'associative ' units, which in turn were connected to a layer of response units. By implementing positive and negative feedback from the responsive to the associative units, a level of equilibrium was achieved which was capable of limited pattern recognition. The work also illustrated the learning mechanisms which exhibited 'self organisation' or 'self associative' traits which were to prove influential in future research.

The next landmark was courtesy of Widrow and Hoff in 1960, with the publication of

"Adaptive switching circuits" (Widrow & Hoff, 1960, pp. 96-104). It introduced a hardware system known as the 'Adaline' or 'Adaptive linear' neuron. The system utilised a +1 and -1 attribute to the synaptic weights as well as an added bias weight to the neuron, and as such created a more adaptive network which was faster in response to the training phase of the set-up. The method also presented the use of a least mean squared (LMS) error value to adjust the synaptic weights. Developing the system further concluded in the 'Madaline' or ', Multi adaptive linear' neuron where the fundamental elements were utilised in layers. The benefits of this configuration are still evident in the telecommunications field where the madeline is utilised in noise reduction of speech circuits (Widrow & Glover *et al*, 1975).

Marvin Minsky and Seymour Papert produced a damning critique of neural networks, and in particular Rosenblatt's perceptron, in 1969 with the publication of the book "Perceptrons" (Minsky & Papert, 1969). Today, the criticisms are considered somewhat political due to the differing and competitive nature of the areas of research. However, the conclusions drew on the negative aspects of the perceptron and in particular its inability to differentiate between solutions which are non-linear, such as the simple Boolean logic function, the Exclusive-OR. As a result neural network research received little funding or attention for the next decade.

During the relatively quiet period that followed, a retrospectively significant step was independently contrived by James Anderson and Teuvo Kohonen during the early 1970s. Anderson's 'linear associator' was based upon Hebbian learning, and the final descriptive state of the trained network was one of n- dimensional space where each dimensional axis was represented by a neuron. The technique was known as Brain-State-in-a-Box (BSB) as the dimensional representation was imagined to be a three dimensional box. Kohonen extended this theory by adapting the learning algorithm so that synaptic weight update was dependent upon its previous value. Perhaps the most significant development of this work was a competitive learning model where processing neurons compete to win or fire when confronted with an input pattern. This resulted in a grouping of similarly valued neurons within the final network. In effect, a method of self-organisation, or 'unsupervised' learning, was achieved similar to the biological mapping of various areas of the brain (Wasserman, 1989, p. 23).

A general revival in the interest of neural networks was instigated by John Hopfield from 1982 to 1984. By describing versions on neural networks in a hardware rather than a biological context, interest was generated amongst engineers rather than pure theorists. By expressing a binary input, but utilising continuously valued neurons with a sigmoidal non-linearity, Hopfield networks trained by attempting to find a state of synaptic equilibrium for each pattern presented to the network. Problems were encountered,

however in that the successive presentation of patterns disrupted the weight values of the previous. The recursive or iterative nature of the training was one of finding a global optimisation of all the weights with respect to the input pattern set. The problem of locating a local rather than a global optimisation still remains, but the concept of the Hopfield network was carried on into further development and is still utilised in modern networks such as back-propagation.

Perhaps one of the most significant steps towards popularising neural networks was the publication of the volumes 'Parallel Distributed Processing' by Rumelhart & McClelland (1986). As well as comprehensively documenting the practical aspects of neural networks, it included the derivation of the back-propagation learning algorithm for multi-layer perceptrons. The back-propagation method has since become one of the most widely adopted networks in practical applications. Originally proposed by Paul Werbos in the early 1970s and independently derived in 1982 by David Parker, the popularised method documented by Rumelhart and McClelland was able to solve the Exclusive-OR problem set by Minsky and Papert (Dayhoff , 1991, pp. 76-77).

To date, the interest and development of neural network paradigms and application has accelerated. Early attempts at application were primarily based around so called 'toy problems' in an attempt to bench mark the performance of networks. For instance, a 'broom-stick' balancer. In this test a broomstick is linked to a cart mounted on rails. By moving the cart from left to right at the appropriate speed and by a certain distance it is possible to maintain the broomstick in a balanced vertical or near vertical position. This is relatively straight forward with conventional analogue control methods. However, the Adaline system has been tested for this task in attempts to compare the techniques (Hecht-Nielson, 1991, pp. 342-345). Today, however, with an increasing industrial interest applications are varied, from financial forecasting to real time pattern recognition and signal processing.

The instigation of neural networks has tended to take either the path of computer software emulation or hard-wired VLSI operation. Artificial vision and auditory systems, as developed by Mead (1989), constitute dedicated hardware devices based on silicon technology to instigate a limited parallel architecture. Pattern recognition, based on video images, was achieved by Wilkey, Stonham and Aleksander. The 'WISARD' system utilised conventional RAM integrated circuits (Aleksander & Burnett, 1983). The key advantage of these systems is speed of operation but, with the exception of the relatively inexpensive components of the WISARD system, they are costly to develop. The main advantage of software simulation is one of flexibility and cost. The proliferation of PC products has enabled financially viable experiments in network architecture developments and trials.

2.9.2 Practical Paradigms and Learning Mechanisms

Studies of biological neural systems have revealed that differing neuronal topologies as well as individual neurons, perform different functions. The various characteristics of recognition and abilities in learning are replicated by ANN architectures and learning algorithms. The learning categories are generally of two types, supervised or unsupervised. Both these methods are utilised by two ANN paradigms which have, to date, emerged to become the most widely adopted methods in applications.

Back Propagation Networks (BPN)

BPNs provide a method of supervised learning . In essence the input data is given with a desired output, and the learning technique is one of re-inforcement or Hebbian learning.

The architecture is generally a three layer network consisting of an input layer, a middle or 'hidden' layer and an output layer (Fig. 2.28). Each layer is fully interconnected to the preceeding layer. The number of neurons in the input and output layers depends on the input pattern organisation and the intended output classification.

The number of hidden layers can vary, however, some mathematical analysis of the architecture suggests that three hidden layers are required to solve the most complex pattern classification problems (NeuralWare, 1993, NC-63). This is, however, contradicted in certain texts. Hertz *et al* (1991, p. 142) state that:-

> *"The answer is at most two hidden layers, with arbitrary accuracy being obtainable given enough units per layer."*

and Cybenko (1989, pp. 303-314) suggests that only one hidden layer is enough to model any continuous function. Hertz *et al* (1991, p. 143) conclude that the use of more than two hidden layers may allow an architecture with fewer neurons which may speed up the overall operation but may present problems in 'learning' complex functions due to the presence of local minima.

The number of neurons in the hidden layer/s is still the subject of much debate, but the optimum number is dependent upon the complexity of the classification problem. There are 'rules of thumb' for the selection of hidden units but the theories suggest a somewhat arbitrary approach . McCord-Nelson & Illingworth (1991, p. 171) suggest four hidden units for every input unit and Eberhart & Dobbins state:-

"We have found that a reasonable number to start with in many cases can be obtained by taking the square-root of the number of inputs plus output neurodes, and adding a few - which sounds like a recipe for baking a cake, perhaps, but it often works."

(Eberhart & Dobbins, 1990, p. 42)

In the training stages, the input vector components are transferred through the connected neuronal transfer functions via the synaptic weights. The resulting values given at the output layer are compared with the desired output. A function of the error is used to adjust the value of the synaptic weights thereby making a subsequently similar input generate an output closer to the desired value. The training can therefore be considered a two stage process.

The feed forward stage, where the input vector components are passed through the network from the input layer to the hidden layer/s to the output layer, typically utilises the following algorithm :-

$$\text{Output } (O_n) = F\left(\sum_i a_i \,.\, w_{ni}\right) \qquad \text{Eqn. 2.9.2.1}$$

where :-

O_n = n th neuron output

a_i = value of neuron i

w_{ni} = Synaptic weight value between neuron a_i and neuron O_n

The transfer function F is typically sigmoidal to give a smoothed step response or hyperbolic to obtain a bi-polar option. The addition of a constant to ALL neurons within the hidden layer/s can be used to act as a threshold or 'bias' term which shifts the transfer function with respect to the X axis. Other mathematical functions, generally trigonometrical, can also be utilised.

The error term generated at the output layer units is calculated as :-

$$\theta_n = (d_n - a_n) \,.\, f'\left(\sum_i a_i \,.\, w_{ni}\right) \qquad \text{Eqn. 2.9.2.2}$$

where :-

d_n = desired output value of n th neuron

a_n = actual output value for n th neuron

The derivative function f' of the weighted sum of inputs to neuron n generates a larger error value for those elements which exhibit a greater rate of change, with the aim of forcing the network towards a state of equilibrium.

The error value associated with neurons within the hidden layer, which constitutes the first step in the feedback stage, is consequently expressed as :-

$$\theta_h = (\sum \theta_n . w_{nh} . f'(\sum_n a_n . w_{nh}) \qquad \text{Eqn. 2.9.2.3}$$

The calculated error value of the hidden element is used to correct or update the connected synaptic weight values such that :-

$$\Delta w_{hi} = \eta . \theta_h . a_i \qquad \text{Eqn. 2.9.2.4}$$

where :-

Δw_{hi} = addition term to old weight value
η = a constant known as the 'learning coefficient'

The learning coefficient, η, reflects the rate of learning of the network, where large values can speed learning at the cost of increasing network instability and small values increase learning time. The learning coefficient is often decreased over a measured number of training epochs.

The success of the BPN model is dependent upon the distribution or the degree of correlation of the input data. Although ANNs have the inherent ability to deal with highly correlated data, the BPN model uses a gradient descent method in its learning algorithm to establish an optimum boundary between similar data patterns. If the separation is complex, it can often lead the network to converge on a local rather than a global minimum resulting in a failure to train or an unreliable output response. Various methods have been employed in an attempt to overcome these problems including the introduction of a 'momentum term' (Dayhoff, 1991, p. 78). This comprises the mathematical addition of a constant to equation 2.9.2.4 where the constant is a fraction of the previous weight change. Another common method is the injection of noise to the training data set and noise addition to the synaptic weight values during training.

BPNs have been referred to as the 'workhorse' of neural computing due to its popularity in research studies and wide field of application. These networks have proved a powerful method in mapping non-linear relationships or constructing non-linear transfer functions between input and desired output vectors. As a consequence uses have been found in data compression (Cottrell *et al*, 1987, pp. 461-473), signal processing

(Lapedes & Farber, 1987) and pattern classification (Gorman & Sejnowski, 1988, pp. 75-89) problems.

Self Organising Feature Map (SOFM)

The SOM model utilises an unsupervised or competitive learning technique and was originally proposed and developed by Teuvo Kohonen between 1979 and 1982. This method replicates 'learning by experience' rather than by taught responses. The SOM consists of two layers, an input layer consisting of a number of neurons suitable for the input data, which is fully interconnected to an output matrix or 'slab' of neurons. The number of neurons in the output slab depends on the required resolution of the desired output classification. Figure 2.29 illustrates schematically the typical architecture.

During the training phase, the input vector is compared with the randomly initiated synaptic values of each output neuron. The output neuron possessing the least error in vector or Euclidean space is considered the winner and its synapses are updated proportionally to the error as dictated by the learning algorithm. Furthermore, each winning neuron has neighbours within the output slab. These neighbouring neurons are also updated in such a way as to encourage or inhibit its chances of winning when a subsequently similar vector is compared. This training continues until an acceptable state of equilibrium is reached.

An input vector of n elements requires a minimum n input neurons and the input training data is considered n- dimensional where :-

$$E = [e_1, e_2, e_3, \ldots\ldots .e_n] \qquad \text{Eqn. 2.9.2.5}$$

and e_1 - e_n are the input neuronal values. Each input neuron is fully connected to all the output neurons such that the synaptic weights are considered as the vector :-

$$U_i = [u_{i1}, u_{i2}, u_{i3}, \ldots\ldots .u_{in}] \qquad \text{Eqn. 2.9.2.6}$$

where i is the matrix identifier of the output neuron. During training the input vector is compared with the weight vector and the winning neuron satisfies :-

$$\| E - U \| = \underset{i}{\text{minimum}} \{\| E - U_i \|\} \qquad \text{Eqn. 2.9.2.7}$$

The winning neuron in the output layer is considered to be at the centre of the neighbourhood to be updated. The shape and size of the neighbourhood are

predetermined, but generally considered initially to be at least half the total size of the output layer (Kangas & Kohonen, 1990, p. 93). The weights of the winning neuron and those contained within the specified neighbourhood are updated according to :-

$$u_{in}^{new} = u_{in}^{old} + \Delta u_{in} \qquad \text{Eqn. 2.9.2.8}$$

where,

$$\Delta u_{in} = \alpha (e_n - u_{in}) \qquad \text{Eqn. 2.9.2.9}$$

where α is the learning rate coefficient of the algorithm. The learning rate is initially relatively large and is decreased over training epochs such that :-

$$\alpha_t = \alpha_o (1 - t / T) \qquad \text{Eqn. 2.9.2.10}$$

where :-

α_t = learning rate at epoch t
α_o = initial learning rate
t = present epoch
T = total No of epochs

The size of the neighbourhood is also decreased over the training epochs in the same manner. Various functions can be utilised to define the shape of the neighbourhood and similarly to update the weight values in a negative (inhibitory) or positive mode such as a 'Mexican hat' function (Eberhart & Dobbins, 1990, p. 56).

In this way similar vector patterns are clustered within a multi-dimensional vector space or hyperspace. Depending upon the density function and the correlation of the input data, natural separations can occur. The degree of separation is measured in terms of error distances within hyperspace.

The number of training iterations required to reach an optimum degree of segmentation is still being researched. Ensuring an equal distribution of associated patterns across the output layer assists this optimisation. Selecting an appropriate learning rate and neighbourhood function are of fundamental importance and a recent adaption known as *'conscience'* has been employed. DeSieno (1988, pp. 117-124) recorded the frequency of winning neurons and applied a threshold such that if a winning neuron fired too often its weights were adjusted away from the input vector to achieve an inhibitory function. Similarly, non-firing neurons were biased to encourage participation. This method reduces redundancy within the output layer and has been particularly successful when dealing with sparse data sets.

By presenting already known classified vectors (if available), known as 'code book' vectors, into an acceptably trained network, winning neurons can be labelled as belonging to that class thus forming a map of definable regions within the two dimensional output layer.

SOFMs organise data in an non-predetermined way, and in some cases patterns will congregate and find associations with other patterns in unexpected ways. This method therefore requires a comprehensive understanding of the nature of the input data and necessitates the re-analysis of the clustered vector patterns.

The SOFM network has found substantial use within the field of speech recognition. Work undertaken by Kohonen utilises these maps to isolate Finnish speech phonemes via the interpretation of the FFT spectrum and Cepstral analysis of spoken words (Kohonen, 1988, pp. 11-22). In this way the words are broken down into the individual phonemes and charted as a directional path around the feature map, which is indicative of the original spoken word.

Sound interpretation has also been an application in medical diagnosis, where the FFT spectra of lung sounds have been classified to identify bronchial disorders (Kallio *et al*, 1991, pp. 803-808).

Many instances are to be found in the field of condition monitoring where vibration signals from plant are mapped and used to indicate faults such as bearing wear. In many cases, to solve such a problem necessitates the use of signals indicative of a faulty condition as part of the training set, a requirement often difficult to obtain. Research within the Neural Applications Group at Brunel University (Harris, 1993) and Helsinki University (Kasslin *et al*, 1992, pp. 1531-1534), have used feature maps trained on faultless data and exploited the inherent error vectors as either a pre-calculated fault identifier or as a warning threshold.

A derivation of this method, again utilising a competitive learning scheme, is that of Linear Vector Quantisation (LVQ). Although competitive, it is not strictly unsupervised as the classes within the data are predetermined. A winning neuron is selected and its neighbours are attracted or repelled depending on its class membership. In this way the classes separate and migrate to areas within the output layer. Finally, in its classification or test mode, a test observation adopts the class of the winning neuron (NeuralWare, 1993, NC-227-233).

Hybrid Networks

The features of certain network training algorithms and recognition capabilities have been combined to form hybrid networks which exploit the advantages and idiosyncracies in a complimentary network. Two such relatively common hybrids which have found success in applications are the Counter Propagation Network (CPN) (Hecht-Nielson, 1991, pp. 147-155) and the Radial Basis Function Network (RBF) (Leonard & Kramer, 1991).

CPNs are typically of three layers including the input layer. The hidden layer utilises a competitive learning schedule whilst the output layer requires the comparison to a target value for weight adjustment and is hence, overall, a supervised learning model. During training only the winning node in the competitive layer becomes active, thereby being solely responsible for the activation of the output neurons. CPNs offer a faster training alternative to the Back Propagation Networks, however, several critical pitfalls have been encountered. The technique requires the desired output classifications to be non overlapping and the training data to be uniformly representative of each output class. Secondly, instabilities can occur within the competitive layer if too few neurons are used for the resolution necessary to represent each output class. (Dayhoff, 1991, p. 215)

RBF networks utilise the same three layer learning technique as the Counter Propagation network. The training, however, is in three distinct phases.

1. The hidden layer employs a self organising feature map learning algorithm together with a neighbourhood function. The trained layer represents the frequency distribution of the input data patterns.

2. The two nearest neighbouring nodes are identified for each node in the input layer. The average Euclidean distance of the two nearest neighbours becomes the radius of each node's transfer function (which is generally Gaussian). Each node then produces an output depending on its distance from the input pattern and the cluster of nodes around it.

3. Each RBF node's response then becomes the input to the output layer which is subsequently trained using the backpropagation strategy.

(Hertz *et al*, 1991, pp. 248-250).

The hybrid networks described here are aimed at highlighting the subtleties encountered in neural network design and application.

The recent proliferation in ANN research has resulted in a plethora of learning algorithms, architectures and application strategies. In most cases the architectures and learning algorithms employed adopt the fundamental approaches described in this section. Some of the more recent advances attempt to replicate biological systems in a more thorough way. Such examples include *Spatio Temporal Networks* (STN) (Hecht-Nielson, 1991, pp. 164-192) which are tailored to deal with the classification of patterns associated with time progressions. The technique adopts an approach based upon the *Adaptive Resonance Theory* (ART) of Carpenter & Grossberg (1988, pp. 77-88). STNs have been used to classify the acoustic emissions generated by the cavitation effects of ship propellers. The signals acquired exhibited a very noticeable repetitive nature. This cyclic feature (a temporal progression feature) provided the segmentation necessary for a successful classification.

For details of these networks and a multitude of other paradigms and learning mechanisms the reader is directed to the given references.

In conclusion, it is evident that no definitive guidelines exist for the application of neural network paradigms. Many authors have suggested methods for building an ANN topology for specific applications but little, if any, mathematical or scientific evidence is given to justify these methods which as a consequence are termed 'rules of thumb'. The general aim, to reach the optimum architecture, is based mainly on the computational requirements of speed and memory space.

There is, however, a general concensus that the selection of the correct method to suit the application is critical to the successful implementation of ANNs.

2.9.3 Data Validation and Network Qualification

The application of neural networks requires the processing of in-coming data to provide a generalised relationship, typically by forming a non-linear transfer function between the input and derived output. The input data pre-processing, manipulation and post processing of network outputs forms a significant and important part of the neural network design and implementation.

The pre-processing of data prior to neural network training and testing is dependent upon the structure and learning algorithm of the chosen model and the nature of the data. General preprocessing consists of normalising the input vector and the randomly initiated weight vectors. The normalisation can either scale the vector components between 0 and 1 or, depending on the importance of the vector component magnitudes, ensure that the total vector magnitude equates to unity. Various methods are used including simple vector scaling and vector augmentation. (Eberhart & Dobbins, 1990, pp. 53-54 : Dayhoff, 1991, pp. 198-200)

The data associated with establishing a network requires a selected data set for three key stages :-

1. 'gold standard' representative data for network training
2. 'gold standard' data for network testing
3. validation data set for operational context testing.

(Ferret, 1993, pp. 193-204)

In the case of supervised learning methods these data sets must consist of both input and targeted output vectors or classifications.

Defining a representative data set for training and testing a network is a subjective decision. As in the case of expert system data base compiling, definitive decisions to a complex question can be a matter of opinion. The conclusions to research in the area of data selection tend to agree, however, that the performance of a network is dependent upon the quality of the training data, as is a human on the quality of its teacher. Representative data is generally accepted as containing the full scale range of a particular class or classes. Furthermore, data of inconclusive classification, that is considered borderline, is also used to assist in establishing boundaries or decision surfaces between other classes.

The overall quantity of data necessary to statistically certify a training/recognition regime is also of high importance. A first impression might point to the use of as much data as is

possible, in fact an infinite amount. This may be true of the test or validation set to enable a statistical affirmation of the results from the recognition stage. For training, however, problems are encountered. Some feature based mapping networks, back-propagation methods especially, suffer from the phenomenon of *overtraining* (Hecht-Nielson, 1991, p. 116). In these cases the networks appear to saturate and lose their generalising capability. Because of its near perfect association with the training set a poor classification of a previously unseen test observation maybe encountered as illustrated in figures 2.30a - b.

The amount of data necessary to achieve a predetermined ambiguity rate within statistical pattern recognition has been addressed by Cover (1965, pp. 326-334). As well as discussing the division of bi-variate data classes within hyperspace and the mathematical definitions for hyperplane development, Cover suggests an approach to sample sizes. The work formulates that for high dimensional data of two classes and a small probability of classification error:-

$$N_{min} > d / 2\,p(e) \qquad \text{Eqn. 2.9.3.1}$$

Where:-

N_{min}	= minimum number of samples
d	= number of vector components in feature vector
$p\,(e)$	= the acceptable probability of error in classification

It can be seen from equation 2.9.3.1 that for a low probability of error the minimum number of observations required can become very large. Furthermore, it shows that a perfect classification rate of 100% is not achievable because as $p(e) \rightarrow 0$, $N_{min} \rightarrow \infty$. This is, obviously, a statistical rather than a structural train of thought. In the same manner Nadler & Smith (1992, pp. 329-330) have provided a reciprocal equation for defining the size of a test sample set to prove a predetermined ambiguity rate independent of the dimensionality d of the data. This theory postulates that for an arbitrary test set N_s a desired error rate can be estimated. Secondly, a statistical confidence interval can be estimated which reflects a degree of confidence that the estimated error rate is within the actual error rate interval (given an infinite test set). Nadler & Smith conclude, although without rigorous mathematical proof, that for a confidence coefficient of 0.9 (90%) and for a required sample size where 10 erroneous observations are tolerable, that:-

$$\frac{10}{3N_s} < p\,(e) < \frac{30}{N_s} \qquad \text{Eqn. 2.9.3.2}$$

Which provides a lower N_{min} as derived by Cover when $d >= 10$.

Although noted within a statistical pattern recognition context, these theories may not hold true for ANN training and testing. In the case of backpropagation networks, non-convergence during training may be the result of too large a training set. Hecht-Nielson (1991, p. 113) states:-

> *"One of the ways of deciding how large a test set must be used is to (if possible) try out progressively larger test sets until the value of the mean squared error starts to converge to a fixed value."*

Another problem regularly encountered is the visualisation and hence cluster assessment of **high dimensional** data (greater than three dimensions). This results in difficulties in defining decision boundaries between clustered classes. Nadler and Smith (1992, p. 297) in discussing the importance of clustering 'real world' high dimensional data for pattern recognition quote Dubes and Jain (1979, pp. 247-260) :-

> "..".....*There is simply no way to accurately represent data in two dimensions unless the data have peculiar properties. Abandoning the two dimensional representation leaves the user with the difficult task of interpreting tables of numbers and creating an image from them.*"..."

Nadler and Smith conclude:-

> "*The two-dimensional diagrams we use, and will continue to use, are purely conceptual, intended merely to illustrate the various concepts we present. They are of very little utility in real problems.*" (italics in original).

Self Organising Maps have been applied to ease this dilemma by labelling trained output nodes via codebook vectors, however, they still negate two dimensional visualisation. Pal and Bezdeck (1993, pp. 2441-2447), have presented a method of visual assessment of clustering tendency for four dimensional data within Self Organising Feature Maps. The extension of the Kohonen algorithm enables VDU pixel alignment and gray scale encoding to emphasise the clustering.

Discussions and research concerning the correct data with which to train and test networks, and how to validate the results, is prolific but definitively inconclusive. The relatively new field of neural network applications is currently without a well defined development strategy. Ferret (1993, p. 195), states the advantages of such a

methodology as being :-

1. an increase in the probability of reaching an optimal solution
2. a reduction in the project development time
3. an acceleration in the acceptance of the technology
4. an increase in the reusability of the acquired knowledge
5. a more thorough understanding of the capabilities and limitations of the technology.

The lack of such a methodology and, as yet, unestablished methods of data validation have resulted in the reluctance to accept the application of the technology in safety critical systems. Who has the confidence to be a passenger in an aircraft being taught or, possibly worse, teaching itself to fly ?

2.9.3.1 Performance Metrics

Performance metrics for networks are not well documented, however, standard statistical testing is generally adopted.

Simple techniques such as a 'percentage correct' measure can be misleading depending on the distribution of the data set. The selection of the test set requires to be fully representative of the target process. This requires a prior data classification which has the inherent subjectiveness problems of who selects the 'gold standard' and by which criteria. Furthermore, the test set requires equal representations of each class regardless of the probability distribution of the classes (Eberhart & Dobbins, 1990, p. 164). Other methods commonly used instead of the 'percent correct' include:-

Averaged Sum Squared Error.
This method is primarily the error calculating algorithm used in the backpropagation model and is defined by:-

$$E_t = 0.5 \sum_{l} \sum_{p} (T_{pl} - O_{pl})^2 \qquad \text{Eqn. 2.9.3.1.1}$$

where,

E_t = Total Error
T_{pl} = Target Value for Neuron
O_{pl} = Measured Value of Neuron

This system provides the error for ALL output neurons and provides a comparative performance test for networks of equal size.

Normalised Error.
This error metric is calculated by:-

$$E_{norm} = \frac{E_t}{E_{mean}} \qquad \text{Eqn. 2.9.3.1.2}$$

where,

$$E_{mean} = 0.5 \sum_{l} \sum_{p} (T_{pl} - \mu_{pl})^2 \qquad \text{Eqn. 2.9.3.1.3}$$

and μ is the average target value for a given neuron. This method reflects a network's output variance as a consequence of the error independently of the architecture (Eberhart & Dobbins, 1990, p. 167).

Receiver Operating Characteristic (ROC) Curves.

ROC techniques can be used to measure such parameters as precision , sensitivity and false alarm rates. The method utilises a direct comparison of neural classification with the gold standard classification in terms of Boolean decisions. The ANN result can be considered either:-

a) True Positive (TP) where both network and gold standard are in agreement.
b) False Positive (FP) where the ANN made a positive classification not included in the gold standard.
c) False Negative (FN) where the ANN failed to concur with the classification of the gold standard.
d) True Negative (TN) where both ANN and gold standard agree on the absence of a classification.

Ratios of these measurements are used to yield the following criteria:-

TP/ (TP + FN) is the *Sensitivity*
TN/ (TN + FN) is the *Specificity*
TP / (TP + FP) is the *Positive Predictive Value*
FP / (FP + TN) = 1 - TN / (TN + FN) is the *False Alarm Rate*

(Eberhart & Dobbins, 1990, pp. 169-174)

Chi-Square Test.

This statistical test is well documented in any text dealing with hypothesis testing. It is based upon analysing the frequency distribution of actual output values against an expected distribution. It is designed to reveal a measurement of a variance greater than what would be achievable by chance alone. As such it is used as a test for a 'goodness of fit' (Owen & Jones, 1992, pp. 386-407).

At present neural network performance can only be assessed by a cross validation and correlation analysis. This necessitates a thorough knowledge of the data, the targeted application and an understanding of the operation of the network itself.

For welding applications this means an understanding of the interaction of the process variables, the correlation of network outputs with monitored data and post weld inspection.

2.9.4 Software Simulation and PC based Systems

The advantages offered by neural networks in signal processing and pattern recognition can be considered as mainly threefold :-

1. a robust generalising capability
2. once trained, speed of operation
3. adaptability.

These characteristics are inherent of the architecture, the parallel connection and interaction of individual processing elements and the adopted mathematical training scheme.

The high speed operation, a fast reaction time, is always a desirable commodity but not always a **technical** necessity. The response time in a real time control or monitoring scenario requires more consideration than a stock market predictor. The need for speed is task dependent.

Depending upon the size, architecture and training algorithm of a neural network the process is inherently computationally intensive during the training phase, but significantly less so during recall. This has resulted, however, in the ongoing development of hardware implementable networks where the main driving aim is speed optimisation. Other advantages include size, power consumption and operational reliability. Even though these advantages are evident there is a distinct commercial absence of dedicated ANN devices.

The parallel nature of ANNs means they lend themselves well to VLSI implementation, but once manufactured they are generally inflexible to modifications in algorithms, as they become available. The design and manufacture of hardware devices is a time consuming and consequently costly process and this, coupled with the immaturity of ANN technology, has limited their commercial viability (Garth, 1992, pp. 1397-1403). Even so, research is current in the area of specific network model implementations including Self Organising Feature Maps (Goser, 1991, pp. 703-708) and feed forward networks (Tomberg & Kaski, 1991, pp. 1561-1564).

The proliferation of PC manufacture has resulted in increased operating speeds. The serial nature of the data transfer has consequently become less of an inhibitory parameter in ANN simulation. The simulation of large networks requires relatively large memory storage and the technique used to manage memory allocation and access has a substantial effect on training times. In general, low level languages are used which allow

dynamic storage operations and, as a consequence, the ultimate limit on network size is the processor speed (Eberhart & Dobbins, 1990, p. 103).

There is an increasing trend amongst ANN theorists for the use of UNIX operating systems. Although comprising a much smaller instruction set than the more popular Disk Operating System (DOS), Unix lends itself well to the 'C' programming language and does not suffer from the memory segmentation problems associated with DOS. This system has, consequently, been used for super-computer applications and in the development of large scale networks. Unix is generally operated within a networked environment and therefore less readily available than PC based DOS systems.

It can therefore be concluded that, as well as presenting a low cost / performance ratio, modern PC systems offer a flexible experimental platform and a valid option for the solution of ANN applications. Careful consideration has to be given, however, to the memory problems associated with large scale networks and the time dependency of the problem.

2.9.4.1 Hardware Platforms

Hardware platforms, in general, process specific instruction sets and data transmission protocols in terms of I/O. Standardised I/O systems are essential for optimised communication to and from peripherals such as monitoring transducers and user interface systems. The speed of operation of ANNs can be negated by the speed of communications between the application and the hardware platform used. It is necessary, therefore, to establish an appropriate communication protocol and hardware platform in which to deploy the network for time dependent, real-time operating scenarios.

The accelerating development of PC systems, such as Pentium® processors, has substantially increased the speed capability of real-time systems, whilst retaining an almost universal compatibility with a plethora of peripherals. However, the external communications protocols remain standard and often create data bottlenecks.

The use of transputers has offered an alternative to PC simulations. The ability of these machines to be connected and operated in parallel, together with their multi-tasking capabilities, offer a powerful tool in neural computing (Murre, 1991, pp 1537-1540). The kernel is, however, software driven and therefore retains flexibility, but the incurred financial outlay can be far in excess of the PC based option.

The application of neural networks entails a substantial amount of signal preprocessing to ensure network compatibility. The computational time for network recall is small in comparison to the necessary on-line DSP. Recall operations of ANNs can be slowed considerably by the communication protocols between the ANN development platform and the DSP hardware.

The TMS series CPUs are tailored towards signal processing applications which rely on floating point and Multiply Accumulate Cycle (MAC) computation at high speed (Texas Instruments, 1994). Executing a neural network in recall is usually very calculation intensive. In the case of backpropagation methods each node requires the repetitive calculation of an inner product term to establish the nodal value or weight. The TMS320C32 is capable of a MAC rate of 20MHz which enables very fast real time recall and has been used to execute ANN functions in welding applications (Swallow, 1995).

The increasing possibilities for the use of flexible, rather than dedicated VLSI hardware, is highlighted by forthcoming DSP systems. At the time of writing, Texas Instruments have introduced TImeline platforms. Utilising 1.8-micron silicon technology, these systems incorporate 125 million transistors (the equivalent of 30 Pentium® processors) per device and are capable of a clock speed of 500 MHz or more (Texas Instruments, 1996, pp. 1-2).

The increasing power of DSP systems, together with software flexibility, offers high potential to the development of neural networks and their commercial application in real-time monitoring and control systems.

2.10 Neural Networks and Welding

The physics and metallurgical aspects of welding technology have become established over the last century, with automation playing a major part in its development during the past fifty years. The optimisation of automated welding processes has been the focus of much research with the main aim of developing a relatively low cost integrated control system to enhance productivity, whilst maintaining quality.

The cycle of producing a good quality weld can be simplified into three key stages (as illustrated in figure 2.31):-

1. Pre-weld parameter setting :

This area is a subjective one and has become an area in which expert systems are prolific, endeavouring to cover all application areas by means of statistical modelling.

2. On-line monitoring and control:

Classical methods in real time control engineering have been the mainstay in this area for many years, with the use of conventional transducers for measuring process parameters via the power source and analogue closed loop control. Recent developments in ultrasonic, video and computer technology have advanced the quality of control systems and have accelerated the discipline into the area of applied robotics.

3. Post -weld inspection:

To maintain quality control of weld production, techniques such as dye penetrants, X-ray and ultrasonic inspection have been developed. The automation of these techniques has yet to be fully accomplished. The identification of flaws from notoriously transient ultrasonic scans requires the interpretation and opinion of a trained human operator (Harris & Wilkinson, 1993, pp. 549-554).

The recent advent of neural networks as a computational technique has provided another tool with which to advance the quality of weld production. Although a comparatively immature technology, the exploitation of the inherent characteristics of ANNs is already evident in all of the areas associated with weld production mentioned.

A recent survey by Shaw & Lucas (1994), highlights the significance of the use of neural networks within the welding field. The paper emphasises the usefulness of the generalising and auto-associative properties of networks and their application in

deriving the non-linear relationships between the many interdependent parameters involved in welding processes. The work concludes that the technology has not, as yet, proven itself and describes the performance and accuracy of already tried networks in weld modelling as "*generally satisfactory*".

Although the same paper expresses a preference towards conventional computing methods for mathematical modelling, various claims to the successful application of ANNs in this area contradict this. ANN techniques have been applied to weld quality prediction in the USA. Work undertaken by the American Welding Institute is described by Jones & Evans (1992, pp. 101-108). A commercial system known as WELDBEAD™ has been developed which models the Gas Tungsten Arc Welding (GTAW) and Flux Cored Arc Welding (FCAW) processes. The neural network kernel receives information on current, voltage and travel speed with the aim of predicting such quality criteria as bead geometry, amount of spatter, penetration and arc stability for various joint configurations. A reversal procedure is also claimed for determining voltage , current and travel speed. A neural based prediction system has also been investigated in the UK by Li *et al* (1994), for the Tungsten Inert Gas (TIG) process, where the same input parameters are used to predict bead height and width. Martin & Bahrani (1993, pp. 89-95), conclude that the ANN approach is faster, more accurate and more efficient than previously tested expert systems.

The operational speed, adaptability and generalising robustness of neural networks, make them an attractive proposition for real time monitoring and control applications. Welding processes yield inherently noise corrupted data. Monitored electrical and acoustic signals are transient and video images are subject to erratic light levels and smoke contamination unless adequately illuminated and filtered. Positional control provides a specific problem in welding robotics. Dilthey *et al* (1992, pp. 147-155), focused on the advantages of ANNs in real time signal processing by developing a welding head positional control system for the Metal Inert Gas (MIG) process. Changes in joint geometry of a fillet weld during welding were reflected in the monitored electrical signals. A feed forward network was trained to compensate for the change in joint geometry by re-positioning the welding head in real time. Over a 30 cm weld length, lateral deviations of up to 5 cm were successfully compensated for.

Acoustic emission analysis has also been investigated for real time monitoring purposes. Research undertaken by Matteson *et al* (1992), utilised a 3 hidden layer feed forward network with a single output neuron to classify acceptable or non-acceptable welds produced during the Gas Metal Arc Welding (GMAW) process. The network used the backpropagation learning algorithm to classify the welds via FFT interpretation.

Claims are made for up to a 90% correct classification rate. However, the validation procedure adopted incorporated a thresholding post processing stage and a percentage correct metric.

An on-line neural base monitoring facility for automated rotor welding processes has been subject to a US patent by Bellows *et al* (1994). Although the system provides no feedback for closed loop control, signals to the network are obtained from a CCD camera and audio spectrum analyser as well as electrical and gas process parameters. The neural network is trained on normal running conditions and designed to output an audible and visual alarm state, with a fault diagnosis, when an abnormal running condition is encountered.

In the area of post weld inspection and NDT, neural networks are being applied to the interpretation of ultrasonic scans to classify weld flaws. Harris & Wilkinson (1994), are utilising the benefits of neural networks for an on-line weld flaw detector and classifier with an aim to develop a commercially viable aid for NDT inspectors to use on site.

It is evident that neural network application is providing a useful tool in the major areas of weld production. New innovations are becoming more apparent in neural network development and welding technology provides experimental avenues in which to implement them. The general trend in the acceptance of this new technology is the development of so called 'hybrid intelligent systems' (Siores, 1992) where already tried and established expert systems are used in conjunction with a neural capability to form an optimum solution.

With the continuing goal of optimising weld quality, improving general working conditions and increasing productivity, it can be concluded that welding technology will benefit from the application of artificial neural networks.

2.11 SUMMARY

It is evident that audible acoustics play an important role in the manual monitoring of welding processes, however, visual information and process knowledge are also fundamental. Acoustic emissions are a consequence of the process and although almost certainly information rich, past research techniques have proven the media unreliable.

Investigation of the acoustic emissions from Submerged Arc Welding has received little attention compared with more widely used techniques such as Metal Inert Gas Welding. Even for acoustic research within this process, there is no clinical proof as to what time/ amplitude or domain functions are indicative of the process state. Most investigations of the arc sound have been based upon the direct mathematical correlations with the already known process variables of arc voltage and current.

Establishing a cybernetic approach by replicating the functions on the inner ear is limited by the present understanding of the human auditory system. Although the biological mechanics of the ear can be readily modelled in terms of commercially available transducers, the neuronal transmission of information and the cognitive interpretation of sound can only, at present, be hypothesised.

Of the tools available to the AI protagonist, artificial neural networks are claimed to be the most biologically plausible. Offering an adaptive and noise tolerant approach to complex pattern recognition, they appear an attractive alternative to established statistical methods.

The problems associated with the interpretation of ANN performance necessitates a thorough understanding of the targeted process to enable cross validation. Furthermore, with the plethora of network architectures and learning / recall algorithms that have been and are presently being developed, the selection of the correct method is crucial in defining an optimum solution.

The selection is compounded by the lack of definitive or standard guidelines for applications. The intended user is faced with mathematical descriptions of network architectures applied in similar and dissimilar contexts with which to make comparisons to construct a solution.

The necessity of high speed operation for real-time control tasks is of fundamental importance. With the rapid development of DSP hardware and PC platforms, reaction times for specific on-line applications are more readily achievable. Consequently, the established time constraints associated with an expert system approach are becoming less

problematic. However, adaptivity and robustness are still a necessary requirement.

The application of ANNs is a relatively new technology and as such is receiving increasing attention within the welding fraternity. However, their acceptance as a viable solution is negated by the lack of commercial use in proven applications.

The information presented in this review illustrates the multi-disciplinary approach necessary to apply ANN technology to specific analytical and control tasks. The time dedicated to the neural network development and application is often outweighed by the learning curve necessary to establish an understanding of the problem and preparation of suitable data. The availability of such data can also present an obstacle to the successful implimentation of ANNs. The acquisition of consistent data with which to train, test and statistically validate a neural system can involve time consuming (and often tedious) experimentation if the information is not readily available.

3.0 EXPERIMENTAL METHOD

3.0 EXPERIMENTAL METHOD

3.1 Welding Specifications

Previous work carried out within the research group included the study of weld penetration measurement and seam tracking by means of ultrasonic transducers (Stroud, 1982 : Harris, 1992). The weld method chosen for this work was Submerged Arc. The relatively thick parent plate enabled easier penetration monitoring, as well as providing a comparatively clean and safe laboratory process.

As a consequence the welding rig was readily available for experimental work. Furthermore, there was little evidence as to the study of acoustic emissions from SAW as opposed to other common welding methods such as MIG. It was therefore considered that Submerged Arc Welding would be the target process.

3.1.1 Set-Up

The test rig, as illustrated in figure 3.1, consisted of a submerged arc welding head mounted on a carriage which possessed three degrees of movement. The head was able to be traversed along a 2.75 metre extruded aluminium beam by means of a d.c. motor drive with optional tachometer feedback. The motion could either be controlled through manual controls at the welding head or by remote pre-setting of the direction and speed via a computer interface. The maximum welding speed obtainable was 2m/min with an accuracy of +/- 1%. Limit switches appropriately positioned at the extremities of the beam provided the safety cut-off mechanism.

Lateral movement at right angles to the weld direction was controlled directly via the computer interface with no manual override. The in/out slide movement was driven by a stepper motor mounted directly on the end of the slide which gave approximately 100mm of motion. This range equated to a control pulse from the computer providing 0.085mm of lateral movement. For the experimental work carried out within this thesis. control of this axis was not considered important and only used for pre-set weld alignment.

The vertical positioning of the welding head was by means of a manually operated slide 60mm long. The movement of this axis was not motorised. As a consequence on-line stick-out / stand off control was not possible. For the complete duration of the experimental work stand off was fixed at 25mm.

The welding head itself comprised a 'Devimatic 5 - standard head' capable of running single filler wires of diameters 0.8mm - 5mm or twin wires up to 1.6mm. The wire feed was controlled via a d.c. permanent magnet motor. This was regulated via the welding installation control box (figure 3.2), manual override or the computer interface. This enabled both pre-set constant feed rate or on-line remote adjustment. Wire feed speed was in the range of 0.6m/min - 6.5m/min.

The aluminium weld nozzle incorporated a manual valve which was connected via a rubber hose to the flux hopper containing approximately 11kg of dry powder flux. This was gravity fed to the weld area.

3.1.2 Power Source

The power source used was a 'SAF Starmatic 800TH' as illustrated in figure 3.3. This was a constant-potential current rectifier transformer specifically designed for automatic (synergic mode) or manual preset welding installations using consumable wire methods.

The characteristics are typical of a submerged arc power source with a voltage range of 16.5V - 56.5V and current ranges of 80A @ 16.5V and 1000A (after modification) @ 56.5V.

In principle the control operation monitors the arc voltage, as sampled by the setting potentiometer, which is compared with a stabilised reference voltage. Any deviation between the two voltages is amplified and imposed on the control circuit in the desired direction to cancel the arc voltage deviation.

The result is a relatively stable arc where voltage variations of +/- 10% do not have any adverse influence on the weld profile.

3.1.3 Consumable (Filler Material)

All the welding experimentation carried out utilised commercial copper coated alloyed steel wire (1.3% Manganese / 0.06% Carbon) of diameter 3.2mm.

3.1.4 Parent Material

The parent material utilised was 25mm thick mild steel, medium carbon (0.4%), flame cut plate measuring 1m x 0.25m.

3.1.5 Shielding Medium

All welds were conducted under a general purpose granular flux of type 'Oerlikon OP 181' to standard DIN 32 522 - BAR 1 88 AC10 SKM. The depth of flux was maintained constant with the stand off at 25mm.

3.1.6 Weld Lengths and Type

The maximum weld length was limited by the traverse distance of the welding head carriage and the length of parent material. Initially weld lengths of 250mm were used for the feasibility study. This was mainly due to the original data acquisition facility utilising PC RAM space which enabled a maximum digital sound recording duration of 2.5 seconds. Further modification enabled experimental sound recordings to be sent direct to the hard drive which enabled longer recordings which were limited only by the memory capacity of the hard disk. Maximum weld lengths were consequently set at 800mm. Typical weld runs for the feasibility tests are shown in figure 3.4 with the modified system welds of full length given in figure 3.5.

The high permutation of possible joint configurations and weld types meant that a specific aspect had to be adopted for study. The general thrust of the work was to research the audible noise activity associated with arc stability and consequently electrical process parameters. Improper parameters together with adverse welding conditions, such as insufficient flux coverage and contaminated plate, can affect bead profile and inflict defects. These criteria were considered readily obtainable from bead on plate welds which negated the need for costly and time consuming plate edge preparation.

3.1.7 The Ultravis Control System

The weld control system consisted of either the power source synergic mode, manually pre-set values via the power source and weld installation control box (figure 3.2) or from a PC hosted software program known as Ultravis. The Ultravis controlling software was devised as part of an ultrasonic seam tracking system by Harris (1992).

In general the weld control is open loop. The ultravis software communicates through a serial link and multiplex arrangement housed in a suitably electrically shielded cabinet (fig. 3.6) to a custom designed interface control box (fig. 3.7). The interfacing electrically isolates the computer hardware from the weld manipulation hardware with optical coupling and/or relay circuitry.

User defined weld speed, voltage and current are set with the PC and is transmitted to the kinetic actuators and power source controller at the start of the weld cycle. The Ultravis system was specifically designed for ultrasonic monitoring of the weld bead and seam tracking and therefore provided no process variable monitoring and/or data logging.

3.1.8 Initial Weld Runs (Bench Test)

Initial short weld runs were conducted (200mm length) under varying voltage and current levels. The resulting weld beads were cross sectioned and measured in an attempt to establish weld stability via consistent weld bead profiles. Typical weld bead profiles are illustrated in figures 3.8 a-e (showing changes in weld current) and figures 3.9 a-d (showing changes in weld voltage). The bead profiles obtained remained consistent with accepted observations and weld conditions as defined by Houldcroft (1989, pp. 28-29), and were used as a dimensional reference for subsequent weld runs.

3.2 Transducer Selection

The characteristics necessary to monitor arc sound are satisfied by the condenser microphone. Electret condenser microphones (ECM) are used in high quality sound meters. They operate by means of a vibrating diaphragm which acts as one plate of a capacitor. The changing distance between the diaphragm and the opposing fixed plate changes the charge held by the capacitor. Typical characteristics of these transducers are,

1. an excellent frequency response between 2.5 Hz - 140 kHz
2. a uniform frequency response
3. a good signal to noise ratio
4. a toleration of various ambient conditions such as heat and humidity.

However, sensitive amplifiers are required in close proximity to the microphone due to the small signals generated. It is also evident that a high magnetic flux can exist in the vicinity of the arc. Certain microphones, such as dynamic types which utilise core magnets, could be influenced by such an ambient flux resulting in spurious signals. Electret microphones were believed to be less likely to exhibit this characteristic.

Although the main aim of the experimentation was to develop a monitoring system based upon audible sound, investigative consideration was also given to the possible influence of ultra and infra sound ranges. As a result three omni-directional ECMs were chosen each with frequency response characteristics suitable for the intended sound range. The chosen transducers were Jin In Electronics ECM - 10 (0-15Hz), ECM - 30 (15Hz-20kHz and ECM - 66 (20kHz - 40kHz), for manufacturers specification refer to Appendix 1.

3.2.1 Pre-Amplification

ECMs require a power supply as well as suitable amplification for the otherwise small signals generated. The circuit schematic for the tri-transducer system amplification is shown in figure 3.10. The amplifier I.Cs (IC1, IC2 & IC3, fig. 3.10) chosen were type AD OP 278N. These bi-polar operational amplifiers exhibit a high signal to noise ratio and are an industrial standard high precision component recommended for use as microphone and phono pre-amplifiers. The arrangement detailed allows a maximum 20dB gain.

The final printed circuit board was mounted within a galvanised steel box and electrically earthed via the welding head saddle. The ECMs were connected via twisted pairs to the amplifier PCB. The final amplifier arrangement is illustrated in figure 3.11.

3.2.2 Positioning

The three ECMs were mounted in rubber insulating sleeves and situated in a custom manufactured microphone head as illustrated in figure 3.12. Further shielding from the heat and fumes experienced within the vicinity of the arc was provided by an aluminium cover plate shown in figure 3.13. The complete microphone head was electrically earthed and mounted at the end of a 200mm flexible arm to enable multiple positioning.

The complete microphone assembly and pre-amplifier unit was mounted on the welding head so that the transducer arrangement remained aimed at the welding arc at a distance of 300mm. The full arrangement in situ of the welding head is shown in figure 3.14.

On completion of the microphone assembly, signal amplitude levels were established by performing and monitoring sample welds with an oscilloscope. Adjustments were made to the gain controlling variable resistor so that the maximum voltage swing exhibited by the oscilloscope was 6V (+/- 3V amplitude). This level was selected so as to be compatible with the ADC arrangement present on the DSP board described in section 3.4.1. Further tests were instigated as part of the frequency response test of the filter set, described in section 3.3.1.

3.3 Design & Development of Active Filters

As previously described in section 2.6.1, the hearing mechanisms within the cochlea are a result of the critical bands along the basilar membrane. These bands denote areas of cilia activation dependent upon the frequency content of the applied signal and can be thought of as a series of bandpass filters.

In an attempt to replicate this a filter bank was designed and constructed. The design was such that a series of frequency bandwidths could be isolated or combined, that is utilised as pass bands or stop bands. Waveforms could then be produced devoid of unwanted noise or irrelevant information. The intended frequency bandwidths for the filter bank is schematically illustrated in figure 3.15. Human hearing deteriorates towards the extremes of the audible spectrum as given in the human hearing sensitivity audiogram in figure 2.9. Such evidence suggests that manual welders would be more likely to extract information from the 1kHz - 12kHz range. As a consequence the series of designed filter passbands narrowed in this region to assist in salient frequency isolation.

The circuit schematic for the filter set is given in figure 3.16. The amplifying components (ICs 8-12) are quad low noise bi-polar operational amplifiers type OP-470GP. These devices were chosen for their low noise voltage susceptibility, excellent gain accuracy and ability for accurate amplifier matching. As such these devices are specifically recommended by the manufacturer for use as accurate low noise active filters.

Each filter consists of a cascaded pair of op-amps which increased the gradient of the roll-off characteristic and hence accuracy of the filter bandwidth. The RC components were calculated to obtain as near the desired upper and lower break frequencies as possible, with a unity gain to maintain the 3V amplitude required by the ADC. The RC components used were the closest commercially available values calculated from the following equations:-

$$\text{Lower Break Frequency (LBF)} = \frac{1}{2\pi C_1 R_1} \qquad \text{Eqn. - 3.3.1}$$

$$\text{Upper Break Frequency (UBF)} = \frac{1}{2\pi C_F R_F} \qquad \text{Eqn. - 3.3.2}$$

The completed filter box is illustrated in figure 3.17 with the internal components shown in figure 3.18.

The power source for all components was via an integral power supply unit, mains driven, which was specifically designed and constructed for this task. The circuit schematic is given in figure 3.19.

3.3.1 Bandpass Filter Bank Frequency Response

On completion of the design both the microphone assembly/pre-amplifier and filter system were subjected to a frequency response test.

An anechoic chamber was constructed from medium density fibre board (MDF) specifically to test the ECM transducers. The chamber consisted of two halves separated by a rubber seal. When clamped together with the ECM inside, the box provided a sealed sound damping system. The face of the chamber opposing the receiver side of the transducer contained an aperture through which was mounted an audio speaker. The speaker was a flat faced 8 ohm impedance type with a frequency response range of 30Hz - 25kHz. The chamber is illustrated in figure 3.20.

The speaker was in turn driven by an LM386 audio power amplifier system. The input to the amplifier was supplied by a 'Feedback F6601' function generator Serial No. 6013089 (figure 3.21).

A dual beam oscilloscope was connected to the function generator output and the output of the filter system. Individual band pass filters were then tested on the range of frequencies with response and overall system gains given in table 3.1.

The physical limitations of the chamber made the frequency testing of the infra sound transducer and associated low pass filter difficult. Testing of the ultra sound transducer was limited due to the frequency response of the audio speaker. Consequently a more complete test was carried out by directly inputting an appropriate sine wave from the function generator to the relevant filter module.

Figure 3.22 shows the complete Bode diagram showing all the frequency responses for the filter series. The resulting overlapping pass bands were intentionally created to eliminate the possibility of stop band 'notches' occurring within a particular frequency range.

Figure 3.23 shows the associated phase lag of the complete system approximated from the separate filter tests.

3.4 Analogue Signal Conversion & Processing

The filtered analogue waveform was then sampled and digitally recorded via a PC hosted commercial DSP facility.

The DSP development board utilised a TMS320C30 processor with integral ADC and memory (although limited) as described in section 3.4.1.

3.4.1 DSP Specifications

The Texas Instruments family of TMS320C3* DSPs utilise 32 bit floating point multiple Harvard architecture as well as the more familiar von Neumann type. They have been specifically designed for real time DSP applications allowing complex and demanding computations such as FFT and digital filter algorithms to be executed.

The processor facilitates 16.7 million instructions per second for an instruction cycle time of 60 nanoseconds. A high throughput is achieved by a five level 'pipeline' or parallel operation implemented by the specific instruction set which also supports up to three operand instructions per cycle. A comprehensive description of the processor architecture is given by Papamichalis (1990, pp. 33-49).

The development system as supplied by Loughborough Sound Images (Serial Number TMSC320-597, figure 3.24) performed at a clock speed of 33MHz (33Mflops computation with parallel operations of the floating point multiplier and adder). Analogue signals with a bandwidth up to 75 kHz could be sampled directly to the board via one of its two 16 bit ADCs of up to 200kHz sample rate per channel. Resistor-Programmable anti-aliasing filters to a conventional Sallen-Key design were integral to the system board. The filters could be configured as 4th order low pass Butterworth with a 24dB/octave roll-off with a unity gain pass band.

Two blocks of Random Access Memory (RAM) were available. The first, 16 kb x 32 bit of Static RAM (SRAM) was dual ported between the host PC and the TMS320C30 processor for user interface communication. The second, 16 kb x 32 bit of fast SRAM provided local memory for sole processor use.

In operation, the overall speed and power of the system allowed an execution of a 10th order complex FFT to be completed in the region of 3 milliseconds.

3.4.2 Host PC Specifications and Software

The use of the PC was fundamental to all the data acquisition, data storage and consequent experimentation and analysis. Furthermore, the DSP development board required the use of a PC host for user interface communication and programming. For the duration of the research certain activities necessitated the use of more than one type of computer. In some instances the computational requirements of a SUN workstation, utilising a UNIX platform, was necessary for complex high dimensional data analysis. For the main, however, the following PC format was adopted for on-line data acquisition and neural network assessment (figure 3.25).-

PC Manufacturer :-	Trigem 486/66F (on-line) & RM Ventra 466 (off-line analysis)
Processor :-	Intel i486DX2, 32 bit, 66MHz. (integrated math co-processor)
Bus Standard ::-	EISA
Memory :-	256kb cache RAM, 16Mb fast RAM
Hard Dive :-	c:\ 340Mb d:\ 1.2Gb (Trigem) & c:/ 1.2Gb (RM)
Monitor :-	SVGA

Some of the experimental analysis required the use of commercially available software in the form of DSP tools, assembler or higher level language compilers, data analysis tools and neural network development tools. The systems utilised for this experimentation were:-

1) Texas Instruments Optimising C - compiler
2) Texas Insrtruments TMS320C** code assembler - compiler
3) Hypersignal Workstation (version 2) - DSP & signal analysis tool
4) Microsoft C6 - compiler
5) Microsoft Visual C++ - compiler
6) Microsoft Excel - statistical analysis & spreadsheet package
7) XGOBI Dynamic Graphics - statistical data analysis tool
8) Neural Technologies NT5000T - ANN development tool
9) NeuralWorks ProII - ANN development tool
10) NeuralWorks Data Sculptor - data anlysis & pre-processing tool

In general the feasibility studies utilised commercial packages which enabled the flexibility and speed of development necessary to form a working solution. Further studies resulted in specific custom software routines to enable adaptability in experimentation.

3.5 Weld process Variable Monitoring

As discussed in section 3.1.7, the existing submerged arc control system (Ultravis) provided no means of feedback control. This in turn meant no monitored information being logged from the welding process.

To provide a valid statistical assessment of the weld scenario and also well labelled data records it was necessary to create such a facility. The interface control box (fig. 3.7), produced by 'Synergic Interactive Systems', provided a configured parallel computer interface which included analogue measurements of voltage, current and travel speed.

A PC compatible ADC arrangement was procured to enable real time digital data logging during welding.

3.5.1 ADC Specifications

The system utilised was the Bytronic MPIBM6 Multifunction I/O Card (fig. 3.26). The utility was commercially designed for educational and industrial use and provided digital I/O, ADC/DAC and counter/timer operations.

The ADC function was provided by an ADC0848, which is a CMOS 8 bit successive approximation type with accuracy to +/- least significant bit (LSB). The system could also be software configured via an 8 channel multiplexer for certain modes of operation including differential, pseudo-differential and single ended.

The ADC clock speed was 25 kHz with an average conversion time of 30 microseconds.

3.6 The Audio Recording System

Initial sound recordings were sampled via the TMS320C30 DSP board by using a PC menu driven user interface program called Hypersignal Workstation (Version 2.03, Serial No. 201864). This program enabled the full configuration of the DSP system including ADC sample rates and the application of suitable digital filters. The system was utilised to record, in digital format, the converted analogue signals generated by the audio filters. Off-line analysis and full editing of data was possible in the form of time/amplitude traces and frequency domain transforms. Further software manipulation enabled the configuration of the DSP for on-line FFT operation which, in turn, generated a real time PC visual display in either relative magnitude, logarithmic dB or power spectral format.

Although providing a very powerful analysis tool, the digital recording to the computer hard drive was limited. Depending upon the data acquisition sampling speed, and hence the sampled signal bandwidth, recorded time durations were limited due to the available run time RAM space.

3.6.1 Initial Recordings

It was initially conceived that the sound generated by the welding rig ancillary equipment could prove detrimental to the quality of the sound recorded from the weld arc area. The sound generated from the extractor fan and/or the power source cooling fan for example could possibly mask more salient audible features.

As a consequence a series of 'dry welds' were performed where no arc ignition occurred. The sound from all the ancillary equipment including the power source, fume extractor fan and kinetic actuators were recorded both in isolation and in combined operation. The transient capture sampling frequency was set to 40kHz and 100ksamples at 16 bit resolution. This enabled a total time domain data file of 2.5 seconds duration.

A library of 'associated' sound data was then created which was subsequently analysed off-line. Time/amplitude signals were windowed and subjected to a 10th order FFT. The full 2.5 second data files were analysed as magnitude and power spectral formats. It was envisaged that this data library could be used to establish the characteristic sounds of the welding rig which could then, if necessary, be de-convoluted from a weld recording providing an effective filtration operation. This would then highlight the bandwidths associated with the welding arc.

Typical time/amplitude traces with associated frequency transforms are illustrated in

figures 3.27 and 3.28.

3.6.2 Initial Weld Recordings

With no changes to the DSP configuration, initial weld runs were instigated with defined parameter settings. It was initially decided, as a signal analysis and feature detection feasibility, to isolate characteristics associated with changes in weld voltage.

With the travel speed and current held constant at 6.5mm/s and 860Amps respectively, weld runs up to the full experimental length of 800mm were instigated. Once the arc had been established and the weld stabilised, the voltage was increased until obvious instabilities occurred with excess spatter. The weld run was then repeated and the voltage reduced until the same instabilities occurred with eventual arc failure. With the acceptable voltage range for the corresponding travel speed and current established, further weld runs were instigated at these extremity levels and sound recordings taken.

Tables 3.2 - 3.4 show the actual weld parameter settings for these welds with corresponding measurements taken from the integral weld monitoring system. Just above the mid-range voltage level (40V) was, for the purpose of this experiment, taken as the optimum stable value as it provided an ideal bead profile and weld penetration for the thickness of parent plate . In general the weld settings adopted were :-

High Voltage	**50V**	542A +/- !0%	6.5mm/s
Mid-Range (optimum)	**40V**	560A +/- !0%	6.5mm/s
Low Voltage	**25V**	550A +/- !0%	6.5mm/s

Fresh, cold plate was used for each recorded weld run. It was initially found that excessive welding on the same material distorted the plate to the point of affecting the stand off distance.

Again the limitations of the Hypersignal Workstation software providing only a recorded file length of 2.5 seconds meant that only a short proportion (16.25mm) of the length of the weld was captured. At this stage the weld runs were repeated with the DSP software providing an on-line FFT and a video camera set up to monitor the computer display. Further analysis showed that there was no appreciative deterioration of the signal and that, in the main, the acquired data remained constant over the entire weld run.

The analysis of the acquired signals is described in section 3.7.

3.7 Signal Analysis

The acquired signal analysis was aimed at establishing the fundamental differences between the traces indicative of the associated voltage settings. Furthermore it would provide the opportunity to locate and ascertain the influence, if any, of the sound generated by the ancillary equipment.

3.7.1 Time / Amplitude

The signals acquired were intrinsically random i.e. they were non-deterministic. Such signals in the time domain can only be described in terms of average values such as r.m.s, mean amplitude values and statistical descriptors such as *cumulative-distribution-functions* (cdf) and *probability-density-functions* (pdf) (see Martin,1991, pp. 289-322). The latter functions, as with most statistical distributions, require a very large number of samples for any degree of accuracy. This would entail a large time window to be assessed, reducing the plausibility of real time action.

Typical traces for the time/amplitude signals for low, optimum and high weld voltages are illustrated in figures 3.29 - 3.31. The signals here are cued to enable an easier comparison. The amplitude modulations which are evident in all traces equated to a time displacement of 20 ms or 50Hz. Although not proven, it was believed that the mains frequency provided the modulation, and as a common feature provided no usable information.

The most obvious visual differences in the signals appeared to be the amplitudes, with the high voltage weld providing least disturbance, followed by the optimum weld and the most disturbance being created by the low voltage weld. Figure 3.32 shows the calculated correlation coefficients for the aligned time / amplitude traces. With coefficients calculated between -0.04 and +0.06, there was no distinct, noticeable correlation between the signals.

The amplitude differences are best illustrated by figure 3.33. This shows the relative r.m.s values for the raw time amplitude traces. Further correlations calculated are shown in figure 3.34 which indicate a factor of 10 improvement in correlation over the raw signals for the entire duration of the signal. There were, however, no distinctly obvious trends or dis-similar functions which could uniquely be defined as features.

Further comparisons with time amplitude traces from the welding ancillary equipment showed that the sound intensity of the ambient non-welding tests were masked by the actual welding noise. Therefore the consistent background sound of the transformer and

extractor fan etc., inflicted negligible influence on the recorded arc noise.

It was decided to analyse signals within the frequency domain for the following reasons :-

1. The time / amplitude signals provided no statistically unique features.

2. The transient nature of the signals would have made sample triggering / cueing very unreliable.

3. The low frequency modulation could more easily be ignored.

4. Real time statistical analysis of the time / amplitude signals with the necessary sample width and time window would have hindered on-line application.

5. Biological systems utilise a frequency / amplitude dependent method as does the FFT function.

3.7.3 Fast Fourier Transforms

The Hypersignal Workstation software was configured to perform an off-line 10th order (1024 point) FFT on the acquired time / amplitude signals. The resulting FFT magnitude plots for the optimum, low and high voltage values are given in figures 3.35 - 3.37.

The noticeable feature was the consistent appearance of a frequency content at 14.3 kHz, with a related harmonic at 11.42kHz, in all the acquired signals. Although sometimes masked within the low voltage trace by the sound intensity, it is still noticeable at the lower intensity levels. Further investigation revealed that the frequency peak was produced by the wire feed motor in operation during the welding cycle.

Most of the noise activity was seen to be exhibited in the 0-10kHz range. The low frequency 15Hz oscillation was also evident although at a much lower relative amplitude.

Changing either the amplitude or the frequency axes to a logarithmic dB or octave scaling respectively provided no additional information.

3.7.4 Auto & Cross Correlation Functions

Auto and cross correlations were instigated on the time / amplitude signals and again subjected to a 10th order FFT function. The auto-correlation, as described in section 2.8.3, is primarily an average measurement of the product of the signal and a time displaced version of itself, rendering a time averaged signal. As such a smoothed response can be obtained.

Similarly the cross-correlation provides the same function between two unrelated signals. Typical results of such functions are shown in figures 3.38 - 3.40. These figures show the FFT result as the affect on the time / amplitude signals is less obvious.

The smoothing functions highlight the areas of most interest, or otherwise, by emphasising the bandwidths in common and removing sporadic noise.

3.7.5 Time Variant *FFTs & Power Spectra*

The FFT functions exhibit a variation with time. The number of FFT frames generated within a certain time span is dependent upon the sampling frequency, the order and the window size of the FFT. Although the DSP enables a fast multiply accumulate cycle for computation of FFT frames, the communication to PC peripheral devices can slow the process. For the system used, approximately 100 FFT frames of 10th order were generated over the 2.5 second audio recording, making each frame representative of 25milliseconds.

Viewed in three dimensions of frequency, amplitude and time a contour is constructed as shown typically in figures 3.41- 3.43.

The Power Spectra was calculated by squaring the amplitude and averaging the FFT frames over time. The result was a smoothed less transient FFT trace as illustrated in figure 3.44. The time averaging, however, increases the time necessary for computation. Furthermore, each FFT frame represented a longer time span i.e. averaging 100 frames resulted in one FFT frame represented 2.5 seconds (for the above scenario). Care had to be taken to ensure that the averaging did not compromise the possibilities of real time application.

Often, more information can be visually gleaned from inspection of spectral contour maps. These are pseudo-two dimensional for frequency and time with amplitude represented by colour or grey scale areas on the plot. Figures 3.45 - 3.47 show spectral contours for the optimum, low and high voltage traces.

From the spectrographic analysis it was noted in particular that :-

1. The vertical readings indicate noise from gas leakage through the flux. This was indicative of excessive spatter and consequently of instability. The concentration of this noise was most apparent on the low voltage weld run.

2. There was a substantial lack of low frequencies, up to 4kHz on the optimum weld voltage setting.

3. With the frequency peaks at 14.3kHz and 11.42kHz identified very little activity was apparent above 10kHz in any of the traces.

As a result of these findings it was decided to reduce the complexity of the FFT as well as the sample rate so that investigations could be concentrated in the 0-10kHz range.

Consequently further weld runs were instigated with the same parameter settings and sampled at 20kHz, enabling a 0-10kHz frequency range capture. The time / amplitude signals were then subjected to an FFT function of lower order (5th order providing 16 real data points) as previously described.

The acquired FFT frames were then considered as descriptive vectors for the process. Investigation began into the groupings and hence separability of the vectors by means of cluster analysis.

3.7.6 Cluster Analysis

The statistical tool used for the analysis was XGOBI, a dynamic graphics program developed at Bellcore. The system was implemented within an X windows system (Unix) on a SUN workstation.

The package provides a scatter plot visual output of three dimensions extracted from a data file. The data file may be of many dimensions rendering it impossible to graphically visualise. The data can be subjected *projection pursuit* where the data is sphered (normalised) by means of principle component analysis, and rotated about multiple axis to locate interesting features such as overlaps and voids.

Typical vectors used for the cluster analysis are shown in figure 3.48. The 16 dimensional data was segregated and identified with suitable labels before analysis. The compiled data file, consisting of 50 examples of each class, was introduced to the XGOBI system and initialised with a projection pursuit algorithm to find overlapping

vector components. A typical analysis display is shown in figures 3.49 and 3.50.

After approximately 8 hours of data manipulation the result showed a high degree of correlation between the optimum and high voltage levels, but more of a clear segregation with the low voltage level vectors.

The next stage was to apply the data to a crude self organising map network, as described in section 2.9.2, in an attempt to segregate the data vectors further.

3.8 Initial Trials of Artificial Neural Networks

The basic initial trails with the ANNs were aimed at data analysis and segmentation. First a self organising map was used to train on the data and the resulting output layer analysed to report on possible segregation or overlaps. Secondly a backpropagation network was instigated in an attempt to sort the three levels of data classes.

3.8.1 Self Organising Feature Map

A basic network was written in 'C' code with no visual interface and with fixed input and output layer dimensions. The input accepted ascii text vectors as inputs and on recall reported only Euclidean distance errors. The output layer weights were also recorded on completion of the learning phase to allow analysis of the generalised vectors.

The SOFM architecture consisted of a 16 node input layer suitable for the FFT input vectors and an output layer measuring 64 neurons (8 neurons squared).

The network was trained on the optimum voltage group only (50 examples), and tested on the remaining groups. The error distances were recorded and are given in table 3.5 together with a graphical representation in figure 3.48 a - f. The resulting errors provided an indication of the relative positions in hyperspace of each group regardless of the classification. It is important to note that at this level the network was not used to learn a classification, but only as a means of attempting to gauge relative position.

The results showed that the reported error for the high voltage set was lower than that for the optimum voltage set, the data it was trained on. This gave an indication that the network perceived this data as a subset of the training data. The low voltage vectors provided a much higher error, indicating more of a separation. The result is shown schematically in three dimensions in figure 3.51.

This theory was further tested by training the same network on the high voltage data and testing with the remaining data. The resulting errors showed a minimal error for the high voltage data, closely followed by the optimum (which still exhibited a degree of overlap) and gave the greatest error to the low voltage data set. This result is illustrated in figure 3.52.

The conclusion was that the data in this form, with appropriate pre-processing, could be adequately segmented for classification purposes. It was hoped that a better separation with more accuracy could be achieved by adding more components to the vector. The FFT function was therefore improved by increasing to a 7th order (128 point) version

This data was then used to test a backpropagation network in an attempt to learn the data classification.

3.8.2 Backpropagation

A three layer feed forward network was created with a commercially available neural network applications tool. The NT5000, developed by Neural Technologies Ltd, was a PC hosted software interface system. The neural network could be constructed, trained, tested and developed before being downloaded to a stand alone hardware system via the PC parallel port. Although the hardware was not used the backpropagation algorithm was extensively used for these tests.

The network architecture consisted of 64 inputs suitable for the now more detailed 7th order FFT vectors and 3 output nodes with a binary level output for optimum, low and high voltage classifications. The single hidden layer initially had 12 nodes. The learning algorithm was backpropagation with a sigmoidal transfer function.

The data set comprised 200 FFT frames from each classification and a further 200 for test and validation sets.

After 1 hour training and 3000 iterations the network showed little sign of converging on a global minimum. The architecture was consequently changed by doubling the number of nodes in the hidden layer to 24. Further training eventually provided a convergence after 25 minutes with the assistance of weight joggling to avoid the gradient descent technique from remaining in a local minimum.

Testing the network entailed the input of the previously unseen remaining data sets and logging the percentage correct classification. Results proved inconclusive with all three test sets yielding between 42% and 51% correct classification.

It was concluded from these tests that the multi-layer perceptron method was not feasible for the classification of these input vectors for the following reasons :-

1. The input data was not totally separable and the degree of correlation hindered the use of an acceptably sized net.

2. The sporadic nature of the input data meant that many local minima existed which resulted in very long training phases with only a small chance of reaching an acceptable convergence.

3.9 Development of a Weld Acoustic Monitor

The results of the feasibility tests undertaken justified the decisions to develop a bespoke monitoring system for the audible sound. The data vectors created by the 7th order FFT (providing an extra 48 bandwidth features with which to work) with a total bandwidth of 100Hz to 10kHz were considered a feasible size for real time analysis.

In addition, a flexible unsupervised learning algorithm needed to be created for interpretation of the acquired vectors. The following system specification was compiled for the experimental work.

3.9.1 Overall Hardware System Set-up.

The transducer system was set-up for use with the audible range ECM and the output from the pre-amplifier checked for gain as previously described in section 3.2.2.

The audio filter system was configured to utilise the audible frequency range of 15Hz - 10kHz. Due to inaccuracies in cut-off frequencies and roll-off this equated to a more likely bandwidth of between 100Hz - 12kHz.

The resistor-programmable anti-aliasing filter within the DSP system was configured to allow a sampling frequency of 20.67kHz using a 2K7 ohm value.

The full system set up is illustrated in figure 3.53.

3.9.2 Data Acquisition Software Development.

The data acquisition facility comprised of a three fold operation :-

1. File handling and DSP configuration.
2. Downloading the FFT software to the DSP facility, initialising and logging the results in a suitable format.
3. Configuring and initialising the ADC interface for the weld variable monitoring.

The FFT program was adapted from assembler source code originally proposed by Papamichalis (1990, pp. 53-136). The adaption included the sample rate and FFT order change together with the associated trigonometrical tables and twiddle factor calculations. For a full description of the operation of the algorithm within a TMS320C30 environment the reader is directed to the reference by Papamichalis (1990).

The remaining operations of the acquisition software were basically PC and data base management. The dedicated system allowed constant data recording to a single file on the PC hard drive. This negated the use of the system RAM and enabled data recordings of greater duration.

Secondary software routines were compiled for data manipulation for post weld sampling, pattern extraction and separation. In this way separate files could be created from the main recording and used as analysis, training, test and validation data sets.

The system flow diagram, screen displays and full source code (ANSI standard C compiled with Microsoft C6) are given in Appendix 2.

3.9.3 SOFM Software System

The self organising feature map software was specifically compiled for use with the data supplied from the acquisition system. However, total flexibility was retained in the number of input and output neurons possible in the design. The system was written with dynamic array routines enabling effective PC memory management. The limiting factor, therefore, on the number of in/out nodes was the available run time RAM space.

The program had two basic routines of training and test procedures. The training algorithm is principally that which is described in section 2.9.2. The core of the routine performs as follows:-

```
learning_procedure( ) {
        read header from input file
        allocate appropriate memory storage
        store input vectors
        optional normalisation of input patterns
        randomise weights
        optional normalisation of weights
        for (iterations done = 0; iterations done < iterations required; iterations done++){
                for (patterns = 0; patterns < total no of patterns; patterns++){
                        compare pattern to weights
                        locate winning unit
                        adapt the weights of the winner and appropriate neighbours
                        decrease neighbourhood size
                        reduce the learning coefficient
                        if (learning coefficient <= 0)
```

```
                    break;
                    }
        }
        store final weights as trained network
        write appropriate header for output file
        write weights to output file
        free all memory
}
```

The test procedure performs a vector comparison with the trained weights of the network and recorded all the information concerning the output layer errors, output node firing order and the frequency of node activations. The principle operation is as follows:-

```
recall_procedure ( ) {
        read header from trained network file
        read header from test pattern file
        allocate appropriate memory storage
        store input weights of trained network
        store test patterns
        optional normalisation of test patterns
        for (patterns = 0; patterns < total no of patterns; patterns++){
            compare pattern to weights
            locate winning unit
            store location of winning neuron
            store Euclidean error from nearest neuron
        }

        write appropriate header for output files
        write firing order
        write frequency activations
        free all memory
    }
```

Graphical procedures were added to assist in visualising the training and test results from the algorithms. These included input vector analysis, Euclidean error distances and network details such as learning coefficient, learning rate and output node activations. A three dimensional topographical representation of the network output layer was also created to monitor the response levels of the nodes to specific pattern presentations on both learning and recall schemes.

The optional normalisation routines allowed either a standard vector component or vector augmentation methods.

The complete source code, flow diagrams and typical visual displays from the system are given in Appendix 3.

3.9.3.1 Benchmark Testing.

In order to assess the training and recall algorithms, a series of tests were instigated utilising 2 & 3 dimensional data of known characteristics. Various configurations of data were tested including two dimensional triangular, cross and random, and three dimensional spherical and random.

The learning coefficients, rates and neighbourhood sizes were also varied for the tests and the results calibrated against speed of learning and accuracy of recall.

A typical learning progression is illustrated in the series of graphical outputs shown in figure 3.54. A set of two dimensional cross data is given to the network to learn. From a random initiation of the network weights, the iterative learning process amends the weights to converge on the input data. The starting learning coefficient, its rate of decrease and the size and reduction of the neighbourhood determine the speed and degree of homogeneous activation of the output nodes.

Examining the output layer activations it was noticed that the even deployment of the weight values was indicated by a relatively even firing of the nodes as shown in figure 3.55 a - b.

Testing the trained network on previously unseen data of the same configuration yielded a similar response in the output layer although, in some cases the overall Euclidean error would increase or decrease depending upon the distribution of the data.

Testing the network on data of a different configuration, such as a two dimensional triangular set, revealed an altogether different response. The distribution of the new test data increased the overall Euclidean error as well as providing a distinctive change in the output layer activation map. (Figure 3.56).

Testing on various other configurations in three dimensions showed the same level of responses. Testing the training process with changes in learning coefficients, learning

rates and neighbourhood sizes yielded different responses in terms of reaching a satisfactory level of learning.

In instances where the learning rate was too large, the output layer tangled as shown for a three dimensional example in figure 3.57. The result was a network which was stationary at a local minima and which failed to learn satisfactorily and completely cover the input pattern space.

This was generally alleviated by intialising the learning coefficient to approximately 0.4 and using a non-square output layer (Kohonen, 1992).

3.9.4 Data File Formats

A format scheme was arranged for all the data files generated by the PC programs to enable full compatibility with the system modules. All files contained information headers to transfer specific data from the data acquisition phase to the train and test phases.

All recorded data was ascii text delimited for full portability to most conventional text editing and spreadsheet packages. Table 3.6 illustrates the complete scheme with associated file extension names and descriptions.

3.10 Weld Voltage Testing

The next level of weld tests were designed following the optimisation of the custom designed software systems.

A series of welds was instigated with the current and travel speed parameters constant, but the voltage changed in incremental steps along the full scale range of 25V to 50V. The welds were all approximately 800mm in length and the audible sound digitally recorded for the entire lengths. Two weld lengths were completed for each voltage level.

The voltage increments together with the recorded current and travel speeds are given in table 3.7.

The recorded data was edited so as to create a training set and a test /validation set. The training set consisted of 1800 FFT frame (7th order) which was composed of 200 FFT frames per voltage level.

The following network statistics were used:-

Input nodes:-	64
Outputs nodes:-	187 (11x17)
No. of input patterns:-	1800
Training iterations:-	200
Ordering counts:-	50
Pattern presentations:-	450,000
Normalisation:-	vector component only
Start learning coeff:-	0.4
Start coeff decrement:-	0.99
Start neighbourhood:-	64.7%
Re-parameterisation:-	3 iterations = 66 steps
Training duration:-	17 hrs (approx.)

The remaining half of the acquired data was used to test the trained network. The result showed conclusively that the FFT frames could not be separated into individual clusters indicative of the incremental changes in weld voltage. It was therefore clear that there were many FFT patterns common to all recorded voltage levels. Figure 3.58 illustrates a typical series of output layer activations. The output node firing remains random with no indication of clustered activation when the network is presented with related pattern groups. Figure 3.59 shows the resulting monitored and predicted voltage trace.

The experiment was then repeated with the same set-up parameters as described above. However, the training data was, this time, the mid point/ optimum voltage level only (500 patterns). The network trained much faster with only two hundred patterns and reached an acceptable level of equilibrium in approximately 100,000 iterations (four hours approx.).

The network was tested on the remaining data from the welds of varying voltage.

3.10.1 Preliminary Results

By training only upon one voltage level any segmentation apparent in the output layer was confined to differences or similarities in the 64 dimensional input vectors or FFTs. This did not provide the pattern classification necessary to yield a metric of the welding process. Further investigation was given to the behaviour of the tested network in terms of output neuron activity.

3.10.2 Firing Order

The output of the tested network provided data files which included the order in which the winning neurons fire. This was an attempt to establish any structured order or temporal progression of the sound emanating from the weld.

Figure 3.60 show typical firing orders for a range of voltage levels. In conclusion no structured order could be established within the number of samples given to a specific window of data.

3.10.3 Output Layer Activations

The output layer activations provided the most encouraging evidence for the discrimination of weld voltage levels. Although, as previously discovered, many of the FFT frames were common to all weld levels, the patterns evidently appeared at different intensities. The change in pattern concentrations altered specific areas of the activation map.

Figures 3.61 shows a series of activation maps generated by the test data and illustrating the neural network interpreted differences in voltage changes.

3.10.4 The Activation Map Classifier.

With the output layer activations appearing to be a satisfactory gauge to the process

state it was necessary to automatically classify the maps. In principle the output layer was considered a three dimensional grid or retina with grey scale coding which constituted another pattern recognition problem.

Past problems involving image recognition have utilised both self organising maps and multi-layer perceptrons. The size of the retina was, however, quite large being the same as the number of output nodes of the SOFM (in this case 187).

It was decided to construct a backpropagation network for the following reasons :-

1. The activation maps were of known classification.

2. Early indications were that segmentation was possible.

3. With the number of output nodes required to produce an acceptable classification rate and generalisation, the SOFM could have become very large and exhibit PC memory problems.

4. The backpropagation would provide a floating metric of the required variables including a generalised measurement for novel maps.

5. The SOFM method could only supply a given label for the class.

Consequently a multi-layer perceptron, utilising the backpropagation algorithm, was used to train upon the activation maps and provide weld metrics.

A neural network development tool, NeuralWorks Professional II (1993), was used for this task. This is a very comprehensive software tool allowing fast development of many ANN architectures and learning algorithms (figure 3.62). For a full specification the reader is directed to the reference.

The architecture used was as follows :-

Input nodes:-	187
Hidden layer nodes:-	24
Outputs nodes:-	1 (voltage level)
Learning rule:-	Delta Rule
Transfer function:-	Sigmoid
Training iterations:-	50000

Pattern presentations:- 9 classes x 10 examples x 50000 iterations = 4.5×10^6

Training duration:- 8 hrs (approx.)

The network appeared to converge at a global minimum without interference in approximately one hour. The training set was, however, quite small for this test and further weld data was required at a later date for statistical confirmation.

The trained network was tested on previously unseen activation maps with the result as given in figure 3.63. As can be seen a close correlation, calculated at +0.997, exists between the actual classifications and the resulting ANN predicted values.

3.11 Conclusions and Experimental Amendments

Following the preliminary tests and results from the experimentation, the decision was made to adopt the methods previously described to establish the major weld control criteria. These were to include the weld current and travel speed as well as on-line stability assessment.

The final experimental method is illustrated in figure 3.64. Some amendments were made to the data organisation, the number of samples used in each of the neural phases and the architectures used. These were as a consequence of the following conclusions.

1. The time duration for collecting and PC processing 200 FFT frames was proving to be too long for effective real time monitoring and control. Consequently the number of FFTs per window was decreased to 50.

2. Analysis of the activation maps showed a large redundancy of neurons within the output layer. Experimentation would be required into the reduction of neurons or improvement of the learning algorithm.

3. The reduction in neurons in the activation map would also reduce the number of neurons in the input layer of the backpropagation classifier rendering an overall faster response.

A series of welds were therefore conducted with various changes in weld control parameters as well as pre-programmed instabilities such as incomplete flux coverage.

3.11.1 Final Data Preparation

The final set of weld recordings was instigated to acquire all data necessary to train and test the final system off-line.

The weld data consisted of edited sets of ascii data files collected from a total of 100 welds, each of approximately 750mm in length. These consisted of :-

1. Voltage Change	= 10 incremental steps x 2 welds	= 20
2. Current Change	= 10 incremental steps x 2 welds	= 20
3 Speed Change	= 10 incremental steps x 2 welds	= 20
4. Stabilty Trial	= 10 incremental steps x 2 welds	= 20
5. Flux Changes	= 3 parameters (V, I, Ts) x 3 levels x 2 welds	= 18
6. Optimum Setting	= x 2 welds for SOFM training	

An average of 800 FFT frames were collected from each weld enabling the total construction of 100*(800/50) = 1600 activation maps for further backpropagation network training and testing.

The list of recorded welds are given in tables 3.8 - 3.12.

4.0 RESULTS

4.0 RESULTS

4.1 Neural Based Weld Acoustic Monitor

The weld runs described in the previous section were completed on cold parent material to ensure consistent stand off (negated plate warpage) and to reduce the possible effects of plate pre-heating. The weld recording was initiated after the arc had established and the weld was considered to be in consistent motion. Similarly the recordings were terminated prior to the weld finishing so as to eliminate as much transient run-in and run-out noise as possible.

One of the main difficulties associated with the data acquisition was the location of PC equipment in close proximity to the welding installation. Establishing a welding arc under poor initial parameter settings often lead to a charge build up at the electrode tip. The sudden discharge at arc strike was enough to create a power surge which would often 'lock up' the PC monitoring equipment.

Subsequent provisions of electrical shielding and positions of earthing terminals alleviated the problem to a degree, but never completely eliminated it.

4.1.1 Weld Parameter Monitoring

Typical process parameter recordings are given in figures 4.1 - 4.3. The method adopted for recording was such that the data files were of alternate FFT traces and parameter readings so as to enable a label for each FFT frame. This incurred a time penalty between the DSP and PC data logging.

The X axis represents the number of sampled pairs of FFT and parameter readings for a complete weld run of approximately 650mm. This equated to a sample period of 100ms for a welding speed of 6.5mm/s over the whole run. This is, however, misleading as the time penalty incurred by on line computer graphics and data downloadings to the PC hard drive compounded the cycle time calculation. As the FFTs were recorded as windows of 50 frames the actual computed sample time for each frame within the window was 24ms (approximated from the system clock) .

The voltage trace appeared to be relatively stable and showed an adequate calibration for the desired preset value. However, transient noise was very apparent on the current and travel speed recordings. In some cases transient spikes across the full scale range were present.

Post processing of the data, off-line, enabled a running average to be calculated which yielded a much more stable trace as shown in figures 4.4 - 4.6. Although the mean values equated favourably with the preset parameter values, this method eliminated any transient information indicative of certain transfer modes within the process.

4.1.2 Sound Monitoring & FFT Traces

The transducer arrangement, pre-amplifier and active filter set created few problems with the operational sound recording. Throughout the entire set of experiments the input signal to the ADC system within the TMS320C30 DSP remained reliable. The amplifier gain proved ideal with no ADC saturation becoming evident.

The on-line FFT calculation and PC logging yielded consistently good information. Post analysis revealed an excellent correspondence with previously acquired off-line data. The FFT frames illustrated in figures 4.7 - 4.9 show overlapping groups of fifty frames as used for the training data for the SOFM phase of the system.

Figure 4.7 a-d shows typical FFT frames associated with changes in voltage. The similarities can quite clearly be seen in overall vector shape but with differing relative amplitude scaling within the 2.5kHz - 5kHz bandwidth.

The low voltage welds exhibited an overall higher amplitude signal especially in the 5kHz - 10kHz range. A more well defined FFT pattern was evident in the mid voltage range with a generally lower amplitude reading for the higher voltage welds.

Changes in weld current are illustrated in figure 4.8 a- d. At the low current setting there was noted a distinct absence of consistent activity in the frequency range above 5kHz. A clearly concentrated overall shape is apparent below this frequency however, with erratic high amplitude traces occurring throughout the recording.

Higher amplitude readings appeared to be in accordance with a higher voltage level. At 350 Amps a more consistent but low amplitude activity began to appear within the 5kHz - 10kHz range with the overall trace becoming more stable. At 625 Amps relative amplitudes increase until at 930 Amps the FFT trace flattens at a higher though more consistent amplitude.

Changes in weld speed also exhibited amplitude changes in FFT traces though not as distinct as in voltage and current. The shape of the FFT remained consistent and virtually unchanged at the low half of the spectrum up to approximately 6kHz. It was noted,

however, that a higher but less erratic signal was obtained above 6kHz for weld speeds above 10mm/s. (Figures 4.9 a- d). The higher amplitudes for increased weld speed could be partly explained by the raised sound pressure level due to a decrease in the flux coverage at the weld head. Due to the design of the weld nozzle, the flow rate of the gravity fed flux was too low, at high welding speed, resulting in an exposed arc.

A more comprehensive set of FFT data frames indicative of weld process parameter changes can be viewed in Annexe 6.

4.1.3 Frequency Activation Maps

The collected weld data files were preprocessed by stripping the parameter measurements from the FFT frames and separated into training and test/validation files.

The FFT training file consisted of 1500 frames collected over two complete welds on the optimum parameter setting of 40 volts, 550 Amps, 6.5mm/s. The SOFM training algorithm as described in section 3.9.3 was instigated with the following strategy :-

Input nodes:-	64
Outputs nodes:-	187 (11x17)
No. of input patterns:-	1500
Training iterations:-	200
Ordering counts:-	50
Pattern presentations:-	375,000
Normalisation:-	vector component only
Start learning coeff:-	0.4
Start coeff decrement:-	0.99
Start neighbourhood:-	64.7%
Re-parameterisation:-	3 iterations = 66 steps

The algorithm was left to run its course with periodic analysis of the activation map during training. At certain stages the activation map exhibited areas of constant activity which indicated the possibility of tangling (local minima) or a concentration of specific data patterns indicative of a non-even distribution. At these times the learning coefficient and/or neighbourhood functions were adjusted with the results monitored at the next iteration.

An acceptable level of homogenous firing was eventually encountered and consequently established as the trained network.

The data files collected from the welds of varying parameter settings were separated into groups of 50 FFT frames for use as test data and introduced to the SOFM recall algorithm as described in section 3.9.3. Examples of the resulting activation maps are given in figures 4.10 - 4.12.

Initial difficulties were encountered in establishing an acceptable size for the output layer of the network. The number of neurons had to be adequate for the separation of the FFT patterns within the training data as well as providing appropriate association classes for consequent recall schemes. It can be seen from figures 4.10 - 4.12 that on recall a substantial part of the output layer remained inactive for each pattern set presentation. However, different areas are activated depending on the applied input class. This indicated that the majority of neurons were not entirely redundant but remained dormant depending on the input pattern set.

It was noted, however, that the central neurons mainly became active during recall of the training set class of patterns, with the remaining peripheral neurons firing with patterns not associated with the training class.

4.1.4 Code Book Vector Labelling and Analysis

Although not considered an ideal solution to the activation map classification problem, an SOFM was further used to investigate the possible segmentation of the activation maps.

The activation map data was organised as a single row vector of 187 components and introduced to a training scheme of the following architecture :-

Input nodes:-	188
Outputs nodes:-	400 (20x20)
No. of input patterns:-	100 (parameter at 10 levels x 10)
Training iterations:-	200
Ordering counts:-	50
Pattern presentations:-	50,000
Normalisation:-	vector augmentation
Start learning coeff:-	0.4
Start coeff decrement:-	0.99
Start neighbourhood:-	90%
Re-parameterisation:-	3 iterations = 66 steps

The input patterns were activation maps created from changing an individual process

parameter. That is, activation maps indicative of changes in voltage, current and travel speed were trained as separate fields with this network.

On completion of an acceptable training period a labelled test set was constructed from the original data files and applied to the network. Each winning neuron adopted the label of the input pattern. The labelled patterns or codebook vectors were iteratively introduced and the Euclidean distance of each winning neuron logged. In the case of a single neuron winning two or more separate labels the label associated with the smallest Euclidean error remained valid.

A typical result for the voltage change class is given in figures 4.13a-b. The range of voltages 25V - 49V and patterns associated with ambient noise were applied to the trained network and colour coded to assist output layer analysis. A relatively clear segregation began to emerge showing the possibility of activation map segmentation.

It was apparent however that anomalies still existed with 'rogue' neurons appearing within an established cluster. Similarly redundant neurons created problems where no clear association was able to be given to a tested input pattern.

The training and output layer labelling was repeated with examples of **all** activation maps (300 in total). It was found that a similar segmentation was possible with less redundancy in the output layer. Rogue neurons and unclassified patterns still existed however.

The results of the final tested networks are given in figures 4.14a-c. One Hundred activation maps for changes in voltage, current and travel speed (300 in total) were tested against the labelled output layer. The test set represented ten incremental changes within each parameter to enable a percentage correct classification measure.

For the voltage monitor, 8% of the output layer remained unclassified and 23% of the maps misclassified. However, out of the misclassified points 30% were outside a +/-10% tolerance of the desired value.

The data classification for current provided a similar 6% redundancy, 28 % misclassified but with 27 % of the misclassifieds outside a +/- 10% tolerance of the desired value.

The worst classification was exhibited by the travel speed node. A 10% redundancy was apparent, 40% misclassified with 12% of the misclassifieds outside +/-10% of the desired value. It was noted that the node had yielded the greatest error when attempting to predict high speed welds greater than 15mm/s.

Providing a classification in this way was considered unsatisfactory as the only generalisation being exhibited was in the original formulation of the network weights. The final recall was not a calculated metric via the networks internal representation but simply a vector label for the associated class. Furthermore, to deal with a compound action, such as the changes in multiple parameters, the output layer size would exponentially rise to accommodate the permutations at the necessary resolution.

4.1.5 Attempts At Further Data Compression

It was anticipated that the sizes of the networks being used could provide a memory / speed problem when instigated on-line. The sizes could have been considerably reduced if the data sets were compressed, coded or enhanced by a more definitive feature. Consequently further trials were instigated in an attempt to create a more sparse data set for both the FFT traces and/or the activation maps.

4.1.5.1 Cepstral Analysis

Cepstral analysis of the sound data could have produced a frequency modulation component more indicative of the overall FFT vector shape as described in section 2.8.3.2. Figures 4.15 a- c show typical results from the cepstral transform of sound signals acquired from an overall change in weld voltage.

It can be seen that although virtually no activity is exhibited above a queferency of 4kHz the traces are almost identical. It was therefore concluded that the sound information possessed no distinguishable attributes within the cepstral domain and the analysis was aborted.

4.1.5.2 Wavelet Transforms

As described in section 2.8.3.3, wavelet transforms are used as a method of data compression. A series of 'daub4' level wavelet transforms were instigated on both the time/amplitude signals and the FFT traces. Figure 4.16 shows the resulting transform for a 4096 sample window of transient data. The experiment re-emphasised the cueing problems originally considered as problematic and was therefore not considered a valid option for signal compression and comparison.

Figures 4.17 a- c illustrates the results from a series of FFT traces associated with a voltage change. Strong similarities were noted between changes in all parameters but subtle differences occurred with the amplitudes of the coefficients offering the possibility of creating sparse representations. It was, however, considered a time consuming

exercise, possibly doubling the necessary on-line pre-processing time which was already at a premium.

The discrete fast wavelet transform necessitates a data window size to be a factor of 2. If used for activation map compression the process would have dictated an SOFM output layer size of 64, 128, 256 or 512 nodes (memory permitting) thereby removing flexibility. In conclusion this method was considered inappropriate for activation map compression at this time.

4.2 Neural Network Performance

The following section relates the overall system performance tested both off-line and on-line real time. The backpropagation classifier was extensively tested off-line against measured parameter readings acquired from previous welds. The training and test data sets were reconstructed by amalgamating the SOFM created activation maps with the appropriate, originally monitored, parameters. The parameter readings were averaged, point by point, over a sample window of 50 and used as the network target values.

Due to the fluctuation of the target values i.e. the changing weld parameters, the network performance metric is expressed more as a control system response test than a hit/miss classifying system.

4.2.1 Backpropagation Classifier

With the activation maps created for set weld parameter changes, the backpropagation network was constructed for the final classification phase.

Initially, it was anticipated that a modular learning system could be constructed where each monitored parameter would be instigated within its own network enabling easier interchangeability. As a consequence separate Multi-Layer Perceptron (MLP) configurations were used to train on and test the individual target parameters of :-

1. Voltage
2. Current
3. Travel Speed
4. Energy Input (calculated from time averaged recordings of V, I & Ts. See equation 2.1.1).

Each network accepted the 187 inputs from the activation map through a 24 node hidden layer to a single node output layer. The resulting networks each readily converged and provided a test correlation over each test set ranging from +0.93 - +0.96.

The problems associated with instigating and testing on-line parallel networks resulted in the development of the final single network of the following specifications :-

Input nodes:-	187
Hidden layer nodes:-	24
Outputs nodes:-	4 (V, I, Ts, E)

Input Patterns:-	240 (80 maps/parameter)
Learning rule:-	Delta Rule
Transfer function:-	Sigmoid
Training iterations:-	50000
Pattern presentations:-	12×10^6

The final network converged with an overall output node r.m.s error of 0.0390 after approximately 30000 training iterations.

The system was tested off-line with the remaining 240 activation maps which yielded the following results.

4.2.2 Monitoring Voltage

Figure 4.18 shows the resulting input/output relationship established by the network for voltage monitoring. Oscillations were apparent across the full scale range with an r.m.s error value of 2.37 volts equating to 9.5% of the full scale range.

The greatest deviations were evident at the extremes of the scale below 25V and above 48V. With a correlation coefficient calculated at +0.9194, the result showed a positive trend towards the actual monitored values. However, in some instances error swings of up to 10V were exhibited.

4.2.3 Monitoring Current

Again, transient oscillations were very much evident in the resulting trace as given in figure 4.19. The predictive values were more stable for the low current welds yielding a more positive correlation. At high current, the network performed less well failing to reach the monitored full scale range.

The r.m.s error was calculated to be 53.9 Amps which equates to approximately 7% of the full scale range. The overall correlation coefficient was established at +0.9375 emphasising a positive trend to the desired value.

4.2.4 Monitoring Travel Speed

Figure 4.20 illustrates the resulting response from the travel speed node. Again, the limits of the full scale range created the greatest differential with high speed welds above 20mm/s causing large oscillations up to 10mm/s or approximately 50% of an instantaneous value.

The r.m.s error was calculated at 1.55mm/s (7% of the full scale range) and the overall correlation coefficient computed as +0.9226.

4.2.5 Energy Input Monitoring

The greatest response oscillation was exhibited by the weld energy input node as illustrated in figure 4.21. Although the oscillations were noted to be occurring about the desired value, individual errors of up to 2kJ/mm were not uncommon for the higher energy welds.

These oscillations could be explained partly by the construction of the training and test sets. The energy node value is dependent upon the voltage, current and travel speed. These other nodes are, in the main, independent. Consequently a fluctuation in **any** of the governing nodes (V, I, Ts) will create disturbances in the dependent energy node. During training, therefore, this node found more difficulty in reaching a steady state.

However, the r.m.s error of 0.51kJ/mm equated to 9.2% of the full scale range and the correlation coefficient was calculated at +0.9085.

4.3 Monitoring Flux Supply

Further networks were developed utilising the same principles, supplementary to the electrical parameter networks already described.

A series of experiments was designed to test the ability of the network to register a depletion in the flux supply during welding. The tests were conducted for three levels of individual parameter settings, high, low and medium for voltage, current and travel speed.

One weld was conducted at each setting with ideal flux supply. The resulting FFT frames were used to train individual SOFMs representing ideal weld set-ups for voltage, current and travel speed. Fifteen hundred FFT frames were used to train each map with a remaining fifteen hundred being used as test sets.

A scale of zero to one was adopted for levels of flux supply relating to no flux and ideal flux (for the adopted standoff) respectively. Four welds were run with the flux supply valve limited to 3/4, 1/2, 1/4 open and completely closed. In the latter case the flux supply was closed after arc initiation. In all cases suitable precautions were taken to avoid the resulting arc flash which occurred during many of the recordings.

The resulting FFTs were labelled into sections of 50 samples. These samples were tested on the trained SOFMs and the activation maps recorded and labelled. The activation maps were used to train individual backpropagation maps with single output neurons giving a floating point output between zero and one corresponding to the flux supply level.

Further welds were then instigated at the three levels of parameter settings and the flux supply incrementally cut off over the length of the weld. The FFT samples were recorded with the sample number at flux cut-off noted. The resulting data was processed off-line and tested against the appropriate trained networks. The results are illustrated in figures 4.22 a-c.

It can be seen that for depletions in flux occurring over changes in voltage (fig 4.22a), the response is relatively clear. Although there are inherent time lags, the final response shows that the network interprets a definite lack in flux coverage. However, deviations were quite noticeable and the full scale range was not achieved.

A similar response was exhibited by flux deprivation over changes in current (fig. 4.22b). Again the full scale range was not achieved and oscillations between 1 and 0.7 were

apparent in what should register as an ideal flux coverage situation. There was, however, a clear indication that the flux had been reduced within two decisions of the network.

For both changes in voltage and current with flux depletion, the response time averaged to approximately within two time slices or sample widths of 50 FFT frames. This equates to a response time of 2.4 seconds for a travel speed of 6.5 mm/s.

The changes in travel speed created problems for this network as shown in figure 4.22c. In the main the network failed to register any depletion in the flux supply during most welds. A slight response was given by the slow speed weld but not truly indicative of the welding situation.

Problems were associated with the data collection for these welds however. Attempting to reduce the flux supply by hand as at high welding speed was especially tricky!. Further more, welding at high speed reduced the effectiveness of the flux coverage (for the design of nozzle used) as the arc tended to outrun the flux supply. The result was that a reduced flux supply at high welding speeds sounded 'normal' as the network was partly trained on high speed weld signals.

4.4 Overall Weld Stability Assessment

As previously described in section 2.1.2, stability is a subjective quality and as a consequence a difficult parameter to measure. Work by Ogunbiyi (1995, p.3) states that spatter is the main visual indication of weld instability, which in turn is mainly associated with globular transfer. Unfortunately, for experimental purposes, submerged arc welding exhibits little spatter due to the flux coverage, which in turn prevents visual inspection of the welding arc. Consequently weld instability assessment for SAW is not particularly straight forward.

Globular transfer is mainly associated with high voltage / low current welding (Chawla, 1993) as given in figure 2.4. A series of welds was therefore created with parameters set as given in table 3.11 in an attempt to force an unstable weld via globular transfer.

The stability indices proposed by Ogunbiyi and Norrish (1996) were used to test the result. Figure 4.23a-d show the calculated indices for an ideal weld with the results from the unstable weld given in figure 4.24a-d.

Comparisons in transfer index, although erratic, show underlying trends that for the stable weld the transfer index remains mostly under 0.1 (ideal spray) and with the unstable weld averaging 0.2 (increasing short circuit randomness). (Figure 2.5)

The Dip consistency index appeared erratic and comparatively inconclusive, whereas the stability index gave clear indications, with the stable weld averaging 1.5 (ideal dip) and the unstable weld averaging 3 (increasing spatter and roughness). The power ratio also gave a clear difference with the stable weld averaging at 0.95 (stable spray) and the unstable weld 0.78 (unstable / excessive spatter).

Ogunbiyi and Norrish state that changes in power source characteristics can cause inconsistencies in readings for Transfer, Dip consistency and Transfer stability indices Ogunbiyi & Norrish, 1996). The main definitive comparison was therefore taken through the power ratio index, although a secondary comparison was made via the calculated stability index.

The FFT traces acquired during the unstable weld recording were introduced to the previously trained SOFM network indicative of an ideal weld. The resulting activation maps were then used to train a new backpropagation network with two output nodes with floating point values for power ratio and transfer stability indices. The final trained neural system was tested against a series of welds with compound parameter settings. That is, the weld voltage was increased at the same time as the weld current was

decreased. This allowed an initial ideal setting and a steady control change to a forced unstable situation. The collected data was tested off-line with the results given in figures 4.25 a-d.

Although the overall correlation between the neural output and the calculated values of the indices was relatively low (+0.56) there was general agreement concerning the point at which the weld was to be considered unstable.

The resulting neural stability index was post processed with an alarm state given at the index value of 2 (fig 4.25.b), a clearly unstable weld was registered, beginning at sample number 600.

It was further noted that a similar response was provided by the power ratio node. At sample number 600 the output dropped below the threshold of 0.8 indicating instability (fig. 4.25d).

The results compared favourably with the calculated indices as calculated via the monitored weld parameters. The off-line comparisons, however, did not take into account any time lag present within a real time system due to the data window size. It was anticipated that a time discrepancy of at least one sample window (1.2 sec) would occur in an on-line scenario for the present system.

4.5 Run Time Monitor

A series of ten test welds was initiated to assess the on-line characteristics and accuracy of the electrical parameter monitoring networks.

The ten welds were instigated with the parameters varied over the length of the weld. All data was logged as normal and compared post weld.

4.5.1 On-Line Results

Figures 4.26 - 4.33 show the typical final results of the overall system. Figure 4.26a illustrates the voltage monitoring result. An overall correlation coefficient of +0.9438 was yielded from the complete weld as compared to the monitored trend. Figure 4.26b however, shows the actual on-line output as given by the neural node. The time lag associated with the network having to assess a window of 50 samples (1.2 sec) increased the overall error.

Similar results were obtained for the remaining nodes. Overall correlation coefficients ranged through +0.9 for the energy node, +0.94 for the voltage and travel speed nodes and +0.95 for the current node.

Figures 4.30 - 4.33, show the neural r.m.s error per window of data for each of the output nodes. The largest single error was shown to be for the travel speed and voltage nodes with a 16% difference over the full scale range. The smallest errors from the series of individual outputs was given by the energy node with a 10% full scale range deviation.

The results obtained on-line were similar in correlation to the results obtained off-line. The greatest error exhibited on any particular node was travel speed where a consistent error was apparent for high speed welds above 15mm/s.

The full results are summarised in table 4.1, which also provided the details of the trained network schemes and data types.

5.0 DISCUSSION

5.0 DISCUSSION

The literature reviewed has taken into account both the biological and mechatronic aspects of signal monitoring, pre-processing and analysis. From the basic principles of audible sound recording to complex digital signal processing the research has attempted to provide an artificial model of the biological concepts.

The inevitable difficulties are apparent in virtually all the areas considered. The missing links within the understanding of biological auditory systems have resulted in a hypothesised scheme of probable events with which to work.

Attempts at replicating these schemes have their own inherent technical problems of memory capacity, operational speed and not least of all, stability. The overall results, however, have demonstrated an artificial neural network technique to interpret erratic sound emissions from submerged arc welding.

Elements of data transforms and compression, fused with pattern recognition and classification have produced a biologically inspired method which has yielded significantly favourable results.

The method used differs from past research experimentation with arc sound, which has primarily been based upon direct statistical correlation of sound pressure levels and/or frequency domain information with known process parameters. The neural network approach adopts a replicated human learning scheme in which the system is given suitable sound data with additional information concerning the process state.

5.1 Manual Welders and Sound

The investigations into what manual welders are actually listening for when welding or what sounds are indicative of particular circumstances, have yielded only a subjective and somewhat vague set of descriptions. Adjectives such as 'harshness', 'smoothness', 'regularity' and 'crispness' are commonly used to describe the sounds of a stable or unstable weld.

What is evident is that manual welders are unable to pinpoint the exact parametric control necessary to produce the correct sound. That is, during welding, the welder is incapable of constantly reciting the voltage, current, travel speed, energy input and a measurable degree of stability. Instead the weld is deemed good or bad and with a tweaking of electrode position it is either maintained or corrected.

What also has to be taken into account is the overall importance of sound to a welder. The audible acoustic emission is only part of the story. A prior knowledge of the process initially enables the correct electrical settings. The Visual analysis of the weld pool is obviously of prime importance to enable accurate seam tracking and travel speed control. The weld pool size and subsequent bead shape can be controlled via the angle of the electrode. The dexterity and reactions of the welder can result in a difference in competence. The question is, therefore, is a manual welder who is deaf equally as inappropriate as an electrician who is totally colour blind?

Although rumours abound concerning the importance of sound in welding, there is little documentary evidence as to its degree of importance to a **manual** welder.

In terms of manual weld monitoring, sound appears to be used as a secondary indicator for the process. It is used in a supplementary sense to provide further evidence as to how the process is being conducted. Furthermore, it is apparent that manual welders are more able to recognise the sound of a stable weld than they are of describing the faults causing an irregular sound pattern. This method of sound monitoring can be assimilated to a form of novelty detector. Experiential learning reinforces the pattern associated with a good weld which the welder constantly strives to attain. Any deviation from the learned pattern registers as inappropriate and avoidance measures are instigated.

What evidence there is to the use of audible sound generally points to an association with the *consistency* of the signal and the overall stability of the weld where irregular sounds indicate instability. This corresponds with the already established fact that unstable welds are conducive to excess spatter which, in turn, is often caused by the inherent characteristics of globular transfer.

It can be concluded, therefore, that the consistent or regular sound patterns are a result of a stable material transfer mechanism which can be controlled via the optimum set-up of electrical parameters and electrode position.

5.2 Audible Airborne Acoustic Emissions and S.A.W

Unlike many welding processes, submerged arc welding provides great difficulties for certain analytical tasks. Although a relatively clean process in terms of reduced arc flash and material wastage, the method precludes any visual analysis. Furthermore, the covering of flux creates a sound damping system which compounds any signal which may be purely indicative of the arc behaviour. Similarly the depth of flux can also change the overall sound signal. Transient noise is often caused by the shielding gases escaping from the molten flux through the granular exterior. The effect is less likely to be so explosive under a heavier flux coverage.

Flux coverage is also responsible for a very low occurrence of spatter (Houldcroft, 1989, p.19). As already discussed, spatter provides a clear indication of instability and with this minimised audible evidence is obviously reduced.

The obscured arc prevents visual evidence of transfer mechanisms. Indications of material deposition rates and transfer types can only be truly verified by expensive infra red thermography or x-ray photography. Within the context of this research such properties could only be assumed through analysis of the transient electrical data.

5.2.1 Temporal Sequences

Initial sound recordings in the frequency domain provided little scope for correlation with known weld parameters. Individual FFT frames provided a 'snapshot' equating to a time window of approximately 20ms. In biological terms to make an audible classification with just this amount of information would be the equivalent of playing a game of 'name that tune' with less than a semi-demi-semiquaver!.

The theories outlined in section 2.6.1 showed that temporal integration intervals can vary depending upon the specifics of the task. If both current theories are combined analysis windows could possibly be sampled at 4ms over a 200ms period rendering a acquisition window of fifty samples. The importance of sampling an appropriate window of data is very apparent as the temporal progression of the sound carries more information than the individual samples themselves.

5.2.2 Data Windows

Selecting the correct window size has implications to both the accuracy of the classification and the response time of the whole system. The current set-up evaluates an FFT frame in approximately 24ms and takes 50 frames (1.2sec) to establish an overall

classification state. In terms of quantity and speed this falls short of the performance of human biological systems. This can be attributed to the following reasons:-

1. Biological systems are massively parallel and asynchronous. Any programs instigated within a digital PC environment are purely serial and synchronous.

2. The human auditory system equates to a forty eight channel monitoring set-up. The data acquisition facility provided only a single I/O port.

3. The manner of data acquisition necessitated the labelling of individual FFT frames with parameter measurements. Furthermore, software driven visual displays were needed for validation purposes. This reduced the sampling time considerably.

4. The action of the basilar membrane equates to the use of approximately 52 ganglion cells for a just noticeable difference in pitch. In terms of frequency this means the use of 30,000 neural cells for a bandwidth of 10kHz! The artificial system utilises only 64 input neurons to process the 7th order FFT vector.

5.3 Hardware Equipment Problems

Problems were encountered with the welding rig when attempting to design suitable experiments. The design of the nozzle was such that the flux feed was via the electrode housing meaning variations in stick out and flux depth were not mutually exclusive. Any substantial increases in stand off resulted in inadequate flux coverage. Both parameters can affect the material transfer mode, rate of transfer, overall bead shape and quality.

Similar problems were encountered when attempting to weld at relatively high speeds. The feed rate of the flux via the welding head was not consistent which resulted in only a partially submerged arc.

The welding head was also permanently fixed in a vertical position. This prevented changes in electrode angle. The use of trailing or leading angles has a substantial effect upon weld bead shape and penetration (Houldcroft, 1989, p.30). To solve these problems would require the re-design of the welding head to allow lateral adjustment together with a separate flux supply tube.

The resulting experiments were, therefore, based upon the more easily measurable electrical characteristics via the manufacturers system interface.

5.3.1 Monitoring the Weld Parameters

The weld parameters were supplied as analogue signals within the welding interface system supplied by the manufacturer (figure 3.7). Although the output was calibrated against the l.e.d monitoring unit supplied with the welding rig, inevitable quantisation errors occurred via the adopted ADC arrangement.

The volatile nature of the welding process together with the large electrical power required resulted in highly transient waveforms even with the sample rate set at 20ms. The sampling time is questionable for this experimentation and faster sampling is required to enable the analysis of certain transfer mechanisms and deposition rates.

Within other welding processes such as MIG, weld parameter monitoring is sampled at much higher rates. Chawla (1993) for instance, with the development of the Arcwatch™ system for MIG, provides sample windows of 250ms with a sample rate of 4kHz or 0.25ms. This speed allows the identification of dip transfer, stable spray and mixed mode transfers from the current and voltage traces. Furthermore, suitable filters were required to enable the cancelling of transients associated with traces sampled at these speeds.

Past research into the physics of the S.A.W arc has shown that droplet frequency rates can exceed 90Hz (van Adrichem, 1966: Lancaster, 1984), requiring at least a 180Hz (5ms) sample rate if evidence is to be located in the electrical traces.

The present ADC system was capable of sampling at the required speed, however, the method used to store the data prevented this. The interleaving of FFT data was necessary for post weld data analysis and on-line verification.

The use of a second I/O port within the DSP system would increase the allowable speed considerably. Alternatively, the separate ADC could be maintained with the data logged independently providing accurate post weld synchronisation with the FFT frames could be guaranteed.

In addition suitable active filtration would have to be instigated to cancel transient spikes evident within the traces.

5.4 Software System Review

The modular software system developed for this project was designed for flexibility and as an aid for visual assessment wherever possible. As a consequence the graphical user interfaces were of considerable importance. The nature of the collected data was initially unknown and the substantial analysis and investigations required the use of on-line visual representations.

Within the data acquisition module the 7th order FFT was constantly monitored visually as well as stored digitally. The same was true of the monitored weld parameters, whereby the numeric values of voltage, current and travel speed were constantly updated via the user interface. As already discussed, the penalty was operational speed.

Speed was less important to the off-line training and recall schemes for the self organising feature map development. Accuracy of the learning algorithm, pre-processing of the data (methods of normalisation etc.) and appropriate data storage received priority. Similarly on recall, the storage of trained network weights and results files together with a visual display of the output layer activation maps were of prime importance.

The final phase, the run-time monitoring system, utilised no graphical output at all apart from the visual display of parameter metrics during the welding operation. However, for validation purposes, it was necessary to monitor and record both the neural output and the measured weld parameters for off-line comparison and statistical assessment.

All programs were compiled in ANSI standard C which proved to be an ideal protocol enabling full control over the PC peripherals and memory management. Embedded DSP programs were compiled in TMS320C30 assembler to facilitate an optimum computational speed.

Within a final proven system, the analytical graphics and dual monitoring of process and neural parameters would not be necessary. With further optimisation of the neural algorithms (see section 5.5.3) the overall response time could be reduced to around 300ms on a 486/66MHz PC platform (calclated at 60kMAC/s). Ideally the whole system could be encoded in assembler and embedded within a DSP card with adequate memory and I/O facilities.

5.4.1 Data Management

The nature of the research and the use of artificial neural networks necessitated the

collection and storage of vast amounts of data. The data protocol adopted have already been described in section 3.9.2. It was found that careful cataloguing and storage of data was crucial to the instigation of experiments and analysis of results.

In an operational software context it was essential to have control of the PC memory to enable allocation of data space by means of dynamic arrays. This was of special concern with trained network weights. Data for training and recall could be stored and utilised from the hard drive, whilst the network weights required to be RAM instigated for on-line use. For the larger networks memory limitations such as the 64kB partitions found within DOS were often exceeded.

These problems were solved in the main by appropriate software compiler models and by the instigation of dynamic array routines whereby the data matrix could be split between memory partitions instead of requiring contiguous memory addresses.

5.5 Neural Network Application

The fundamental study of this research was based upon the application of suitable artificial neural network models to interpret arc sound. The absence of formal rules for the construction of such tools necessitates thorough knowledge of the data and the operation of network algorithms. The type of data being investigated is often an indication of the type of network that should be tried, that is if a neural network should be tried at all! A known pattern classification, for instance, may warrant the use of an MLP network utilising a backpropagation algorithm. Data of unknown class and relationship may require an SOFM. It must also be remembered that ANNs are good at producing a generalised output. That is the reason why they are robust enough to deal with novel and/or incomplete input data. Consequently, although the amount of generalisation can often be controlled, their role in straight forward statistical computation is limited because of accuracy.

It is apparent from the reviewed literature that artificial neural networks are a new instrument in the signal processor / pattern recognition engineers toolbox. In some instances ANNs are being hailed as a universal panacea, a black box into which masses of data can be bundled with a miraculous answer dropping out of the bottom. This is obviously not the case and a great deal of consideration has to be afforded to data selection, pre-processing, network architecture choice, learning / recall algorithm set-up and data interpretation.

It is true, however, that new technology brings forth new uses, and trials within unknown situations can reveal important limitations. Setting these limitations is just as valid as providing a successful application.

5.5.1 Biological Plausibility

Biological plausibility may not be the main reason a particular ANN architecture is tried or experimented with. It is true, however, that many of the theories behind the formulation of certain ANN models were biologically inspired.

Within this research, even though a cybernetic approach was adopted from the start, the nature of the data together with the resulting ANN outputs provided, to a certain degree, a biologically plausible method. The primary stage of data acquisition replicates the general mechanics of the middle and inner ear. The FFT performs, albeit very limited, the action of the cochlea where the sound pressure levels are transformed into frequency and intensity components.

Within the secondary stage the generalised frequency patterns are mapped into a neural matrix by the action of the SOFM learning algorithm. During recall the resulting activation map illustrates the nature of the received signal. This appears to fit reasonably well with the present theories of auditory cognition as stated by Wang & Shamma (1995, p. 189):-

> *"It thus appears that features are represented and mapped into response areas in the brain. One can tell what features are there by examining which response areas are triggered by the acoustic stimulus."*

How these response areas are further processed and transformed into thought processes within the brain is unknown and purely speculative. The resulting activation areas however, provided a coded pattern of known classification. It is generally accepted that the SOFM methods are more biologically plausible than the backpropagation method, but within this context backpropogation suited the classification requirement for the input data.

5.5.2 Data Compression

The overall method used supplied a system of data compression. The initial acquired data, once transformed into the frequency domain and analysed as time variant vectors, provided a complex contour mapping. If taken as a 50 frame window of 7th order FFTs this equates to 64 x 50 = 3200 data points per window. Furthermore, these data points are erratic in nature.

By utilising the SOFM strategy the resolution of the output mapping can be controlled via the number of neurons within the output layer (subject to the nature of the input data). The generalising capability of the network also further suppresses the transient nature of the input data. Past research into the use of arc noise has highlighted difficulties in dealing with ambient noise conditions associated with workshop activities (Chawla & Norrish, 1992: Chawla, 1993). The combination of frequency transforms and data compression assists in relieving the overall effect of sporadic noise.

For this particular instance the 3200 input data points are reduced and characterised by the value of 187 output nodes, a reduction of approximately 94%.

5.5.3 Neural Network Optimisation

Creating an optimised neural network should be an important requirement within any

application. Methods of achieving it can take several forms, from the correct selection of network type and number of nodes, correct settings of learning algorithm coefficients through to the speed optimisation of the final source code.

The network architectures used were chosen mainly because of the nature of the data and complexity of the problem. Past research has highlighted the use of BPNs and SOFMs for use in many pattern classification and segmentation problems. It could be argued, however, that more efficient models exist. Most ANN paradigms have been theoretically devised to achieve some degree of pattern recognition, signal/image processing, speech recognition or control diagnostics. For instance ART and STNs are designed for temporal pattern recognition a syndrome, although not repetitive, of the arc sound problem. These networks, although seemingly applicable, present application problems. ART networks are very sensitive to noise in the input data and are memory hungry (Hertz et al, 1991, p. 232), whereas Spatio Temporal Networks perform better on repetitive or cyclic input patterns rather than sporadic signals characteristic of arc noise. Many network systems could have been adapted and experimented with, however, the SOFM and BP networks are the most well understood and are relatively easy to adapt and implement.

The number of nodes used for an architecture is partly dictated by the characteristic of the input data and desired output classifications, as well as the complexity of the problem. Defining the optimum number of hidden nodes for an MLP or the output layer configuration for an SOFM can be a matter limited by available memory space, subject to a rule of thumb, an educated guess, or simply a matter of trial and error.

The networks created for this work were initially designed from the rules of thumb and general do's and don'ts found within much of the literature reviewed. The input layer for the SOFM was dictated by the 64 components of the FFT vector. The number of output nodes was limited partly by the available memory space. The configuration was such that a rectangular layer was created, as square layers are more likely to promote tangling (Kohonen,1992). The 11 x 17 was of a 5:8 ratio providing a golden section.

Within the backpropagation classifying system, the number of hidden nodes was derived mainly from trial and error. The rule of thumb given by Eberhart & Dobbins (1990, p.42) was initially applied yielding 16 neurons ((input + output) ^0.5 + a few). The networks were trained and tested and hidden units added two at a time until no better performance was achieved. The final value remained at 24 hidden units.

Optimising the training algorithms with changes in learning coefficients and rates of learning was achieved using an initial rule of thumb setting with a constant iterative

update whilst analysing the progression of training. Problems were encountered whilst training due to local minima and/or uneven pattern distribution within the training set. In some cases, especially within the SOFM network, output nodes showed degrees of redundancy and uneven learning. Although trained networks were eventually created yielding acceptable recall results, it is believed that faster training and more homogenous output layer activity with less redundancy would have been achieved with the use of a conscience term.

The software optimisation for speed was not taken as a priority. The use of graphics and analytical routines compromised the speed requirement. However, streamlining of the software would be a necessary amendment. In general such techniques as keeping all variables global (not necessarily good C programming practice!) and all procedures as voids would speed the program.

5.5.4 Response Times and Accuracy

The response times acquired by the experimental set-up, as it presently stands, creates an obstruction to its use within a real time control and/or monitoring system. To provide a response time to conform with Houldcroft (1977, p. 291) such that corrective signals are supplied within one tenth of the bead width would correspond to 2mm/6.5mm/s = 310ms (3.2Hz) for a typical weld run. At high speeds, however, for the same bead width with the highest speed for the present weld rig a response time of 2mm/27mm/s = 74ms (13.5Hz) would be required. It must also be remembered that this is the full closed loop control time and not just the monitoring time! The present system, as already shown, responds in around 1 second for electrical and stability monitoring and up to 2 seconds for flux supply monitoring.

The response time of the actual networks is small in comparison with the data processing overheads. For the trained SOFM network of 64 inputs x 187 outputs it would require approximately 12550 MACs to locate 50 winning nodes to construct an activation map. The backpropagation network of 187 x 24 x 4 configuration has 4584 interconnections giving a total of 17134 MACs. A TMS320C30/32 DSP system is capable of approximately 4 - 5 MMACs per second (fully optimised) which results in an overall single classification in 4.3ms. In addition the present system processes 50 FFT frames which represents a potential DSP computation time of 18.5ms (Papamichalis, 1990, p. 47) yielding a total processing time of 22.8ms. This does not take into account the time necessary to acquire a sufficient amount of data with which to make a suitable decision however. Reducing the overall time window could have adverse affects on the time variant FFT contour which is used as the training data for the initial stage of the system.

The suitability of the neural response for automatic control purposes is debatable in terms of accuracy. Although the overall correlation coefficients indicate a close similarity between neural response and measured values, these are representative of a complete weld run. Individual neural responses would provide the measured variable within any control system. The r.m.s error values for particular output neurons are more indicative of the performance, and at present are not within acceptable tolerance levels which are recorded at between +/- 10%-20%. The power source is reputed by the manufacturer to be able to suppress changes in weld voltage of +/- 10% (see section 3.1.2). To justify its use within a real time control system it would be desirable to monitor the weld voltage and the remaining parameters within a statistically constant +/- 5% (within 50% of the power source manufacturers reputed tolerance) at 15Hz (67ms). Increasing the accuracy would require further training iterations with more clearly defined training data.

In comparison to statistical methods however, the ANN approach could eventually provide a faster alternative in some instances. In assessing the use of stability indices Ogunbiyi states:-

> *".....all stability indices require a fairly long sample time, from which standard deviation of process cycle times can be calculated."*
>
> (Ogunbiyi, 1995)

Statistically based weld condition monitors require a large data sample together with relatively long computational time to establish a valid result. With the previously approximated execution time of 22.8ms, the neural method could provide a useful time saving.

5.5.4 Networks as Novelty Detectors

The detection of novelty is certainly an accepted part of psychological thinking, and not a new concept within ANN technology. As early as 1976 work by Kohonen and Oja (Kohonen & Oja, 1976, pp. 85-95) demonstrated the use of SOFMs as a *'novelty filter'* to highlight previously unseen details within a character recognition system. Within condition monitoring the recognition of temporal differences has played a substantial role in bearing fault diagnosis, where changes in vibrational frequency are indicative of certain types of wear (Brunel University, 1993).

The method of novelty detection for changes in arc sound has yielded substantial differences from the trained norm. These differences are highlighted by the changes within a neuron activation map, which are capable of further segmentation and

classification.

The reliability of such a method would require further investigation and substantial statistical testing and examining the portability of the method. Changes in power source characteristics, consumable and parent material would affect the overall strategy of the method.

5.6 Data Validation

The quality of training and validation data necessary for ANN applications has already been discussed in section 2.9.3.

In the context of this research, primary test and validation data was taken as direct measurements from the on-line process. Secondary networks such as the stability network, utilised calculated indices from the measured process parameters. The monitoring of flux coverage was a somewhat subjective and less clinical test.

The measurement of the parameters was taken as a direct correlation with the initial weld trials which determined the bead geometry produced by the welding rig. However, a great deal of faith had to be placed on the manufacturer in that the feedback lines within the PC interface system provided the correct scaled and calibrated information.

The noise experienced within the process data was also a problem. To overcome this the acquired data was averaged over a time window which reduced the transient spikes but at the cost of losing information such as would be indicative of dip transfer. As a consequence the calculated indices may not have been accurate enough. Within a modified system, the use of suitable active filtering would be benificial.

The overall level of accuracy of the electrical characteristics was certainly within an acceptable tolerance of the desired value after averaging however, and as such considered valid for monitoring purposes.

The testing of the neural outputs was instigated against fresh welds to ensure a degree of statistical validation. The amount of data required for this purpose is debatable as in any statistical requirement.

The final results from the neural system showed acceptable correlations to pursue further development. Although the accuracy of the training data maybe questioned in terms of parameter measurement levels, this only reflects a relative scaling difference. The adopted normalisation routines reduce these effects. The relationships between the parameters remain unchanged and thus the overall network performance would not be compromised.

5.7 Limitations of the Present Method

The amount of data required to train the networks is also questionable in terms of viability in a commercial environment. The method necessitates the training of a network on an ideal weld. What is considered 'ideal' can change, not only from process to process, but also between certain applications. The bead geometry applicable to one weld may not be for a different application. This would consequently entail the formation of a sound data library indicative of every conceivable joint geometry together with its novelty forming variations!

This, however, would only be true if monitoring all the electrical parameters. A more viable option would be ANN use for stability monitoring and fault diagnosis associated with flux feed etc. Sounds indicative of transfer mechanisms associated with unstable welds are more readily obtainable without the necessity of performing endless welds to provide a suitable training and test set.

The random initiation of weights within the SOFM prior to training provides certain limitations in portability. Retraining the same network on the same data most certainly would not provide the same results. That is retraining an initial SOFM network would necessitate a complete re-test on all remaining data if the acquired activation maps are to remain valid. This would further necessitate the retraining of the MLP network as different areas of the activation maps would have been established. This, however, would not require the repitition of the welds. The already recorded sound data would still remain valid for that particular weld setup and only software modification in the form of network re-training would be necessary.

5.8 Proposed Developments

This research work has provided a basic method of monitoring arc sound to assess the welding process parameters and overall stability. As with any fundamental research project the development necessitates an iterative process.

The methods employed provide a large amount of scope for amendments and further optimisation. All areas, from the quantity and resolution of the input data, methods of transform, data compression techniques, size and configuration of neural network schemes to hardware implementation, require further consideration. All these parameters are controllable in some way and the large permutation of possible scenarios increases the chances of configuring a fully optimised system.

It is anticipated that priority should be given to the further investigation of the following areas.

5.8.1 Speed Optimisation

The optimisation of the response time has already been discussed to some extent in section 5.5.4. Of initial importance is the necessary time window to assess the process state. In terms of response times it is desirable to reduce it as far as possible. Iterative experiments would have to be carried out to analyse smaller and smaller time windows to establish the limits acceptable for the ANN system to differentiate between successive activation maps. The effect of reducing the time window would be similar to magnifying or 'zooming' in, which in Gestalt terms would mean losing the whole picture for the sake of potentially meaningless detail.

An interesting method to attempt would be the use of a rolling window for data acquisition. At present the system provides a predicted state based upon a fixed discrete time window. A rolling window would be relatively easy to implement and could provide a temporal aspect to the data interpretation as well as speeding up the overall process.

Further speed increases could be obtained from improved hardware devices such as ADC components and the use of machine code and/or assembler language within dedicated devices.

5.8.2 Hardware Implementations of Neural Systems

At present, there is a substantial lack of VLSI devices (utilising parallel configuration or otherwise) dedicated to ANN implementation. Within a commercial environment the use

of high speed super computers is not a viable option. Even the use of PC systems can create problems within the potentially hostile and volatile environment of high power welding equipment.

Industrial DSP systems, suitably shielded, can provide a robust platform on which to instigate on-line networks.

Although computationally DSP systems suit the application of ANNs, limitations in RAM can cause problems as trained networks, training and recall data can consume large amounts of memory. The present system requires 64 x 187 x 32 bit (float) data space for the SOFM, and (187 x 24 + 24 x 4) x 32 bit (float) for the BPN. This renders a data memory space of at least 66kB for the network weights alone. Further space is required for sine/cosine tables for the FFT routines as well as programming functions.

DSP systems such as the TMS320C32 & C40 and AT&T DSP32, are commercially available and provide suitable memory and speed for viable real time application. Speeds up to 270 Mflops can be achieved which offers the potential of a neural response time of under 1ms.

5.8.3 Real Time Control Prospects

Providing the time requirements and monitoring accuracy can be met, the use of audible sound emission within a closed loop control scheme can be substantiated. Its use though should be limited to a secondary monitoring status due to the accuracy. Within the present system the total accuracy is compromised by the generalising and noise supression aspects.

The monitoring of electrical parameters is more readily and accurately fulfilled by statistical and expert system methods. The monitoring of such parameters as stability and flux depth however, may be more viable and quicker through sound monitoring rather than electrical methods.

Closed loop control has its own inherent stability problems. Sound monitoring would not provide the stable inputs required to update the system confidently, at speed.

5.9 Implications For Weld Monitoring and Control

It has been shown that more robust and compact methods of weld process monitoring are required if quality standards are to be improved. The use of artificial neural network technology within the welding community is generally increasing as new methods are being tried and tested in many other areas.

Improved methods of monitoring are constantly sought to maintain weld production quality. As a consequence information concerning the process state is desirable from any conceivable avenue. Much research in the past has been afforded to the use of acoustic emissions due to the human use in manual welding. Limited success has highlighted the computationally intensive process required for the task.

The research work carried out here provides a further insight into the use of ANNs for weld monitoring and control. Positive correlations have been exhibited between the audible sound and the controllable electrical process parameters such as voltage, current, travel speed and their combined relationship of energy input. Furthermore, it has been shown possible to monitor quality indicators such as stability and flux coverage for the SAW process.

5.10 Recommendations for Further Research

The choice of SAW as the target process created some problems for data and process validation. However, this welding process requires further monitoring techniques due to the lack of visual information. This paradox should set the foundations for further research areas. Within this experimental work, limitations were experienced with the welding rig configuration. Certain process parameters have been proven to be indicative of transfer mechanisms, such as stand-off, electrode angle and flux depth, which in turn are responsible for stability and changes in process sound. Research should be afforded to transfer mechanisms and related process parameters in SAW to further explore the possibilities of audible sound in weld monitoring.

The success of the decribed method for the monitoring of the electrical parameters provides an indication that a direct prediction of the weld bead profile by the neural network could be possible. Research within this area could be justified by the already understood mathematical relationships that exist between the process parameters and bead profiles and dimensions.

Other areas of interest within the SAW process would be vibrational analysis. Although analysis of vibration within the parent material could be limited, weld pool oscillations, indications of transfer mechanisms and weld stability could be responsible for adverse vibrational changes at the welding head. This method could be especially useful to monitor arc strike which is notoriously transient and difficult to monitor. Similar methods of neural network application could be utilised within this task.

The portability of the present system to other welding processes is also worthy of investigation. Common production processes such as MIG would benefit from adaptable methods of stability monitoring, as well a providing a more visible and consequently an easier process to validate.

It is anticipated that changes in joint geometry, such as the production of fillet welds, would yield physical variations in audible sound emissions due to the change in reflective surfaces and weld bead profiles. Furthermore, changes in the constituents of the flux, the size and composition of the filler wire and the metallurgical properties of the parent material could affect the patterns found within the acoustic emission. It is recommended that all of these aspects should be afforded substantial research to provide further development and validation of the neural network approach.

6.0 CONCLUSION

6.0 CONCLUSIONS

1. Current literature has shown that submerged arc welding presents validation problems concerning material transfer mechanism due to lack of viable visual assessment and documented research evidence.

2. The transient sound pressure levels associated with submerged arc welding can be used as an on-line monitoring medium.

3. Sound within the frequency domain can provide information concerning the submerged arc welding process state. However, Gestalt principles dictate that individual fast fourier transform frames provide little information.

4. The temporal differences of audible sound are important in determining the process state of submerged arc welding. That is, the *changes* in sound emission with time is a crucial element in determining the process state.

5. Existing literature justifies the use of artificial neural networks as a biologically plausible method of replicating human auditory processes.

6. The use of self organising feature maps provides a method of data compression which in turn can suppress transient noise encountered within arc welding working environments.

7. Full replication of biological auditory processes is hindered by limited memory capacity and speed of current PC systems. Massively parallel systems are required if an acceptable emulation is to be achieved.

8. Neural Networks exhibit sufficient robustness to process transient sound data indicative of arc noise.

9. Artificial neural networks offer a fast alternative to statistical methods of weld stability assessment.

10. The accuracy of neural networks for assessing weld process parameters via audible acoustic emissions is limited. Existing statistical methods of direct electrical measurement and analysis currently provides a more reliable and accurate method.

11. Current digital signal processing platforms provide a suitable real time hardware system for artificial neural network implementation.

12. Research literature has shown that optimised source code in assembler, instigated within a TMS320C32 digital signal processing system, could adequately provide the necessary sampling and response speed increase (for the described method) to allow acceptable real time monitoring.

7.0 REFERENCES

REFERENCES

Aleksander, I., Burnett, P., 1983, "Reinventing Man, The Robot Becomes Reality", Kogan Page Ltd, 1983.

Becken, O., 1969, "Metal transfer from welding electrodes", IIW Doc 212-179-69, 1969.

Bellows, J.C., *et al.*, 1994, "Automated Rotor Welding Processes Using Neural Networks", United States Patent, No 5283418, Feb. 1st 1994.

Beranek, L.L., 1986, "Acoustics", American Institute of Physics, Acoustical Society of America, 1986, pp 165

Berger, W., *et al.*, 1994, "WELCOME - a knowledge based system for the selection of consumables", Paper 20, TWI, Computer Technology in Welding, Conf. Proc., Paris, France, 15-16th June 1994.

Bishop, C.M., 1994, "Bayes for Beginners", NCRG 4321, Neural Computing Research Group, Aston University, presented at NCAF, Oxford, UK, June 1994

Blumschein, E., 1994, "Acoustic Tuning Versus Predetermination of Parameters for Short Arc Welding", TWI, Computer Technology in Welding, Paper 46, Paris, France , 1994.

Bogart, B.P., *et al.*, 1963, "The queferency analysis of time series for echoes: cepstrum, pseudo-autocovariance, cross spectrum and saphe cracking", Time Series Analysis, edited by M. Rosenblatt, New York: Wiley, chap. 15, pp. 209-243, 1963.

Boilot, J.P., *et al.*, 1994, "Intelligent vision brings a new generation of welding robots to light", Paper 32, TWI, Computer Technology in Welding, Paris, France, 1994.

Bregman, A. S, 1990, "Auditory Scene Analysis: The Perceptual Organization of Sound", MIT Press, 1990.

Brunel University, 1993, U.K. Patent 9307096.9 , Brunel University, U.K., priority file date April 1993.

Burley, I., 1991, "It's a fuzzy old world", Practical Electronics, October 1991.

Carlson, N.R., 1984, "Psychology, the science of behaviour", 3rd Edition, Allyn & Bacon Publishing, 1984.

Carpenter, G.A, Grossberg, S., 1988, "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network", computer, pp. 77-88, March 1988.

Chawla, K., 1993, "Objective on line assessment of the performance of flux cored wires by real time computer based monitoring", Ph.D Thesis, Cranfield Institute of Technology, UK, 1993.

Chawla, K., Norrish, J., 1992, "Real time monitoring using analysis of the arc sound", TWI, Computer Technology in Welding, Cambridge, UK, 1992.

Childers, D.G., *et al.*, 1977, "The Cepstrum: A Guide to Processing", Proc. IEEE, Vol. 65, No. 10, pp. 1428-1443, Oct. 1977

Cottrell, G.W, *et al.*, 1987, "Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming", 9th ANN. Conf. of the Cognitive Science Society, 1987.

Cover, T.M., 1965, "Geometrical and Statistical Properties of Systems of Linear Inequalities With Applications in Pattern Recognition", IEEE Trans. on 'Electronic Pattern Recognition', EC-14, No. 3, pp. 326-334, 1965.

Crichton, A. B., *et al.*, 1978, "Gas Metal Arc Welding", ch. 4 of Welding Handbook, Welding Processes-Arc and Gas Welding and Cutting, Brazing and Soldering, 7th Edition, Vol. 2, American Welding Society, Macmillan Press Ltd., 1978.

Cybenko, G., 1989, "Approximation by Superpositions of a Sigmoidal Function", Mathematics of Control, Signals and Systems, No. 2, pp. 303-314, 1989.

Dai, H., Wright, B. A., 1995, "Detecting signals of unexpected or uncertain durations", Journal of the Acoustical Society of America, Vol. 98, No. 2, Pt. 1, pp. 798-805, 1995.

Daubechies, I., 1990, "The Wavelet Transform, Time Frequency Localisation and Signal Analysis", IEEE Transactions on Information Theory, Vol. 36, No. 5 p. 961, Sept 1990.

Dayhoff, J.E., 1991, "Neural Network Architectures, an introduction", Van Nostrand Reinhold, 1991.

DeSieno, D., 1988, "Adding a Conscience to Competitive Learning", IEEE Proc. Int. Conf. on Neural Networks, New York, Vol. 1, pp. 117-124, 1988.

Dilthey, U., *et al.*, 1992, "Trainable Arc Sensor for Varying Welding Joint Geometries - Use of Artificial Neural Networks", IIW Doc. XII-1294-92, Madrid, pp. 147-155, 1992.

Doumanidis, C. C., 1994, "Multiplexed and Distributed Control of Automated Welding", Control Systems, Vol. 14, No. 4, IEEE, pp. 13-24, August 1994.

Dubes, R., Jain, A.K.,1979, "Clustering Techniques: the User's Dilemma", Pattern Recognition 8, No 4, 1979, pp. 247 - 260.

Durlacher, C., 1994, "The Butchers Blade", Programme 1 of "White Heat", BBC Television, BBC2, (5/9/94), Uden Associates Ltd, London, 1994.

Eberhart, R.C., Dobbins, R.W, 1990, " Neural Network PC Tools, a practical guide", Academic Press Inc., 1990.

Erdmann-Jesnitzer, F., *et al.*, 1966, "Acoustic Investigations of the Welding Arc", Report Document 212-86-66, Institute of Metals, TH Hannover.

Everest, F. A., 1981, "The Master Handbook of Acoustics", TAB BOOKS Inc., 1981.

Fallick, J.C., *et al.*, 1978, "Shielded Metal Arc Welding", ch. 2 of Welding Handbook, Welding Processes-Arc and Gas Welding and Cutting, Brazing and Soldering, 7th Edition, Vol. 2, American Welding Society, Macmillan Press Ltd., 1978.

Ferret, E., 1993, "Improving the Neural Network Testing Process", Adaptive Intelligent Systems, S.W.I.F.T (ed), Elsevier Science Publishers, 1993, pp. 193-204.

Funk & Rieber, 1985, "Handbook of Welding", Breton Publications, 1985.

Garth, S.C.J., 1992, "To simulate or not to simulate...", ICANN'92, Brighton, U.K., Sept. 1992, Vol. 2, pp. 1397 - 1403.

George, F. H., 1977, "The Foundations of Cybernetics", Gordon and Breach Science Publishers, ISBN 0 677 05340 1, 1977.

Gorman, R.P., Sejnowski, T.J., 1988, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets", Neural Networks 1, pp. 75-89, 1988.

Goser, K., 1991, "Kohonen's Maps -Their Application and Implementation in Microelectronics", ICANN'91, Espoo, Finland, Jun. 1991, Vol. 1, pp. 703 - 708.

Green, N.P.O, *et al*, 1986, "Biological Science, Organisms, Energy and Environment", Cambridge University press, ISBN 0-521-28407-4, 1986.

Harris, T.J., 1992, "Neural Network characterisation of ultrasonic data and its use in the control of submerged arc welding", Ph.D Thesis, Brunel University, UK, 1992.

Harris, T.J., 1993, "Neural Networks & COMADEM", 5th Int. Conf. COMADEM, Bristol, 1993.

Harris, T.J., Wilkinson, P.D., 1993, "The Development of an Automated NDT System using Neural Networks", JOM 6, Helsingor , Denmark, April 1993, pp. 549 - 554.

Harris, T.J., Wilkinson, P.D., 1994, "The further development of an automated NDT system using neural networks", TWI, Computer Technology in Welding, Paris, France, June 1994, paper 26.

Hecht-Nielson, R., 1991, "Neurocomputing", Addison-Wesley Publishing Company, 1991.

Herraty, A.G., 1993, "Bearing Vibration - Failures and Diagnosis", Mining Technology Journal, Feb. 1993.

Hertz, J., *et al.*, 1991, "Introduction to the Theory of Neural Computation", ISBN 0-201-51560-1, Addison Wesley, 1991

Houldcroft, P.T., 1977, "Welding Process Technology", Cambridge University Press, 1977.

Houldcroft, P.T., 1989, "Submerged Arc Welding", Abington Pub, 2nd Edition, 1989.

Jackendoff, R., 1993, "Patterns in the Mind: Language and Human Nature", Harvester Wheatsheaf, (ISBN 0-7450-0963-8), 1993.

Javed, M.A., *et al.*, 1992, "Modern trends in condition monitoring technology", COMADEM '92, Senlis, Paris, July 1992, pp. 357-362.

Jones, J., Evans, J., 1992, "Artificial Neural Networks for Welding Applications", IIW Doc. XII-1294-92, Madrid, pp. 101-108, 1992.

Jubak, J., 1992, "In The Image Of The Brain", Littlebrown & Co. Publishing, 1992.

Kallio, K., *et al.*, 1991, "Classification of Lung Sounds by Using Self Organizing Feature Maps", ICANN'91, Espoo, Finland, June 1991, Vol. 1, pp., 803 - 808.

Kangas, J.A., Kohonen, T.K., 1990, "Variant of Self Organizing Maps", IEEE Trans. Neural Networks, Vol. 1, No 1, March 1990, p. 93.

Kaneko, Y., *et al.*, 1993, " Neural Networks and fuzzy control in welding robots using CCD camera and touch sensor", Proc. Int. Joint Conference on Neural Networks (IJCNN) 1993. Vol. 1, pp. 677 - 680

Kaskinen, P., *et al.*, 1986, "Acoustic Arc Length Control", Advances in Welding Technology and Science, TWR'86, pp. 763-765.

Kasslin, M., *et al.*, 1992, "Process State Monitoring using Self Organizing Maps", ICANN'92, Brighton U.K., Sept. 1992, Vol. 2, pp. 1531 -1534.

Kirk, R.E, 1990, "Statistics an Introduction", 3rd Edition, Holt Rinehart Winston, ISBN 0-03-020424-0, 1990.

Kohonen, T.K., Oja, E., 1976, "Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements", Biological Cybernetics, No.22, pp. 85-95, 1976.

Kohonen, T.K., 1988, "The 'Neural' Phonetic Typewriter", IEEE Computer, March 1988, pp. 11 -22.

Kohonen, T.K., 1992, " A Tutorial on the Optimisation os Self Organising Maps", ICANN, Brighton, UK, 4th-7th September 1992

Kosko, B., 1994, "Fuzzy Thinking", Harpins Collins Publishers, 1994.

Kralj, V., Tušek, J., 1988, "Some Findings and Characteristics About the Material Transfer in the Submerged Arc Welding with Parallel Wires", IIW Document 212-695-88, 1988.

Lancaster, J.F., 1979, "Metal transfer in fusion welding", Welding Institute conference on Arc Physics and Weld Pool behaviour, May, 1979.

Lancaster, J.F., 1984, "The Physics of Welding", produced by IIW, Pergamon Press, 1984.

Lapedes, A., Farber, R., 1987, "Non-Linear Signal Processing using Neural Networks: Prediction and System Modelling", Los Alamos National Laboratory Report, LA-UR-87-2662, 1987.

Larson, L.O., *et al.*, 1993, "Laser scanners as a measuring device and its application in arc welding", JOM-6, Helsingor, Denmark, 1993, p. 44.

Lazzaro., J, Mead, C., 1996, "A Silicon Model of auditory Localisation", available from http://www.pcmp.caltech.ede/anaprose/lazzaro/bib.html, 1996.

Le Lionnais, F., 1966, "Cybernetics", Ch. 9 of "Science in the Twentieth Century", edited by R. Taton, Thames & Hudson Ltd., 1966.

Leonard, J.A., Kramer, M.A., 1991, "Radial Basis Functions for Classifying Process Faults", IEEE Control Systems, April 1991.

Li, P., *et al.*, 1994, "A neural controller for TIG welding parameter generation", TWI, Computer Technology in Welding, Paris, France, June 1994, paper 22.

Lindsay, P.H., Norman, D.A., 1977, "Human Information Processing", 2nd Edition, HBJ Publishing, 1977.

Lynn, P. A., 1977, "An Introduction to the Analysis and Processing of Signals", The Macmillan Press Ltd., ISBN 333 14353 1, 1977.

McCord Nelson, M., Illingworth, W.T., 1991, "A Practical Guide to Neural Nets", ISBN 0-201-52376-0, Addison Wesley Publishing Co. Inc., 1991.

McCulloch, W.C., Pitts, W., 1943, "A Logical Calculus of the Ideas Immanent in Nervous Activity", Bulletin of Mathematical Biophysics, No. 5, pp. 115-133, 1943.

MacIntyre, J., 1993, "Condition Monitoring and Artificial Intelligence : A Literature Review", Internal paper for Blythe Power Station, National Power, Sept. 1993

Mao, Y.L., *et al.*, 1993, "Real-Time Fast Fourier Transform Analysis of Acoustic Emission During CO_2 Laser Welding of Materials", Journal of Laser Applications, Vol. 5, No 2 & 3, Autumn 1993.

Martin, J.D., 1991, " Signals and Processes, a foundation course", ISBN 0-273-03256-9 Pitman, 1991.

Martin, K.F., 1994, "A Review by Discussion of Condition Monitoring and Fault Diagnosis in Machine Tools", Int. Jnt. Machine Tools Manufacture, Vol. 34, No. 4, 1994, pp. 527-551

Martin, P.J., Bahrani, A.S., 1993, "Neural Networks for Welding Applications", JOM 6, Helsingor , Denmark, April 1993, pp. 89 - 95.

Marvin, C., Ewers, G., 1994, "A Simple Approach to Digital Signal Processing", Texas Instruments, TI Mentors series, ISBN 0-904 047-00-8, 1994.

Massaro, D.W., 1989, "Experimental Psychology, an information processing approach.", HBJ Publishing, 1989.

Matteson, M.A., *et al.*, 1992, "An Optimal Artificial Neural Network for GMAW Arc Acoustic Classification", 3rd Int. Conf. Trends in Welding Research, USA, June 1992.

Mead, C., 1989, "Analog VLSI and Neural Systems", Addison-Wesley, Reading MA, 1989.

Merken, M., 1989, " Physical Science with Modern Applications", 4th edition, Saunders College Publishing, 1989.

Minsky, M., Papert, S., 1969, "Perceptrons", MIT Press, 1969.

Murre, J.M.J., 1991, "Transputer Implementations of Neural Networks: An Analysis", ICANN'91, Espoo, Finland, Jun. 1991, Vol. 2, pp. 1537 - 1540.

Musliner, D. J., *et al.*,1995, "The Challenges of Real-Time AI", Computer, IEEE, pp. 58-66, January 1995.

Nadler, M., Smith, E. P., 1993, "Pattern Recognition Engineering", John Wiley & Sons Inc., ISBN 0-471-62293-1, 1993.

Neumann, J. Von, 1958, "The Computer and the Brain", Yale University Press, 1958. reprinted in Farris & Fadiman, "The World Treasury of Physics, Astronomy and Mathematics", Little, Brown & Co., 1991.

NeuralWare, NeuralWorks Professional II (Anon), 1993, "Neural Computing", NeuralWareInc.,1993.

Newland, O.E., 1993, "An Introduction to Random Vibrations, Spectral and Wavelet Analysis", 3rd Edition, Longman Scientific and Technical, ISBN 0582-21586-6, 1993.

Nishar, D.V., *et al.*, 1994, "Adaptive control of temperature in arc welding", Control Systems, IEEE, vol. 14 No 4, August 1994.

Norrish, J., Richardson, I.M., 1988, "Metal transfer mechanisms", Welding and Metal Fabrication, Jan. / Feb., 1988.

O'Brien, J.C, Macintyre, J., 1994, "Wavelets: an alternative to Fourier analysis", Proceeding of a seminar on "Vibration in Fluid Machinery" (ISBN 0852 989369), IMechE, London, 1994, pp. 75-86.

O'Mard, L.P., *et al.*, 1996, "LUTEar: Core Routines Library. A Flexible Auditory Simulation Development Computing System", Speech and Hearing Laboratory, Loughborough University of Technology, UK, 1996.

Ogunbiyi, B., *et al.*, 1993, "Improved Manufacturing in Welding by Innovative Use of Neural Networks in Real Time", Technical Specification, Brite EuRam Project No. 7918, 1993.

Ogunbiyi, B., 1995, "Control Models and Algorithms", working paper, Brite EuRam project No.7918, Improved Manufacturing in Welding by Innovative Use of Neural Networks in Real Time, 1995.

Ogunbiyi, B., Norrish, J., 1996, "GMAW Metal Transfer and Arc Stability Assessment Using Monitoring Indices", Paper 11, TWI'96, Computer Technology in Welding, Lanaken, Belgium, 9-12th June, 1996.

Ohshima, K., *et al.*, 1993, "Neuro-fuzzy control in pulsed MIG welding", IIW DOC XII 1330- 93, pp. 188-193, 1993.

Owen, F., Jones, R., 1992, "Statistics", 3rd Edition, Pitman Publishing, ISBN 0-273-03194-5, 1992.

Pal, N.R., Bezdek, J.C., 1993, "Extensions of Self-Organizing Feature Maps for Improved Visual Displays", IJCNN'93, Nagoya, Japan, Vol. 3, pp. 2441-2447, Oct. 1993.

Palotas, B., 1992, "Computer determination of welding parameters for arc welding processes", Paper 15, TWI, Computer Technology in Welding, Conf. Proc., Cambridge, UK, 3-4th June, 1992.

Papamichalis, P.E., 1990, "Digital Signal Processing Applications with the TMS320 Family", Vol. 3, Prentice Hall & Digital Signal Processing Series, Texas Instruments, 1990.

Paton, A., 1994, "The development of advanced image based sensors for automated TIG welding", Paper 21, TWI, Computer Technology in Welding, Paris, France, 1994.

Pecas, P., Quintino, L., 1993, "State of the art on the use of expert systems in welding", IIW DOC, XII-1329-93, pp. 179-186, 1993.

Pilous, V., *et al.*, 1987, "Acoustic Emission in Automatic Submerged Arc Surfacing with an Austenitic Strip Electrode", Kovove Materialy, Vol. 25, No 4, pp. 468-481, 1987.

Press, W.H., *et al.*, 1992, "Numerical Recipes in C : the art of scientific computing", 2nd Edition, Cambridge University Press, ISBN 0-521-43108-5, 1992

Prout , J.H., Bienvenue, G.R., 1990, " Acoustics for You", Robert. E. Krieger Publishing Company, 1990.

Quintino, L., Pecas, P., 1993, "Welding automation in ship building", JOM-6, Helsingor, Denmark, April 1993, pp. 10-21.

Robinson, D.W., Dadson, R.S., 1956, "A redetermination of the equal-loudness relations for pure tones", British Journal of Applied Physics, No.7, pp. 166-181, 1956.

Rolan-Mieszkowski, M., 1993, "Common Misconceptions About Hearing", Canadian Acoustics, Vol. 21, No 1, March 1993.

Ross, J., Houtsma, A. J. M., 1994, "Discrimination of auditory temporal patterns", Perception & Psychophysics, vol. 56, No. 1, pp. 19-26, 1994.

Rumelhart, D.E., McClelland, J.L., 1986, "Parallel Distributed Processing", Vols. 1&2, MIT Press, 1986.

Scharf, B., 1970, "Critical Bands", Foundations of Modern Auditory Theory, Tobias, J.V. (Ed.), Vol. 1, New York Academic Press, 1970.

Schlebeck, E., 1982, "Some Investigation of acoustic Emission at MIAB Welding", IIW Working Group 212, December 1982.

Shannon, C.E., 1949, "Communications in the Presence of Noise", Proceedings of the IRE, Vol. 37, pp. 10-21, January 1949.

Shaw, H.F., Lucas, W., 1994, "The potential of neural networks in welding", TWI, Computer Technology in Welding, Paris, France, June 1994, paper 52.

Siores, E., 1992, "Hybrid intelligent systems for real time welding process control", TWI, Computer Technology in Welding, Cambridge, UK, June 1992, paper 14.

Slaney, M., 1996, "Cochlear and Auditory Models in Matlab", available from ftp.apple.com/pub/malcolm, 1994. (Still valid November 1996)

Stearns, S.D., Hush, D.R., 1990, "Digital Signal Analysis", 2nd Edition, Prentice Hall International Editions, 1990.

Stroud, R.R., 1982, "Measurement and autonomous penetration control during submerged arc welding of mild steel", Ph.D Thesis, Brunel University, 1982.

Stroud, R.R., *et al.*, 1991, "Neural Networks in automated weld control", TWI, Gateshead, UK, 1991.

Stroud, R.R., Swallow, S., 1992, "Neural Networks for real time signal processing and control in robotised welding", Paper 192, Proc. 2nd IASTED, Alexandria, Egypt, 1992.

Stroud, R.R, McCardle, J.R, *et al.*, 1996, "Improved Manufacturing in Welding by Innovative Use of Neural Networks in Real Time (WINNER Brite EuRam Project No. 7918)", Paper 31, TWI'96, Computer Technology in Welding, Lanaken, Belgium, 9-12th June, 1996.

Swallow, S.S., 1995, "Improved Manufacturing in Welding by Innovative Use of Neural Networks in Real Time", Brite EuRam Project No. 7918, Contract No. BRE-2.CT93.0601, 1993. Month 18 Progress Report, July 1995.

Taylor, W.A., 1988, "What every engineer should know about AI", MIT Press, 1988.

Texas Instruments, 1994, "TMS320CX Users Guide", Texas Instruments Inc. 2558539-9721 revision J, October 1994.

Texas Instruments, 1996, "Integration", European Edition, Vol. 4, No. 5, pp. 1-2, July 1996.

Tomberg, J., Kaski, K., 1991, "Digital VLSI Architecture of Backpropagation Algorithm with On-Chip Learning", ICANN'91, Espoo, Finland, Jun. 1991, Vol. 2, pp. 1561 - 1564

Turing, A., 1950, "Can a machine think ?", Mind, Vol. 59, Oxford University Press, 1950. reprinted in Farris & Fadiman, "The World Treasury of Physics, Astronomy and Mathematics", Little, Brown & Co., 1991.

Van Adrichem, T. J., 1966, "Metal Transfer in Submerged Arc Welding", IIW Document 212-78-66, 1966.

Vu, V.V., *et al.*, 1991, "Time Encoded Matrices as Input Data to Artificial Neural Networks for Condition Monitoring Applications", COMADEM '91, Southampton, UK, 1991.

Wamack, N. S., *et al.*, 1978, "Submerged Arc Welding", ch. 6 of Welding Handbook, Welding Processes-Arc and Gas Welding and Cutting, Brazing and Soldering, 7th Edition, Vol. 2, American Welding Society, Macmillan Press Ltd., 1978.

Wang, K., Shamma, S., 1995, "Auditory Analysis of Spectro-temporal Information in Acoustic Signals", IEEE Engineering in Medicine and Biology, pp. 186-194, Mar/Apr 1995.

Wasserman, P. D., 1989, "Neural Computing Theory and Practice", Van Nostrand Reinhold, ISBN 0- 442-20743-3, 1989.

Welch, N., 1996, "Auditory Perception", McGill University, Cal., USA, Internet Web site, http://www.music.ca/auditory/Auditory.html, 1996.

Wheddon, C., Linggard, R., 1990, "Speech and Language Processing", Chapman & Hall, ISBN 0 412 378000 0,1990.

Widrow, B., Hoff, M., 1960, "Adaptive Switching Circuits", 1960 IRE WESCON Convention Record, Pt. 4, pp. 96-104, 1960.

Widrow, B., Glover, J.R, *et al.*, 1975, "Adaptive Noise Cancelling: Principles and Applications", Proc. IEEE, Vol. 63, No. 12, 1975

Wiener, N., 1961, "Cybernetics", MIT Press, Cambridge, Mass. USA, 1961.

'WINNER' Consortium, 1993, "Improved Manufacturing in Welding by Innovative Use of Neural Networks in Real Time", Brite EuRam Project No. 7918, Contract No. BRE-2.CT93.0601, 1993.

Zadeh, L.A., 1965, "Fuzzy Sets", Information and Control No. 8, pp., 338-353, 1965.

Zhang, X., Li, Y., 1993, "Self-Organising Map as a New Method for Clustering and Data Analysis", Proc. IJCNN'93, Vol. 3, pp. 2448-2451, Nagoya, Japan, 25th-29th Oct., 1993.

8.0 TABLES AND FIGURES

Designation of Transfer Type	Welding Processes (examples)
1. Free flight transfer	
1.1 Globular	
1.1.1 Drop	Low -current GMA
1.1.2 Repelled	CO2 shielded GMA
1.2 Spray	
1.2.1 Projected	Intermediate current GMA
1.2.2 Streaming	Medium current GMA
1.2.3 Rotating	High current GMA
1.3 Explosive	SMA (coated electrodes)
2. Bridging transfer	
2.1 Short-circuiting	Short-arc GMA
2.2 Bridging without interruption	Welding with filler wire addition
3. Slag-protected transfer	
3.1 Flux wall guided	SAW
3.2 Other modes	SMA, cored wire, electroslag

TABLE 2.1
Material Transfer Mechanisms for Arc Fusion Welding
(IIW Classification)

Quality	Parameters	Control	Model	Classification	Frequency
Metal Transfer Mode	I, V, Sp (Transient)	I, V	ANN (Stat.)	Mode + Confidence	Online (10Hz)
Stability	I, V, Sp (Transient)	I, V	ANN (Stat.)	Stable/Unstable	Online (10Hz)
Spatter	I, V ,Sp (Transient)	I, V	ANN (Stat.)	Spattery / Unspattery	Online (10Hz)
Arc Ignition	I, V, Sp,Wfr (Transient)	I, V,Wfr	ANN (Ref. US Stat.)	Good Ignition / Bad Ignition	Online (100Hz)
Bead Geometry	(Averaged Data)		ANN (Stat.)		
Leg Length	I, V, Ts Standoff, Sp		ANN (Stat.)	Measurement + Confidence	4Hz
Throat	I, V, Ts Standoff, Sp		ANN (Stat.)	Measurement + Confidence	4Hz
Side Wall Fusion (Fillet Welds)	I, V, Ts Standoff, Sp		ANN (Stat.)	Measurement + Confidence	4Hz
Undercut	I, V, Ts Standoff, Sp		ANN (Stat.)	Undercut/No undercut	4Hz
Burnthro'	I, V, Ts Standoff, Sp, Video		ANN (Stat.)	Burnthrough/No Burnthrough	4Hz
Standoff	I, V, Video	Z	ANN (Stat.)	Measurement + Confidence	4Hz
Gap Width	I, V, Sp, Video	I, Ts, Z, V	ANN	Measurement + Confidence	4Hz
Seam Track	Video	X, Ts, Z, V	ANN	Offset + Confidence	4Hz
Weld Pool Size	Video	I, Ts	ANN	Width and Area	4Hz
Tip Wear	Trend Electrical Analysis		ANN	Tip Aperture size	Off Line

Key

I - Current
V - Voltage
Ts - Travel Speed
Sp -Sound Pressure Level
Wfr - Wire Feed Rate

TABLE 2.2
Welding Control Objectives (as defined by theWINNER consortium, Stroud *et al.*, 1996)

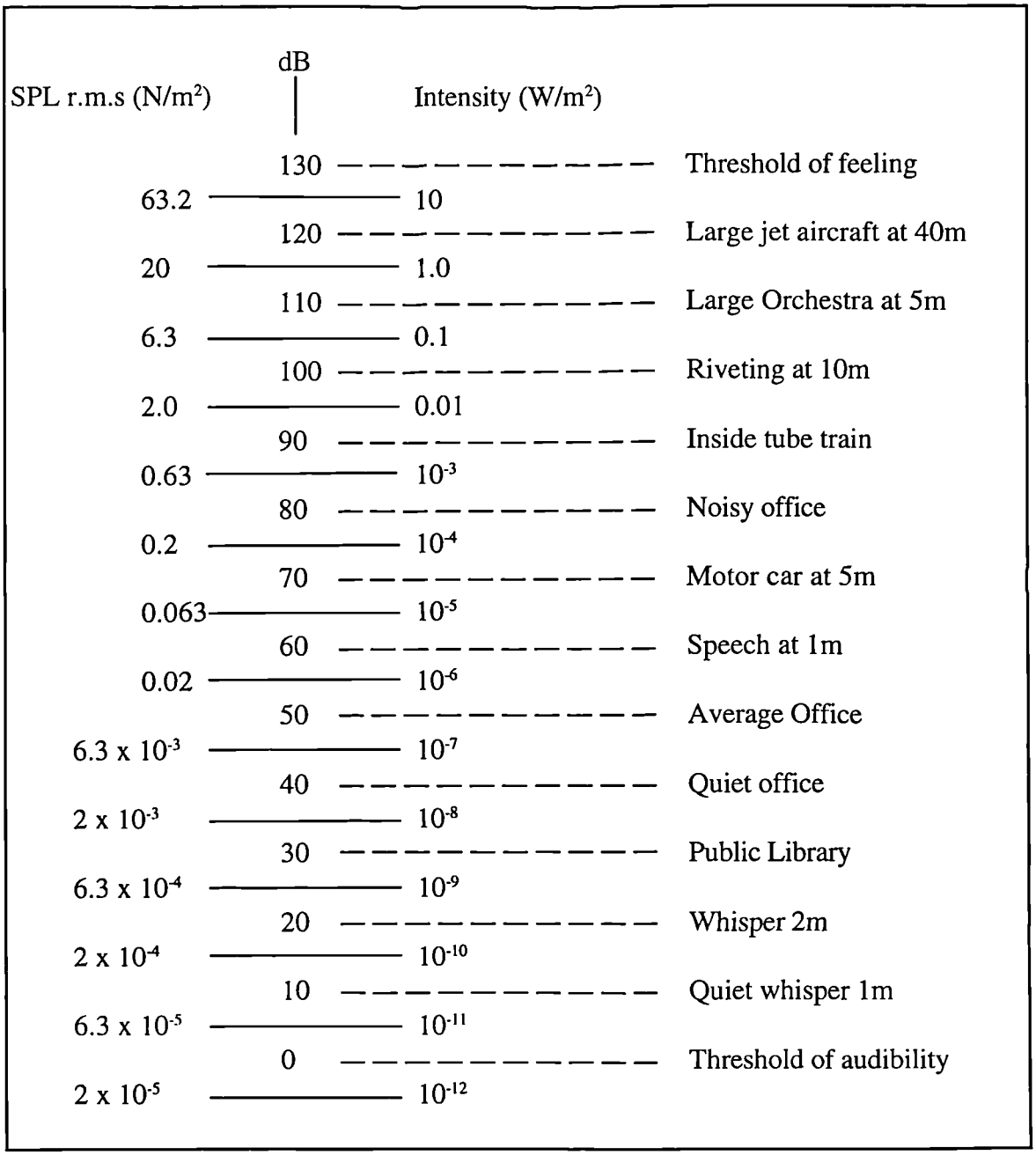


TABLE 2.3
Relative Sound Pressures and Intensity Levels

Input Frequency (Hz)	Infra Low pass	Gain (dB)	15Hz - 6kHz	Gain (dB)	6kHz - 8kHz	Gain (dB)	8kHz - 9kHz	Gain (dB)	9kHz - 10kHz	Gain (dB)	10kHz - 11kHz	Gain (dB)	11kHz - 12 kHz	Gain (dB)	12kHz - 14kHz	Gain (dB)	14kHz - 20 kHz	Gain (dB)	Ultra High pass	Gain (dB)
1	2.7	-0.915149811																		
2	2.7	-0.915149811																		
3	2.7	-0.915149811																		
4	2.7	-0.915149811																		
5	2.7	-0.915149811	0.15	-26.0205999																
6	2.4	-1.93820026	0.2	-23.5218252																
7	2.4	-1.93820026	0.3	-20																
8	2.34	-2.158107946	0.38	-17.9467532																
9	2.22	-2.615365605	0.38	-17.9467532																
10	2.1	-3.0980392	0.5	-15.563025																
11	2.04	-3.349821746	0.7	-12.6404643																
12	1.92	-3.87640052	0.78	-11.700533																
13	1.86	-4.15216621	0.78	-11.700533																
14	1.8	-4.436974992	0.82	-11.266148																
15	1.74	-4.731440129	0.9	-10.4575749																
16	1.68	-5.03623946	1	-9.54242509																
17	1.56	-5.679933127	0.9	-10.4575749																
18	1.4	-6.619864381	1.2	-7.95880017																
19	1.37	-6.808013751	1.2	-7.95880017																
20	1.32	-7.13064647	1.25	-7.60422483																
22	1.2	-7.958800173	1.25	-7.60422483																
28	0.9	-10.45757491	1.3	-7.26355805																
30	0.72	-12.39577517	1.5	-6.0205999																
40	0.54	-14.8945499	2	-3.52182518																
50	0.3	-20	2.22	-2.61536561																
60	0.25	-21.58362492	2.22	-2.61536561																
70	0.15	-26.0205999	2.22	-2.61536561																
80	0.13	-27.26355805	2.22	-2.61536561																
90	0.11	-28.71457139	2.22	-2.61536561																
100	0.08	-31.48062535	2.22	-2.61536561																
200			2.22	-2.61536561																
500			2.22	-2.61536561																
1000			2.22	-2.61536561																
2000			2.22	-2.61536561																
3000			1.92	-3.87640052																
3200			1.85	-4.19899053	0.1	-29.542425														
4000			1.8	-4.43697499	0.5	-15.563025														
4500			1.7	-4.93344667	1.5	-6.0205999														
5000			1.68	-5.03623946	2	-3.5218252														
5500			1.23	-7.74432287	2	-3.5218252														
5890			1.2	-7.95880017	2	-3.5218252														
6000			1.15	-8.32846829	2	-3.5218252	0.1	-29.542425												
6500			0.9	-10.4575749	1.8	-4.436975	0.5	-15.563025												
7000			0.9	-10.4575749	1.5	-6.0205999	0.7	-12.640464												
7560			0.8	-11.4806254	1.4	-6.6198644	1.5	-6.0205999												
7890			0.7	-12.6404643	1	-9.5424251	1.8	-4.436975												
8000			0.6	-13.9794001	0.8	-11.480625	2	-3.5218252												
8290			0.6	-13.9794001	0.75	-12.0412	2	-3.5218252	0.2	-23.5218252										
8500			0.55	-14.7351713	0.5	-15.563025	2.14	-2.9341496	0.6	-13.9794001										
9000			0.5	-15.563025	0.5	-15.563025	1.7	-4.9334467	1.2	-7.95880017	0.12	-27.95880017								
9200			0.48	-15.9176003	0.3	-20	1.4	-6.6198644	1.8	-4.43697499	0.4	-17.50122527								
9500			0.45	-16.4781748	0.2	-23.521825	1	-9.5424251	2.2	-2.69397148	1.8	-4.436974992								
9700			0.42	-17.0774393	0.1	-29.542425	0.8	-11.480625	2	-3.52182518	2.4	-1.93820026								
10000			0.4	-17.5012253	0.05	-35.563025	0.6	-13.9794	2	-3.52182518	2.2	-2.693971478	0.5	-15.56302501						
10200			0.3	-20			0.5	-15.563025	1.7	-4.93344667	2	-3.521825181	0.5	-15.56302501						
10500			0.2	-23.5218252			0.5	-15.563025	1.3	-7.26355805	2	-3.521825181	1.5	-6.020599913	0.12	-27.95880017			0.05	-35.56303
10700			0.1	-29.5424251			0.3	-20	1.3	-7.26355805	1.8	-4.436974992	1.8	-4.436974992	0.44	-16.67337156			0.05	-35.56303
11000							0.2	-23.521825	1	-9.54242509	1.5	-6.020599913	2.2	-2.693971478	0.75	-12.04119983			0.1	-29.54243
11200							0.18	-24.436975	0.8	-11.4806254	1.4	-6.619864381	2.2	-2.693971478	0.9	-10.45757491	0.2	-23.52182518	0.1	-29.54243
11500							0.15	-26.0206	0.75	-12.0411998	1.2	-7.958800173	2.1	-3.0980392	1.5	-6.020599913	0.5	-15.56302501	0.14	-26.61986
11700							0.1	-29.542425	0.5	-15.563025	0.8	-11.48062535	2	-3.521825181	1.9	-3.967353075	0.85	-10.95404658	0.22	-22.69397
12000									0.5	-15.563025	0.75	-12.04119983	2	-3.521825181	2.1	-3.0980392	0.9	-10.45757491	0.2	-23.52183

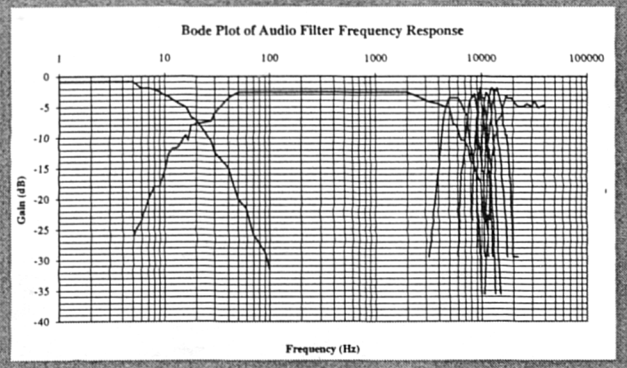


Table 3.1
Audio Filter Frequency Response (figures in white columns represent signal amplitude in Volts)

Input Frequency (Hz)	Infra Low pass	Gain (dB)	15Hz - 6kHz	Gain (dB)	6kHz - 8kHz	Gain (dB)	8kHz - 9kHz	Gain (dB)	9kHz - 10kHz	Gain (dB)	10kHz - 11kHz	Gain (dB)	11kHz - 12 kHz	Gain (dB)	12kHz - 14kHz	Gain (dB)	14kHz - 20 kHz	Gain (dB)	Ultra High pass	Gain (dB)
12200									0.35	-18.6610642	0.62	-13.6945913	1.8	-4.436974992	2.3	-2.307868374	1.2	-7.958800173	0.4	-17.50123
12500									0.16	-25.4600254	0.48	-15.91760035	1.4	-6.619864381	2.4	-1.93820026	1.35	-6.935749724	0.45	-16.47817
12700									0.1	-29.5424251	0.32	-19.43942553	1.35	-6.935749724	2.4	-1.93820026	1.8	-4.436974992	0.65	-13.28416
13000									0.1	-29.5424251	0.2	-23.52182518	1	-9.542425094	2.3	-2.307868374	1.95	-3.741732867	0.75	-12.0412
13200											0.1	-29.54242509	0.8	-11.48062535	2	-3.521825181	2.1	-3.0980392	0.8	-11.48063
13500											0.05	-35.56302501	0.8	-11.48062535	1.7	-4.933446667	2.3	-2.307868374	0.9	-10.45757
13700													0.5	-15.56302501	1.45	-6.31506505	2.3	-2.307868374	1	-9.542425
14000													0.25	-21.58362492	1.2	-7.958800173	2.4	-1.93820026	1	-9.542425
14500													0.1	-29.54242509	1	-9.542425094	2.2	-2.693971478	1.32	-7.130946
15000													0.05	-35.56302501	0.75	-12.04119983	1.9	-3.967353075	1.4	-6.619864
15500															0.56	-14.57866455	1.75	-4.681664121	1.45	-6.315065
16000															0.39	-17.72113295	1.5	-6.020599913	1.7	-4.933447
16500															0.34	-18.91284675	1.2	-7.958800173	1.8	-4.436975
17000															0.2	-23.52182518	1	-9.542425094	2.1	-3.098039
17500															0.1	-29.54242509	1	-9.542425094	2	-3.521825
18000																	0.82	-11.26614805	2	-3.521825
18500																	0.55	-14.7351713	2	-3.521825
19000																	0.45	-16.47817482	2	-3.521825
19500																	0.25	-21.58362492	2	-3.521825
20000																	0.1	-29.54242509	1.9	-3.967353
22000																	0.1	-29.54242509	1.7	-4.933447
25000																			1.7	-4.933447
27000																			1.8	-4.436975
30000																			1.7	-4.933447
32000																			1.9	-3.967353
35000																			1.67	-5.088096
37000																			1.7	-4.933447
40000																			1.75	-4.681664

Table 3.1
Audio Filter Frequency Response (figures in white columns represent signal amplitude in Volts)

Run	Ultravis Setting				SAF Reading		DATA FILE SAVED AS :-
	Voltage V	Current A	/ Wire Speed 0-255	Travel Speed mm/S	Voltage V	Current A	41v1cl.nwd - 41v10cl.nwd
1	40	520	130	6.5	40.5	500 +/- 10	
2	40	520	130	6.5	40.5	510 +/- 10	
3	40	520	130	6.5	40.5	515 +/- 10	COMMENTS :-
4	40	520	130	6.5	40.5	490 +/- 10	all welds on clean cold plate
5	40	520	130	6.5	40.5	500 +/- 10	Weld current self compensating
6	40	520	130	6.5	40.5	510 +/- 10	
7	40	520	130	6.5	40.5	505 +/- 10	
8	40	520	130	6.5	40.5	500 +/- 10	
9	40	520	130	6.5	40.5	500 +/- 10	
10	40	520	130	6.5	40.5	495 +/- 10	

Table 3.2
Initial Weld Recordings for Optimum Voltage

Run	Ultravis Setting				SAF Reading		DATA FILE SAVED AS :-
	Voltage V	Current A	/ Wire Speed 0-255	Travel Speed mm/S	Voltage V	Current A	50v1.nwd - 50v10.nwd
1	50	520	130	6.5	40.5	525 +/- 10	
2	50	520	130	6.5	40.5	478 +/- 10	
3	50	520	130	6.5	40.5	508 +/- 10	COMMENTS :-
4	50	520	130	6.5	40.5	480 +/- 10	all welds on clean cold plate
5	50	520	130	6.5	40.5	544 +/- 10	Weld current self compensating
6	50	520	130	6.5	40.5	481 +/- 10	
7	50	520	130	6.5	40.5	472 +/- 10	
8	50	520	130	6.5	40.5	465 +/- 10	
9	50	520	130	6.5	40.5	500 +/- 10	
10	50	520	130	6.5	40.5	450 +/- 10	

Table 3.3
Initial Weld Recordings for High Voltage

Run	Ultravis Setting				SAF Reading		DATA FILE SAVED AS :-
	Voltage V	Current A	/ Wire Speed 0-255	Travel Speed mm/S	Voltage V	Current A	25v1.nwd - 25v10.nwd
1	25	520	130	6.5	40.5	480+/- 10	
2	25	520	130	6.5	40.5	497 +/- 10	
3	25	520	130	6.5	40.5	528 +/- 10	COMMENTS :-
4	25	520	130	6.5	40.5	524 +/- 10	all welds on clean cold plate
5	25	520	130	6.5	40.5	507 +/- 10	Weld current self compensating
6	25	520	130	6.5	40.5	495 +/- 10	
7	25	520	130	6.5	40.5	492 +/- 10	
8	25	520	130	6.5	40.5	548 +/- 10	
9	25	520	130	6.5	40.5	519 +/- 10	
10	25	520	130	6.5	40.5	485 +/- 10	

Table 3.4
Initial Weld Recordings for Low Voltage

Training Mode						Training Mode			Training Mode		
Low Volatge	Optimum Voltage	High Voltage		Low Volatge	Optimum Voltage	High Voltage		Low Volatge	Optimum Voltage	High Voltage
20.851299	5.4313004	7.0213001		41.330969	35.360163	6.9989717		511.84386	161.93102	72.55954
13.7773	4.3843004	9.0133005		41.715967	38.752012	7.1019715		191.72789	321.161551	65.313661
17.820298	3.0903002	16.658298		35.467973	11.546409	9.9669724		456.27382	0.83649426	94.213473
17.669299	3.4533002	6.2323004		39.9456969	18.299908	12.242972		550.57638	67.330825	91.704976
18.584299	3.4903001	6.2523007		50	13.96481	6.9879716		377.65178	0.76401311	77.460646
12.6183	13.074272	3.2963001		36.0700972	13.683465	12.402703		45.584666	44.988754	73.066312
16.3753	9.3232727	3.1203001		38.23297	16.553676	11.441		22.214667	52.733439	75.663983
15.0773	5.8102726	3.9393001		44.075971	12.956792	19.143734		43.188667	0.79283601	78.76926
18.308298	1.5303273	3.9333		43.189972	33.741337	22.424378		46.211668	207.76049	118.39537
13.586299	4.909273	3.0182999		50	26.519307	12.46344		35.890668	122.794438	80.0393578
24.712271	3.3100864	4.1952728		77.664583	17.275972	15.575051		38.6180672	16.953667	6.6766679
26.978272	2.8713452	4.3242728		74.905633	30.97797	8.65977184		417.614746	30.65567	6.7796683
25.574272	2.2226489	4.2362728		65.644717	6.8439716	8.99897		450.84386	6.5221196	9.6446686
14.810273	4.0197205	3.4322731		69.202476	9.6439725	7.1245		188.72789	9.3216687	11.920669
20.380273	1.88108	4.7252729		157.56069	6.5269714	9.9669724		432.27382	6.2051194	6.6656678
23.827271	3.7472361	7.0242726		67.908352	6.6659718	13.242972		400.57638	6.3436681	8.558669
22.596271	1.9214838	5.9912729		68.18521	8.6529713	6.9879716		286.65178	8.3306688	7.4906682
27.006271	2.510997	6.387273		83.262741	6.5389716	11.402703		46.584666	6.2171196	8.9826691
32.118272	2.8404211	3.7372729		79.017138	21.86297	12.5689		22.21896	21.540668	11.443669
25.461271	1.0000497	3.9812728		128.72588	14.388971	19.3256		33.188667	14.066667	9.4596689
9.4767999	3.249615	2.3195646		87.207242	12.93787132	20.424378		46.0986	12.61570362	8.76226858
10.0108354	16.230617	2.3342284		41.9541763	24.1377879	12.36543		34.8998	98.15477614	82.712658
9.6952569	2.455618	2.3832048		74.905784	13.6835	14.5786		37.61806	8.52211	78.39537
9.908342	9.843619	2.6993602		65.644618	16.55369	8.40876		46.614746	9.321	20.0393578
10.3733162	7.2516185	1.0366179		69.2025	12.956814	10.65432		65.6755432	7.2051194	6.67

TABLE 3.5

Initial Euclidean Errors - Optimum / Low / High Voltage Settings

(figures are Euclidean vector errors equating to FFT component amplitudes in Volts)

Software System	Input File/s	Description (all files comma delimited ascii text)	Output File/s	Header	Description (all files comma delimited ascii text)
Data Acquisition	TMS320C30 fft data monitored weld data	data from DSP board data from ADC board	*.nwd	1)No. fft frames 2)window size	7th order fft frames monitored parameters
SOFM Train	*.nwd		*.nww	1)training file name 2)No. fft frames 3)input dimension 4)output dimensions	weight values
SOFM Test	*.nwd test / validation files		*.nwf *.ord *.mtx	1)training file name 2)No. fft frames 3)input dimension 4)output dimensions	frequency activation firing order activation matrix
BP Classifier	*.nlf	converted *.nwf files with labels	*.nna		NeuralWorks ProII data file
BP Test	*.nlf test / validation files	converted *.nwf files with labels	*.nnr		NeuralWorks ProII results file
SOFM Classifer	*.nwf		*.nwl	1)fft window size 2)input dimension 3)output dimensions	trained and labelled SOFM network
SOFM Class Test	*.nlf	converted *.nwf files with labels	*.res	1)fft window size 2)input dimension 3)output dimensions	SOFM labelled network classifier results file
Run Time Monitor	*.nww *.nna		*.run	1)*.nww file name 2)*.nna file name 3)fft window size 4)input dimensions 5)output dimensions of both ANN levels	run-time results file (V, I, Ts)

TABLE 3.6
Data File Format Table

Run	Ultravis Setting				SAF Reading		DATA FILE SAVED AS :-
	Voltage V	Current A	/ Wire Speed 0-255	Travel Speed mm/s	Voltage V	Current A	f1_25v.nwd - f1_50v.nwd
1	27	520	130	6.5	25.8	491 +/- 10	
2	30	520	130	6.5	29.6	520 +/- 10	
3	34	520	130	6.5	34.2	470 +/- 10	COMMENTS :-
4	37	520	130	6.5	38.9	510 +/- 10	all welds on clean cold plate
5	40	520	130	6.5	41	553 +/- 10	Weld current self compensating
6	44	520	130	6.5	44.6	536 +/- 10	
7	47	520	130	6.5	47.6	520 +/- 10	
8	50	520	130	6.5	47.1	536 +/- 10	
9	50	520	130	6.5	47.4	543 +/- 10	
10	47	520	130	6.5	47	537 +/- 10	

TABLE 3.7
Experimental Weld Readings - Full Scale Range Voltage Tests

Run	Ultravis Setting				SAF Reading		DATA FILE SAVED AS :-
	Voltage V	Current A	/ Wire Speed 0-255	Travel Speed mm/s	Voltage V	Current A	f1_25v.nwd - f1_50v.nwd
11	44	520	130	6.5	45.2	527 +/- 10	
12	40	520	130	6.5	41.5	540 +/- 10	
13	37	520	130	6.5	37.6	530 +/- 10	COMMENTS :-
14	34	520	130	6.5	32.9	553 +/- 10	all welds on clean cold plate
15	30	520	130	6.5	29.2	520 +/- 10	Weld current self compensating
1 6	27	520	130	6.5	24.8	504 +/- 10	
17	25	520	130	6.5	23.6	523 +/- 10	
18	25	520	130	6.5	23.5	500 +/- 10	
19							
20							

TABLE 3.7 (continued)
Experimental Weld Readings - Full Scale Range Voltage Tests

TITLE : V2 (Voltage Change / Batch1) **Date:-** 5/5/96
DESCRIPTION: Constant Current / Travel Speed
Experimental Sheet No: 1 of 2

Run	Weld Setting					File Details		
	Voltage V	Current / Wire Speed A Measured	mm/s 0-255	Travel Speed mm/s	Weld Length mm	Name	No FFTs	Comments
1	25	505	130	6.5	730	2513006L	873	Filters 15Hz - 10kHz
2	27.5	510	130	6.5	800	2713006L	982	Filters 15Hz - 10kHz
3	30	496	130	6.5	760	3013006L	811	Filters 15Hz - 10kHz
4	32.5	546	130	6.5	770	3213006L	996	Filters 15Hz - 10kHz
5	35	500	130	6.5	750	3513006L	936	Filters 15Hz - 10kHz
6	37.5	481	130	6.5	700	3713006L	1077	Filters 15Hz - 10kHz
7	40	500	130	6.5	600	4013006L	930	Filters 15Hz - 10kHz
8	42.5	524	130	6.5	680	4213006L	1061	Filters 15Hz - 10kHz
9	45	487	130	6.5	690	4513006L	1104	Filters 15Hz - 10kHz
10	47.5	492	130	6.5	700	4713006L	896	Filters 15Hz - 10kHz

TABLE 3.8
Final Weld Run Recordings - Variable Voltage

TITLE : I2 (Current Change / Batch1) **Date:-** 5/5/96
DESCRIPTION: Constant Voltage / Travel Speed
Experimental Sheet No: 1 of 2

Run	Weld Setting					File Details		
	Voltage V	Current A Measured	/ Wire Speed mm/s 0-255	Travel Speed mm/s	Weld Length mm	Name	No FFTs	Comments
1	40	000	25	6.5	000	4002506L	000	No Arc Ignition
2	40	175	50	6.5	130	4005006L	333	Arc Extingiushed
3	40	277	75	6.5	560	4007506L	1059	Filters 15Hz - 10kHz
4	40	350	100	6.5	660	4010006L	1117	Filters 15Hz - 10kHz
5	40	476	125	6.5	650	4012506L	946	Filters 15Hz - 10kHz
6	40	625	150	6.5	640	4015006L	1016	Filters 15Hz - 10kHz
7	40	696	175	6.5	645	4017506L	937	Filters 15Hz - 10kHz
8	40	776	200	6.5	660	4020006L	830	Filters 15Hz - 10kHz
9	40	870	225	6.5	665	4022506L	806	Filters 15Hz - 10kHz
10	40	930	250	6.5	620	4025006L	789	Filters 15Hz - 10kHz

TABLE 3.9
Final Weld Run Recordings - Variable Current

TITLE : T2 (Travel Speed Change / Batch1) **Date:-** 5/5/96
DESCRIPTION: Constant Current / Voltage
Experimental Sheet No: 1 of 2

Run	Weld Setting					File Details		
	Voltage V	Current A Measured	Wire Speed mm/s 0-255	Travel Speed mm/s	Weld Length mm	Name	No FFTs	Comments
1	40	516	130	5.0	500	4013005L	1022	Filters 15Hz - 10kHz
2	40	498	130	7.5	700	4013007L	836	Filters 15Hz - 10kHz
3	40	504	130	10	720	4013010L	563	Filters 15Hz - 10kHz
4	40	512	130	12.5	715	4013012L	376	Filters 15Hz - 10kHz
5	40	515	130	15	720	4013015L	318	Filters 15Hz - 10kHz
6	40	477	130	17.5	725	4013017L	284	Filters 15Hz - 10kHz
7	40	502	130	20	650	4013020L	232	Filters 15Hz - 10kHz
8	40	489	130	22.5	630	4013022L	184	Filters 15Hz - 10kHz
9	40	500	130	25	650	4013025L	172	Filters 15Hz - 10kHz
10	40	510	130	27.5	640	4013027L	142	Filters 15Hz - 10kHz

TABLE 3.10
Final Weld Run Recordings - Variable Travel Speed

TITLE : Cyclic Stability Test **Date:-** 6/5/96
DESCRIPTION: Variable Voltage / Current / Travel Speed
Experimental Sheet No: 1 of 2

Run	Weld Setting					File Details		
	Voltage V	Current A Measured	Wire Speed mm/s 0-255	Travel Speed mm/s	Weld Length mm	Name	No FFTs	Comments
1	40	158	50	6.5	730	2513006L	873	Filters 15Hz - 10kHz
2	42	250	75	6.5	800	2713006L	982	Filters 15Hz - 10kHz
3	45	375	100	6.5	760	3013006L	811	Filters 15Hz - 10kHz
4	47	175	50	6.5	690	4007506L	1059	Filters 15Hz - 10kHz
5	50	280	75	6.5	660	4010006L	1117	Filters 15Hz - 10kHz
6	50	390	100	6.5	650	4012506L	946	Filters 15Hz - 10kHz
7	50	168	50	6.5	650	4013020L	232	Filters 15Hz - 10kHz
8	50	300	75	6.5	630	4013022L	184	Filters 15Hz - 10kHz
9	50	360	100	6.5	650	4013025L	172	Filters 15Hz - 10kHz
10	50	170	50	6.5	700	4713006L	896	Filters 15Hz - 10kHz

TABLE 3.11
Final Weld Run Recordings - Stability Test (attempted forced instability)

TITLE : Variable Flux Coverage **Date:-** 7/5/96
DESCRIPTION: Constant Voltage / Current / Travel Speed
Experimental Sheet No: 1 of 1

Run	Weld Setting					File Details		
	Voltage V	Current A Measured	/ Wire Speed mm/s 0-255	Travel Speed mm/s	Weld Length mm	Name	No FFTs	Comments Filters 15Hz - 10kHz
1	25	505	130	6.5	730	2513006f	850	Flux Cut - off 321
2	40	550	130	6.5	800	2713006f	967	Flux Cut - off 356
3	50	496	130	6.5	760	3013006f	769	Flux Cut - off 312
4	40	175	50	6.5	560	4007506f	980	Flux Cut - off 359
5	40	625	150	6.5	660	4010006f	1100	Flux Cut - off 405
6	40	930	250	6.5	650	4012506f	920	Flux Cut - off 364
7	40	552	130	5	650	4013020f	240	Flux Cut - off 123
8	40	515	130	15	630	4013022f	178	Flux Cut - off 97
9	40	520	130	27	650	4013025f	167	Flux Cut - off 102
10								

TABLE 3.12
Final Weld Run Recordings - Variable Flux Coverage

Test Data	Result (Input to 2nd Level)	2nd Level Network (BP) Details	Outputs	Test Data	Statistical Results
					Average Correlation
240 x 50 FFT frames of changing parameters	240 Activation Maps	187 Inputs x 24 Hidden X 4 Outputs	V	10 On-Line Welds	0.9194
240 x 50 FFT frames of changing parameters	240 Activation Maps	187 Inputs x 24 Hidden X 4 Outputs	I		0.9375
240 x 50 FFT frames of changing parameters	240 Activation Maps	187 Inputs x 24 Hidden X 4 Outputs	Ts		0.9226
240 x 50 FFT frames of changing parameters	240 Activation Maps	187 Inputs x 24 Hidden X 4 Outputs	E		0.9085
					Lowest Level Registered
64 x 50 FFT frames Changing Flux levels	64 Activation Maps	187 Inputs x 24 Hidden X 1 Output	Vflux	6 Off-Line Welds (flux reduction)	0.59, 0.51, 0.28
64 x 50 FFT frames Changing Flux levels	64 Activation Maps	187 Inputs x 24 Hidden X 1 Output	Iflux	6 Off-Line Welds (flux reduction)	0.33, 0.42, 0.38
64 X 50 FFT frames Changing Flux levels	64 Activation Maps	187 Inputs x 24 Hidden X 1 Output	Tsflux	6 Off-Line Welds (flux reduction)	0.6, 0.72, 0.93
					Average Correlation
120 x 50 FFT frames forced unstable welds	120 Activation Maps	187 Inputs x 24 Hidden X 2 Outputs	Stability Index	5 Off-Line Welds (forced unstable)	0.5622
			Power Ratio	5 Off-Line Welds (forced unstable)	0.5626

Table 4.1
Set-Up and Results Summary

	Network Parameter Monitored	Input	1st Level Network (SOFM) Details	Training Data	Result
	Voltage	7th Order FFT	64 inputs x 187 outputs (11x17)	1500 FFT frames (Ideal Weld)	Trained SOFM
	Current	7th Order FFT	64 inputs x 187 outputs (11x17)	1500 FFT frames (Ideal Weld)	Trained SOFM
Electrical Data					
	Travel Speed	7th Order FFT	64 inputs x 187 outputs (11x17)	1500 FFT frames (Ideal Weld)	Trained SOFM
	Input Energy	7th Order FFT	64 inputs x 187 outputs (11x17)	1500 FFT frames (Ideal Weld)	Trained SOFM
	Flux Coverage	7th Order FFT	64 inputs x 187 outputs (11x17)		
				High, Medium, Low V (Good Flux)	Trained SOFM
				High, Medium, Low I (Good Flux)	Trained SOFM
Quality Monitor				High, Medium, Low Ts (Good Flux)	Trained SOFM
	Weld Stability	7th Order FFT	64 inputs x 187 outputs (11x17)	1500 FFT frames (Ideal Weld)	Trained SOFM

Table 4.1
Set-Up and Results Summary

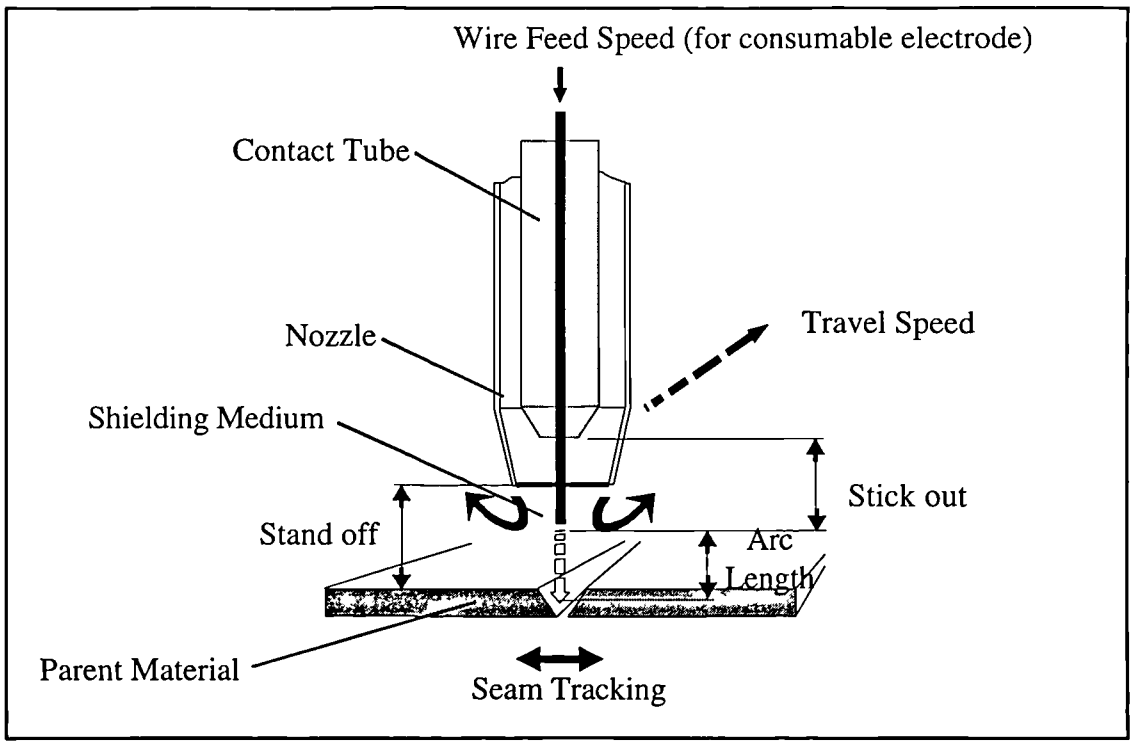


FIGURE 2.1
Typical Fusion Welding Set-Up.

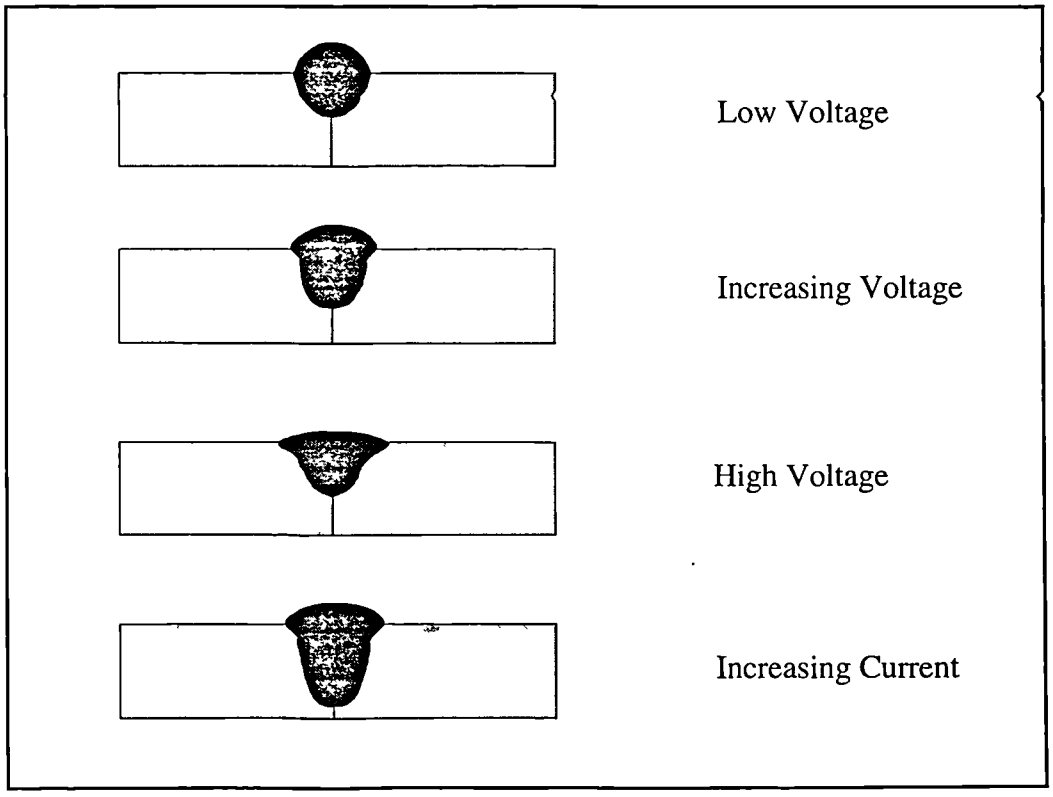


FIGURE 2.2
Varying Weld Bead Profiles

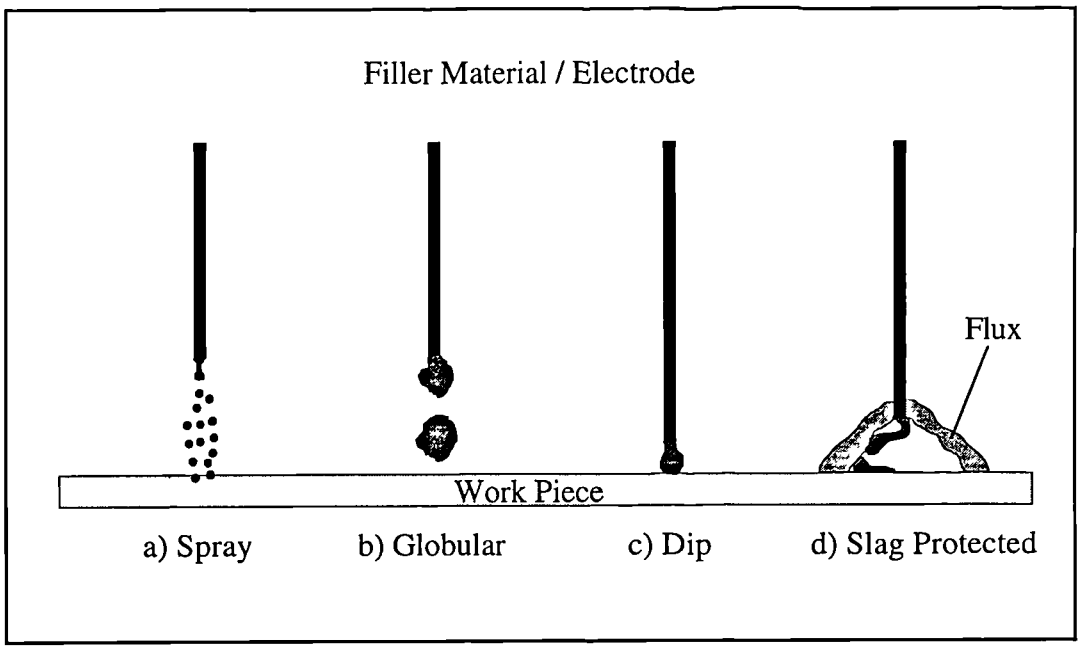


FIGURE 2.3a-d
Transfer Mechanisms Encountered in Fusion Welding Processes

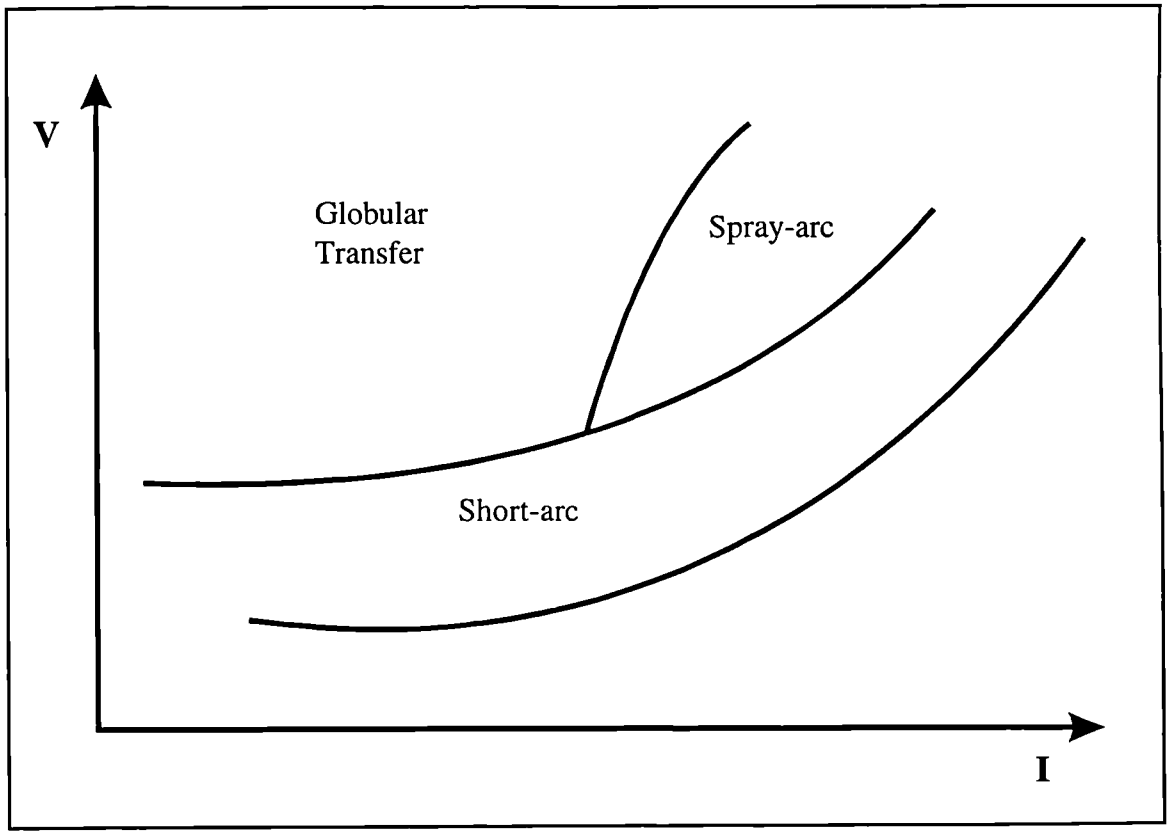


FIGURE 2.4
V/I Characteristics for Transfer Modes (in principle) . As given by Chawla(1993).

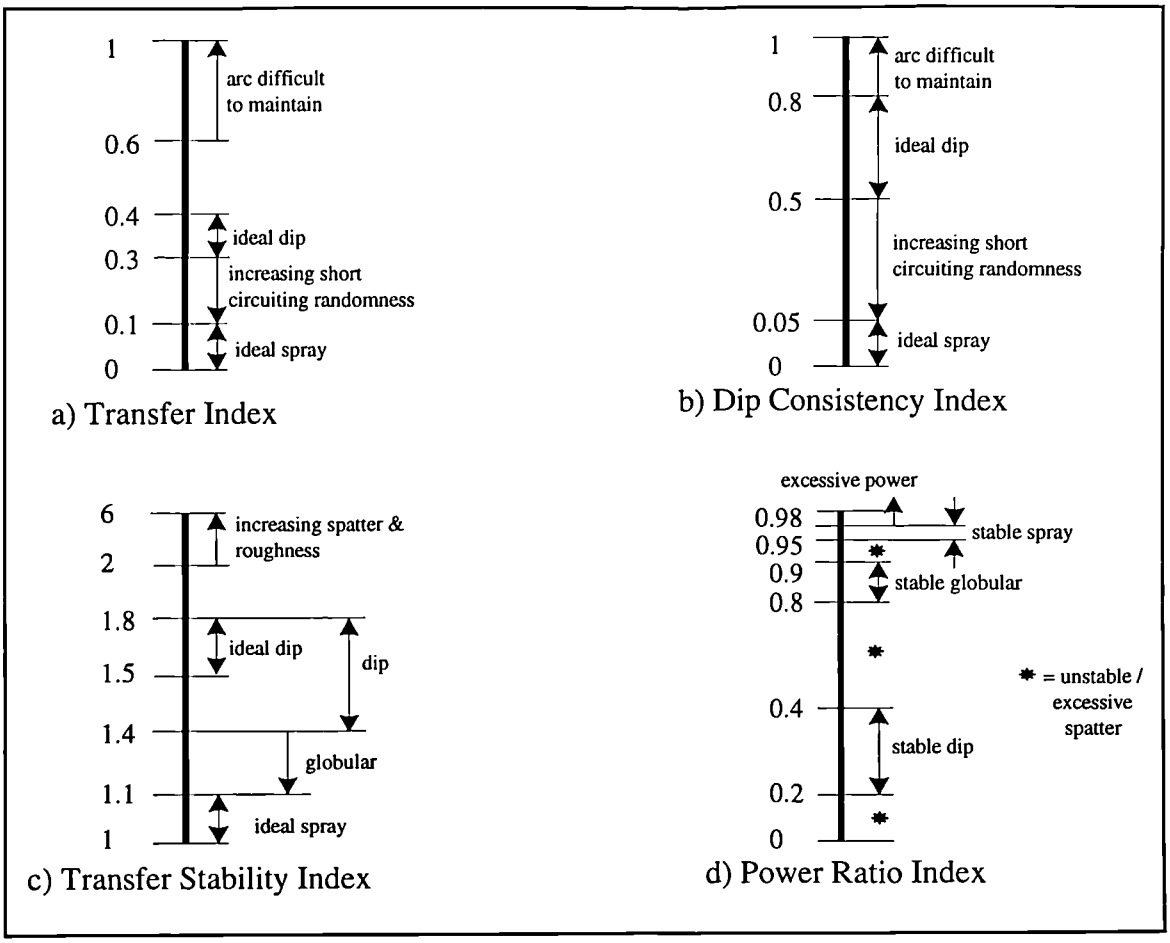


FIGURE 2.5

Stability / Transfer Mode Indices (Ogunbiyi & Norrish, 1996)

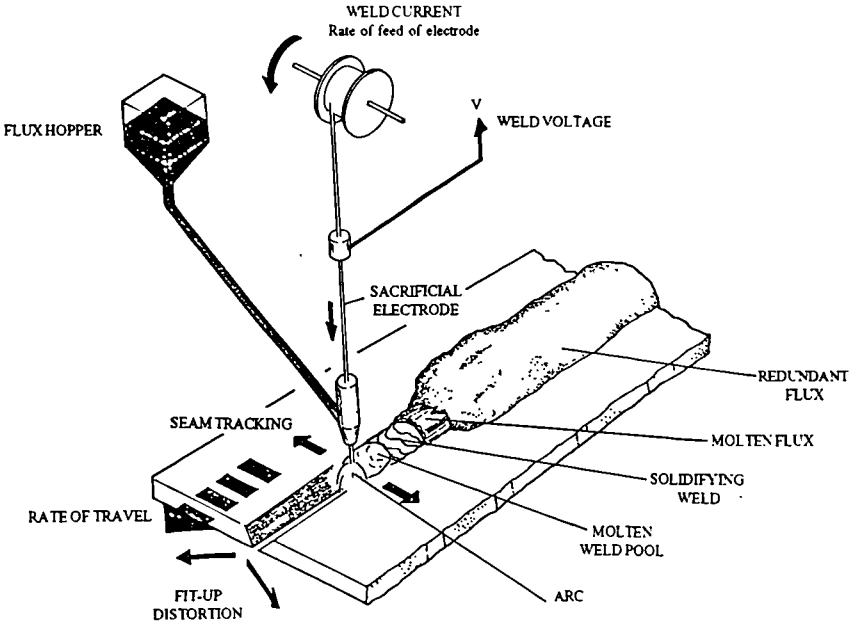


FIGURE 2.6

Submerged Arc Welding Set-Up

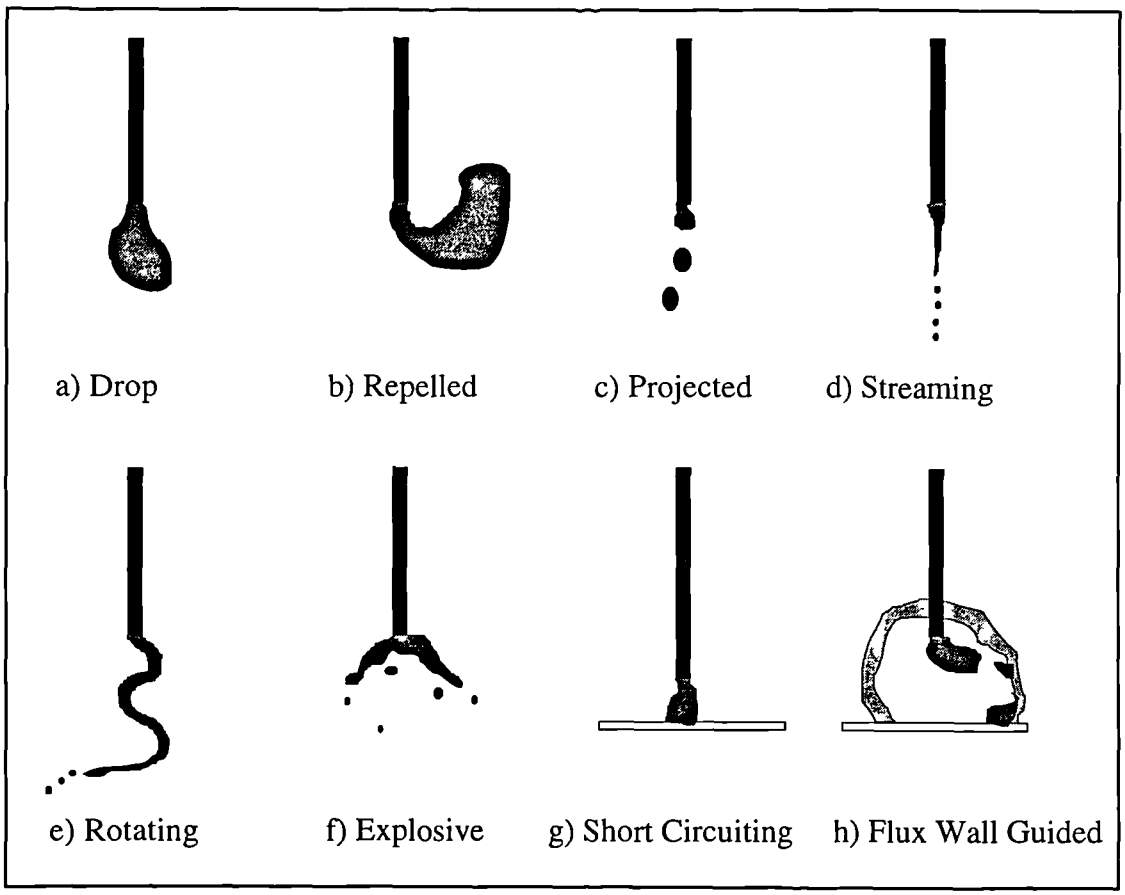


FIGURE 2.7a-h
Transfer Modes in SAW (Van Adrichem, 1966)

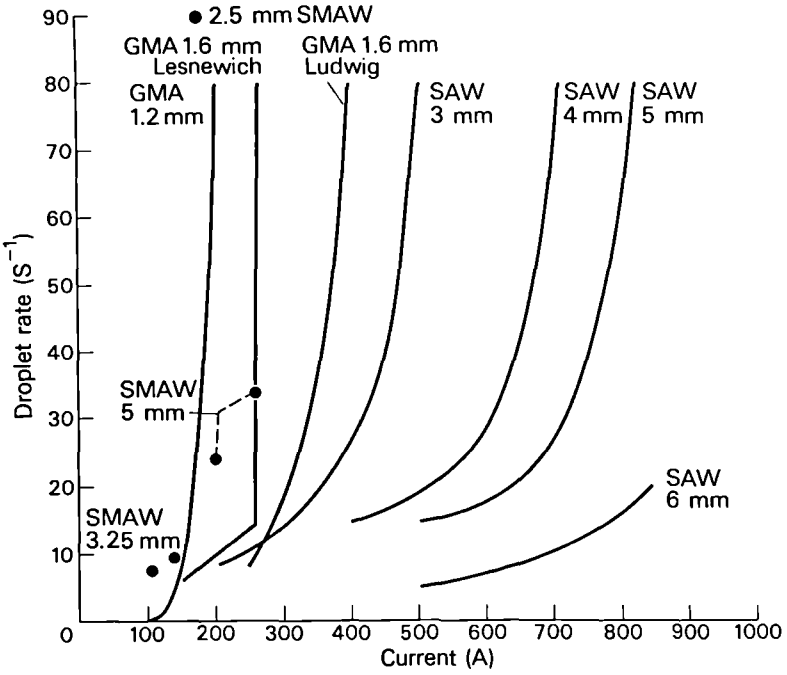


FIGURE 2.8
Material Deposition Rates (Lancaster, 1984)

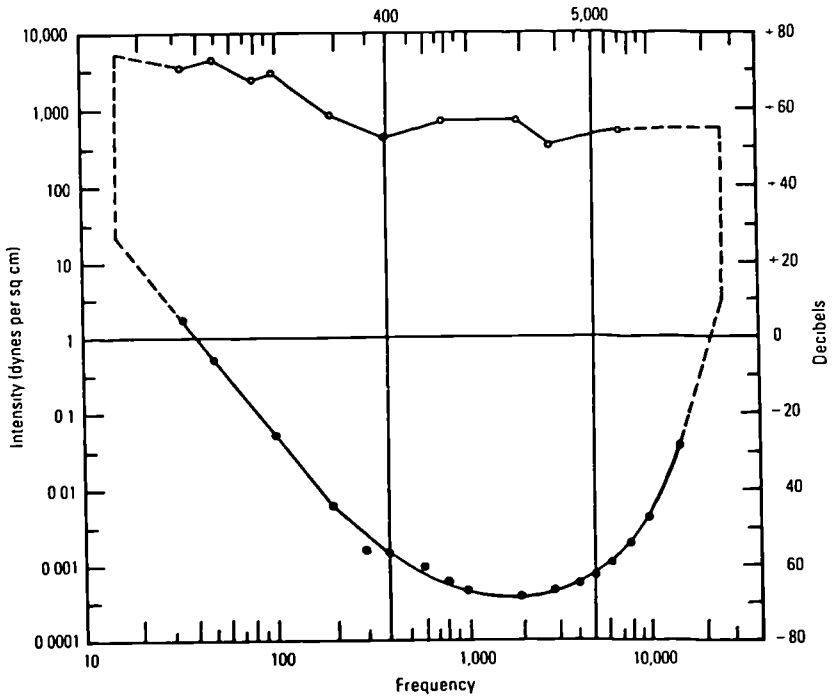


FIGURE 2.9

Human Audiogram (Massaro, 1989)

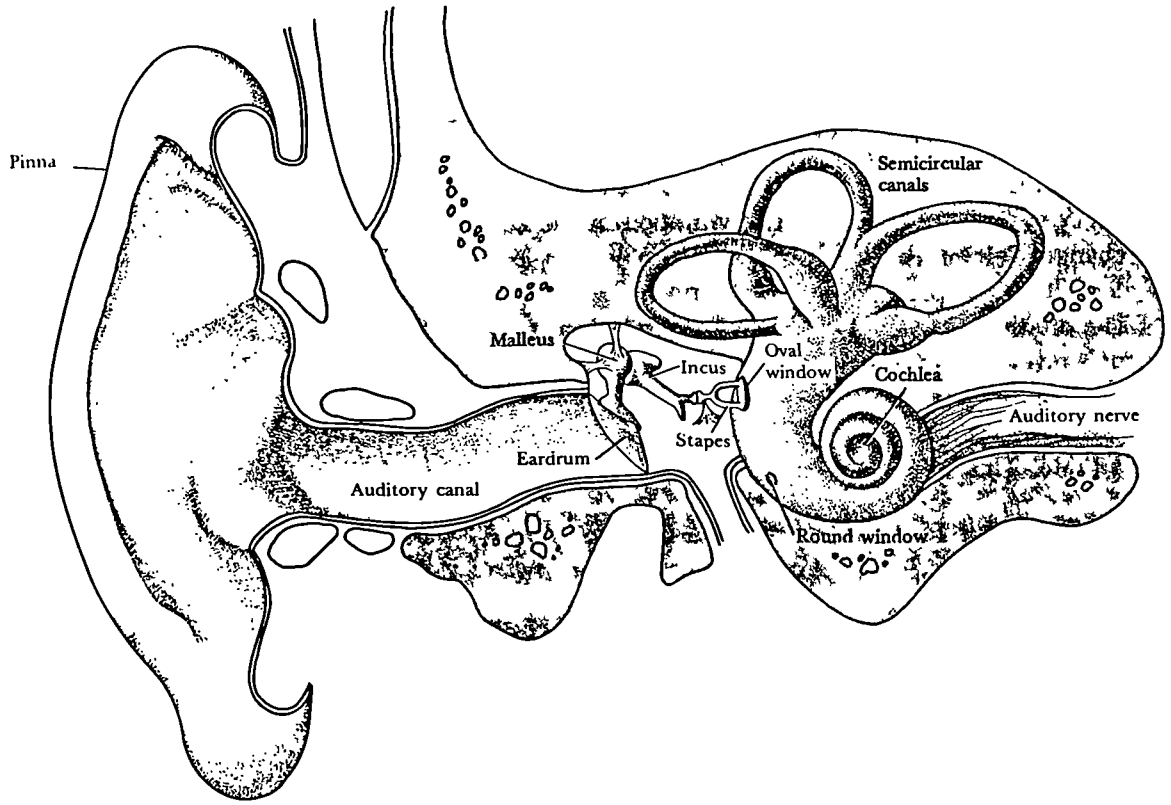


FIGURE 2.10

The Human Ear (Lindsay & Norman, 1977)

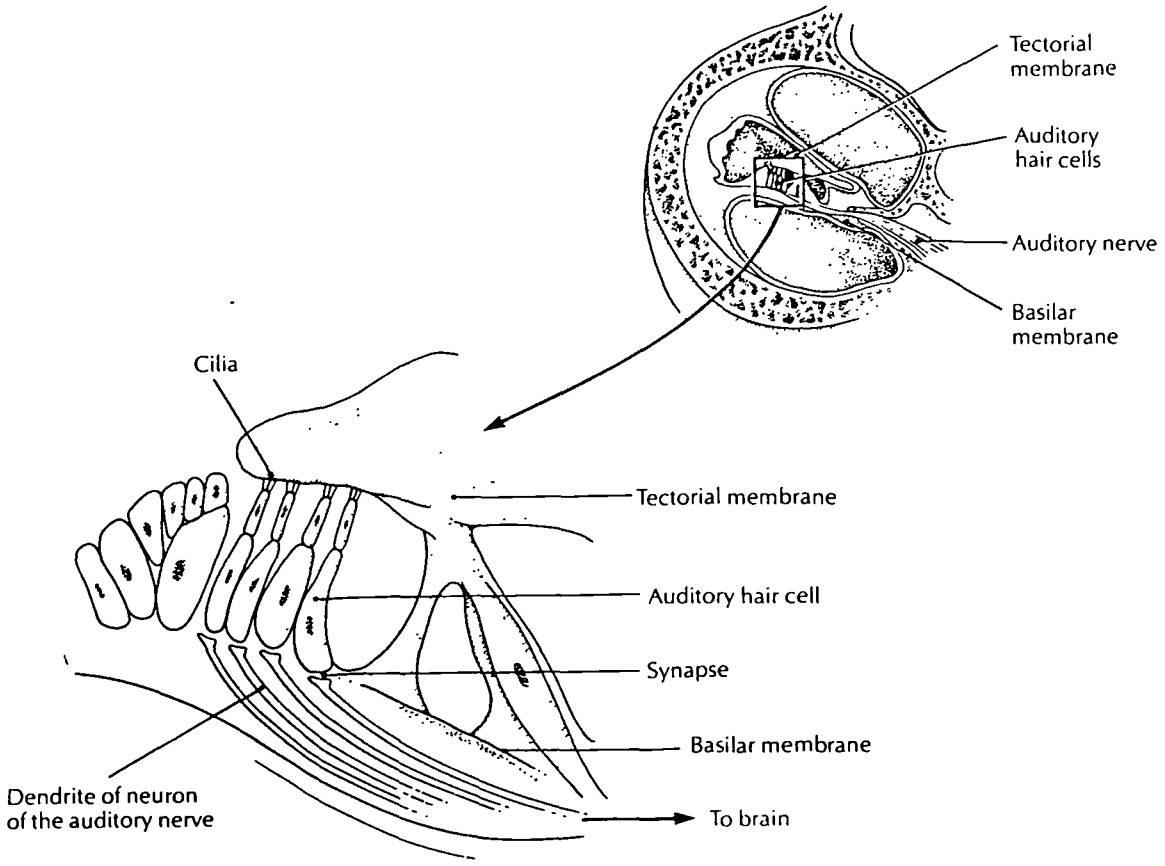


FIGURE 2.11
The Human Cochlea (Carlson, 1984)

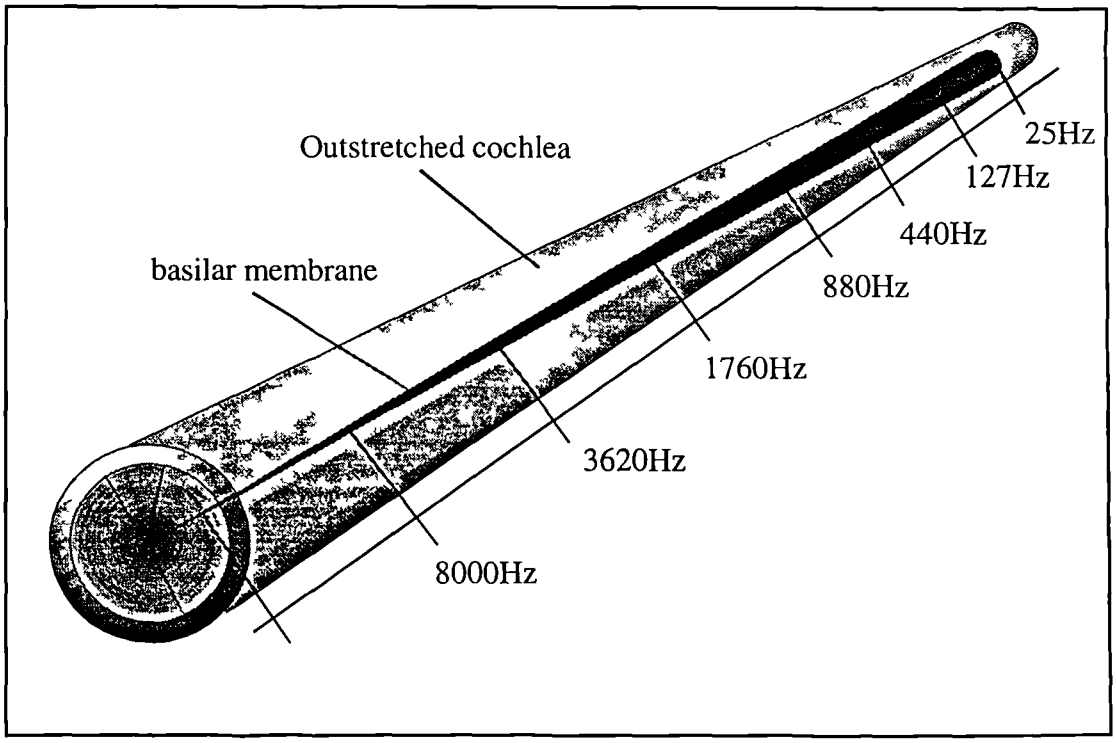


FIGURE 2.12
Frequency Dependency of the Human Cochlea

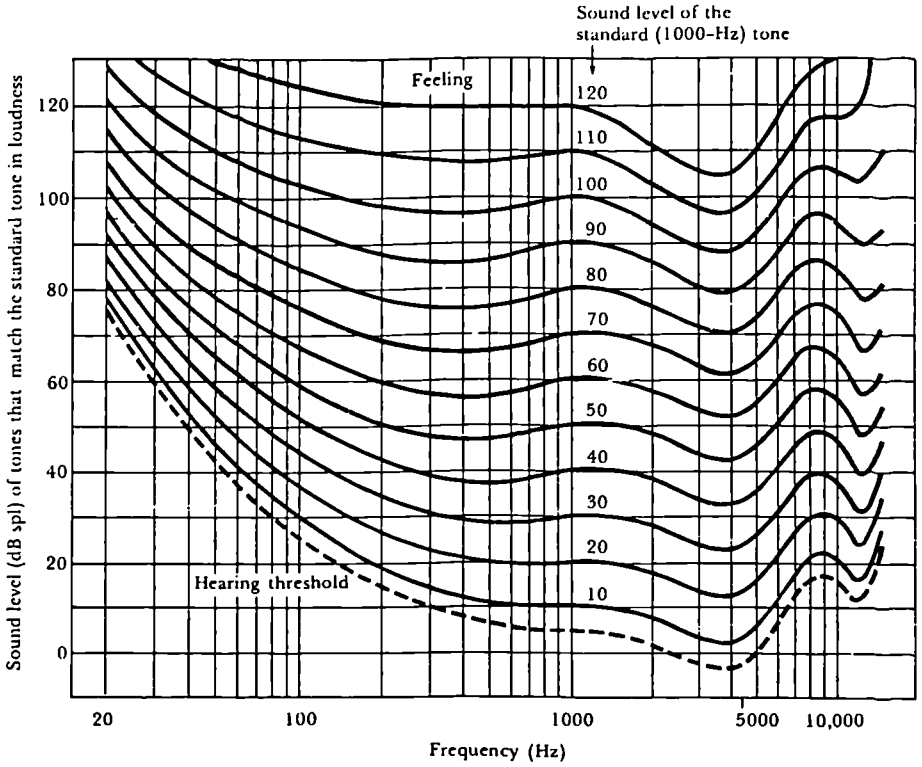


FIGURE 2.13

Relationship of Frequency, phons and SPL (Robinson & Dadson, 1956)

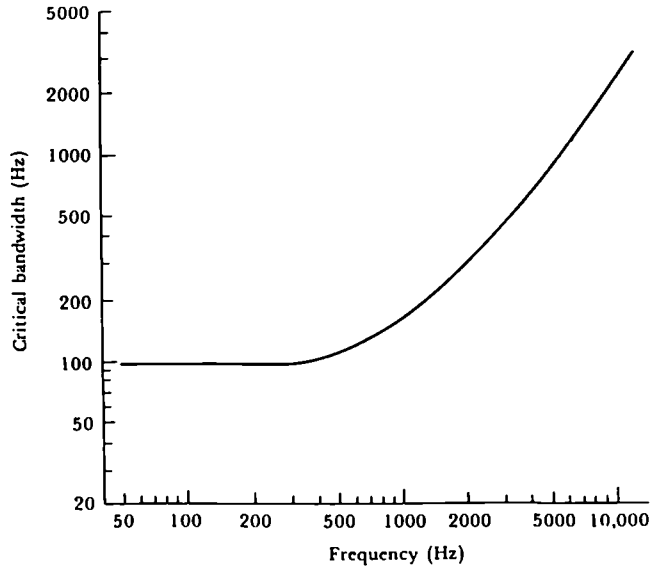


FIGURE 2.14

Relationship of Critical Bands and Frequency (Scharf, 1970)

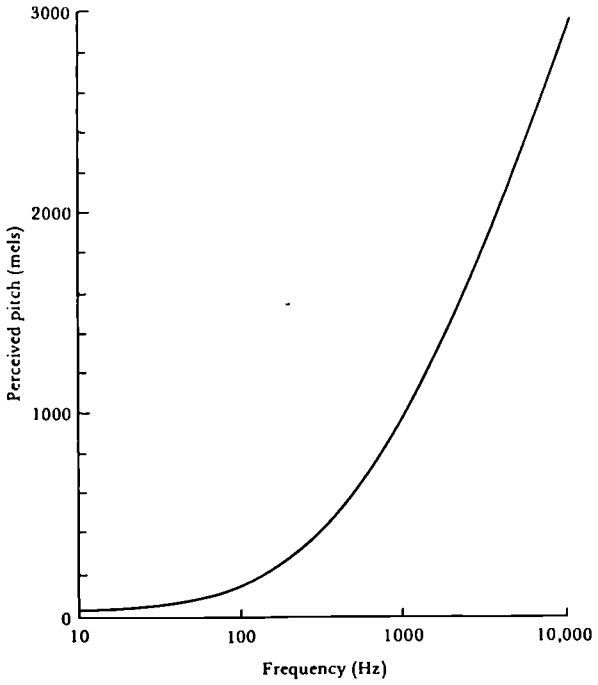


FIGURE 2.15
Relationship of Pitch and Frequency

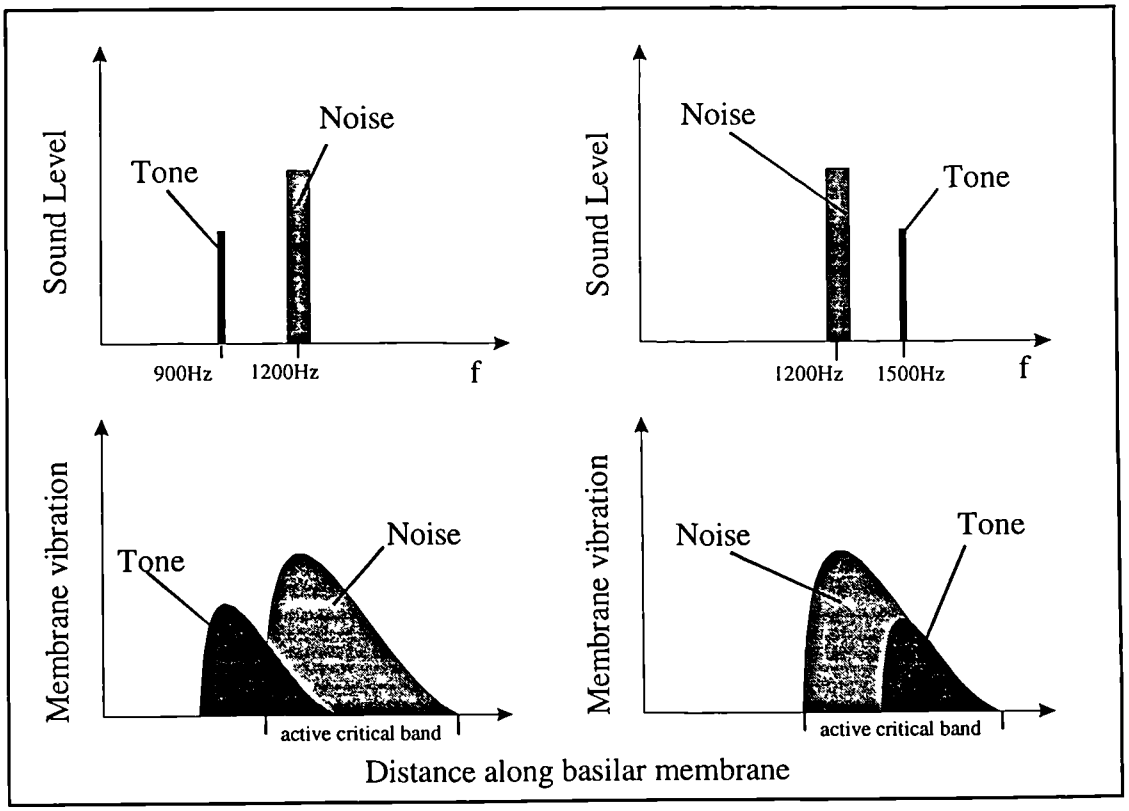


FIGURE 2.16
Masking Frequencies and the Basilar Membrane

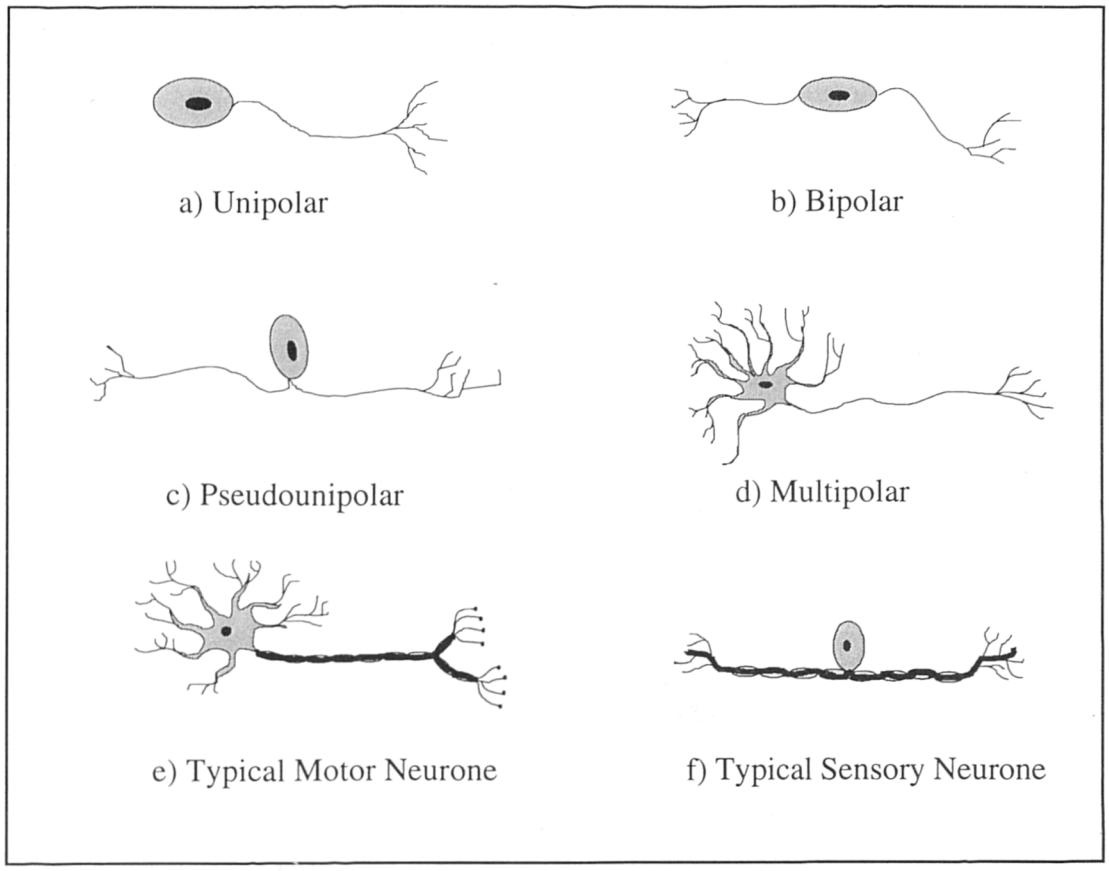


FIGURE 2.17a-f
Types of Biological Neuron

a) The detail

b) The whole picture

FIGURE 2.18a-b
Illustration of Visual Gestalt (Massaro, 1989)

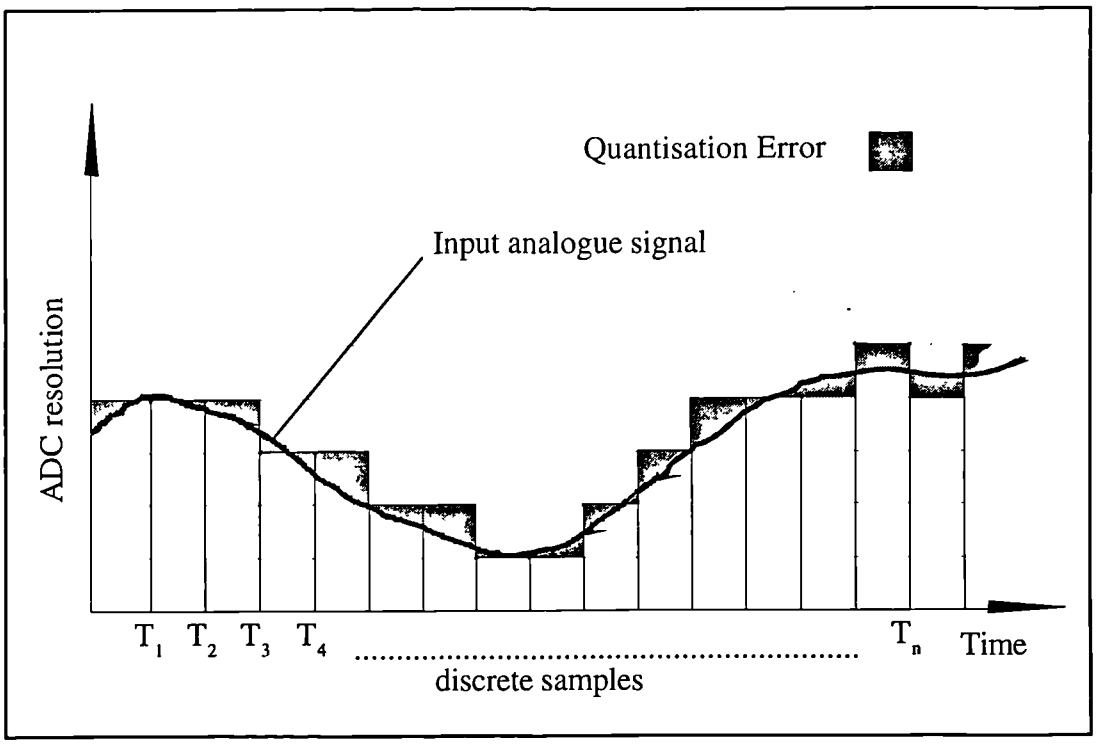


FIGURE 2.19
Sampling Quantisation Error

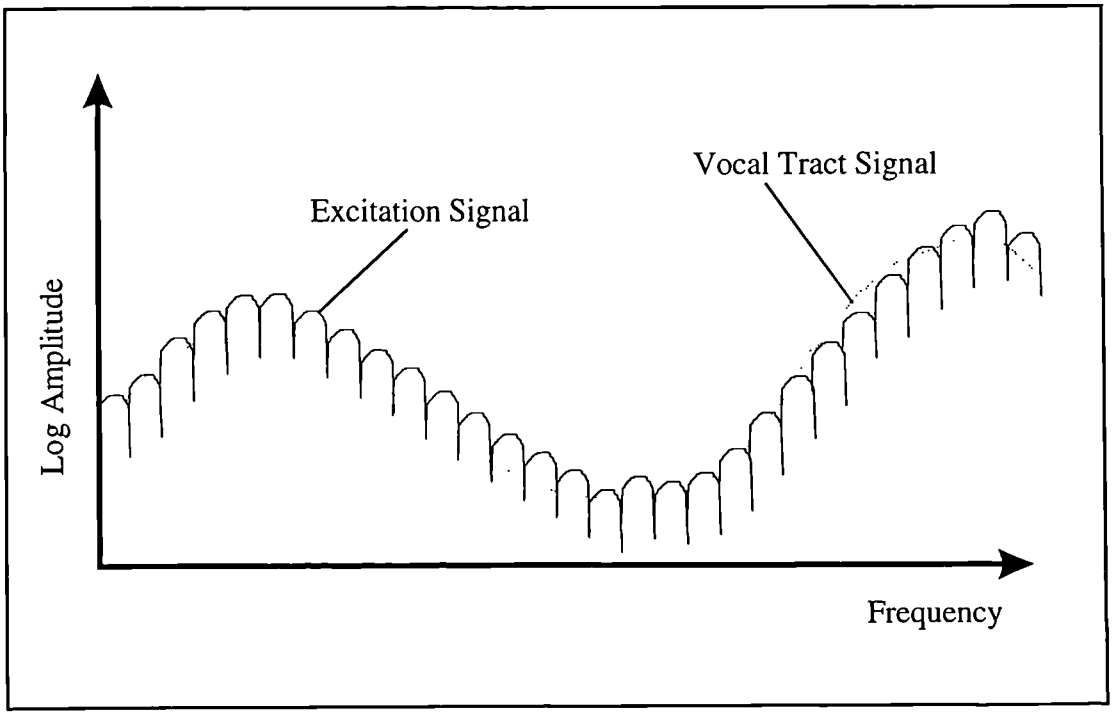


FIGURE 2.20
FFT of Vocalised Speech

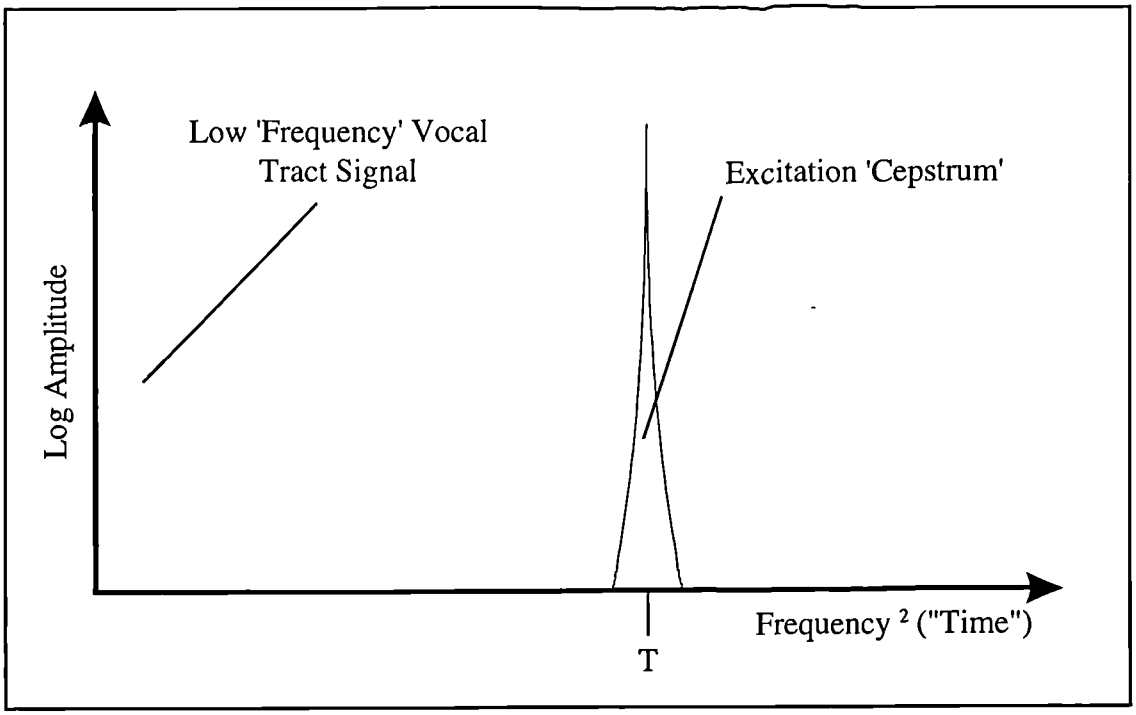


FIGURE 2.21
Cepstral Transform of Vocalised Speech

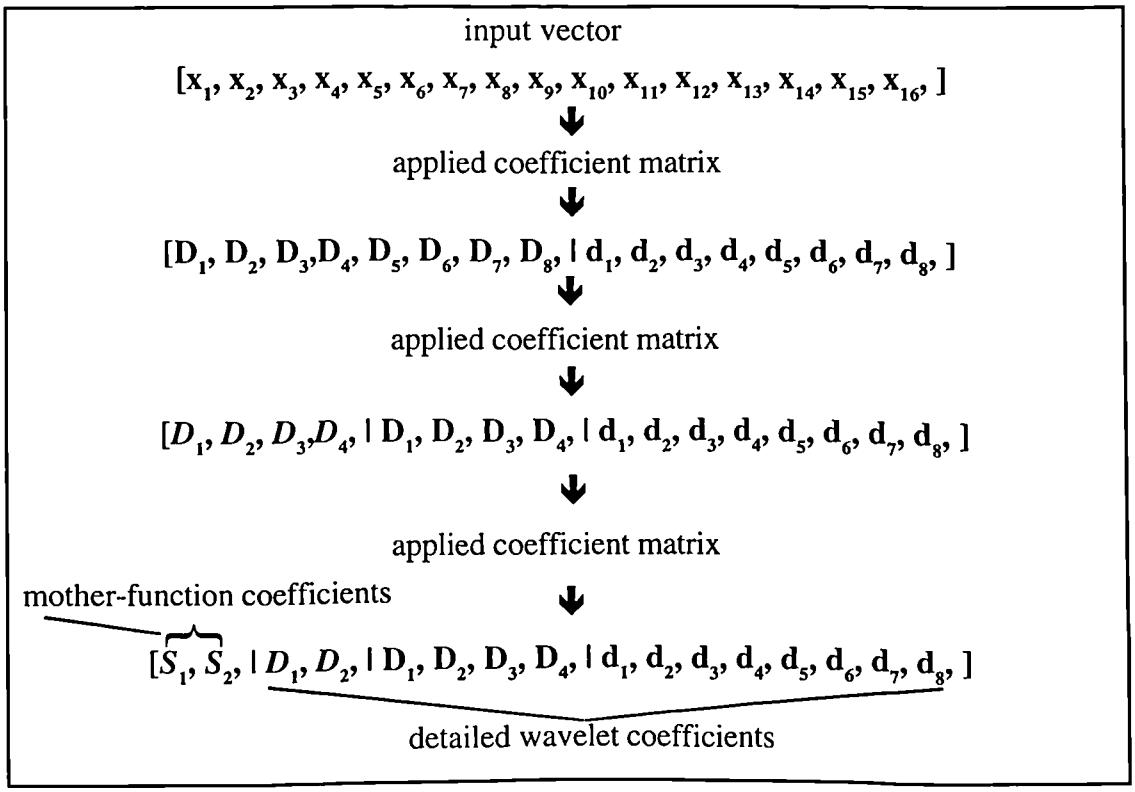
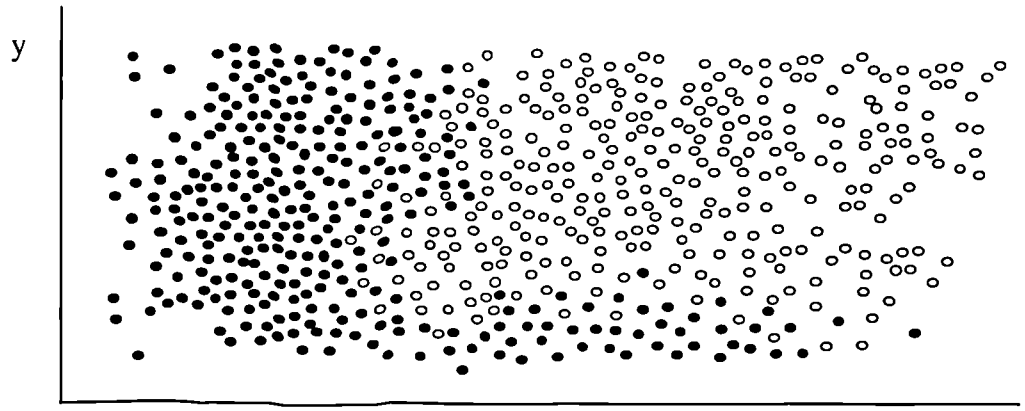
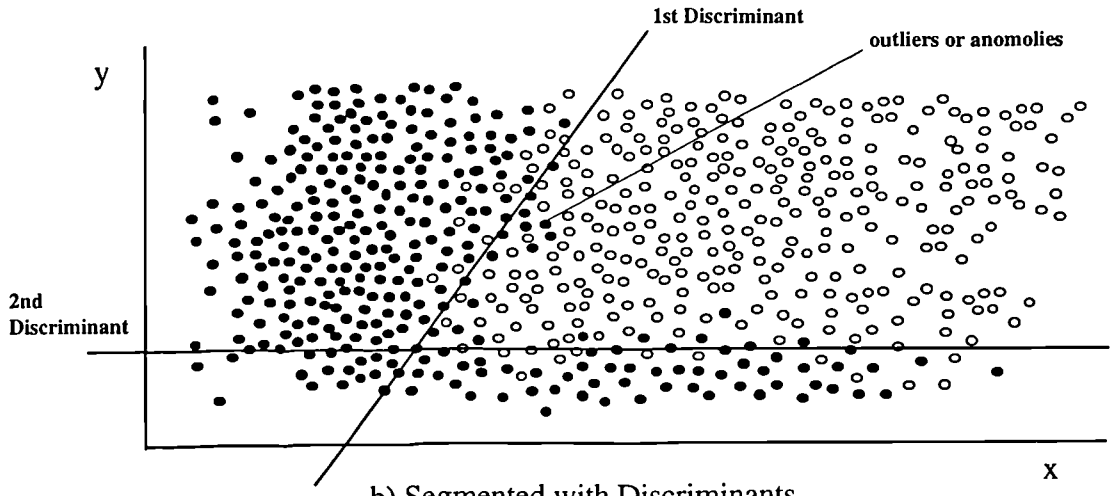


FIGURE 2.22
Application of Wavelet Transform Matrix



a) The data segmentation problem



b) Segmented with Discriminants

FIGURE 2.23a-b

2D Classification Problem with Linear Discriminants

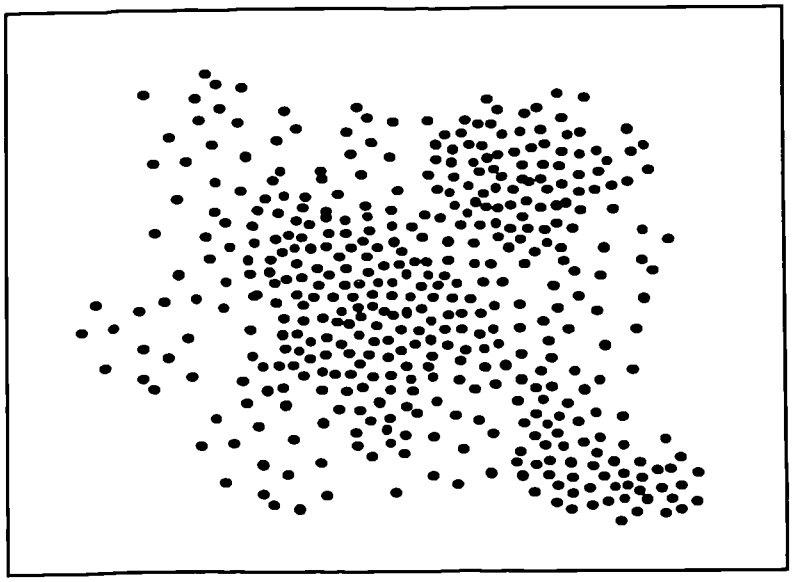


FIGURE 2.24

The Observation of Clustered Data

One, Two or Three Clusters ?

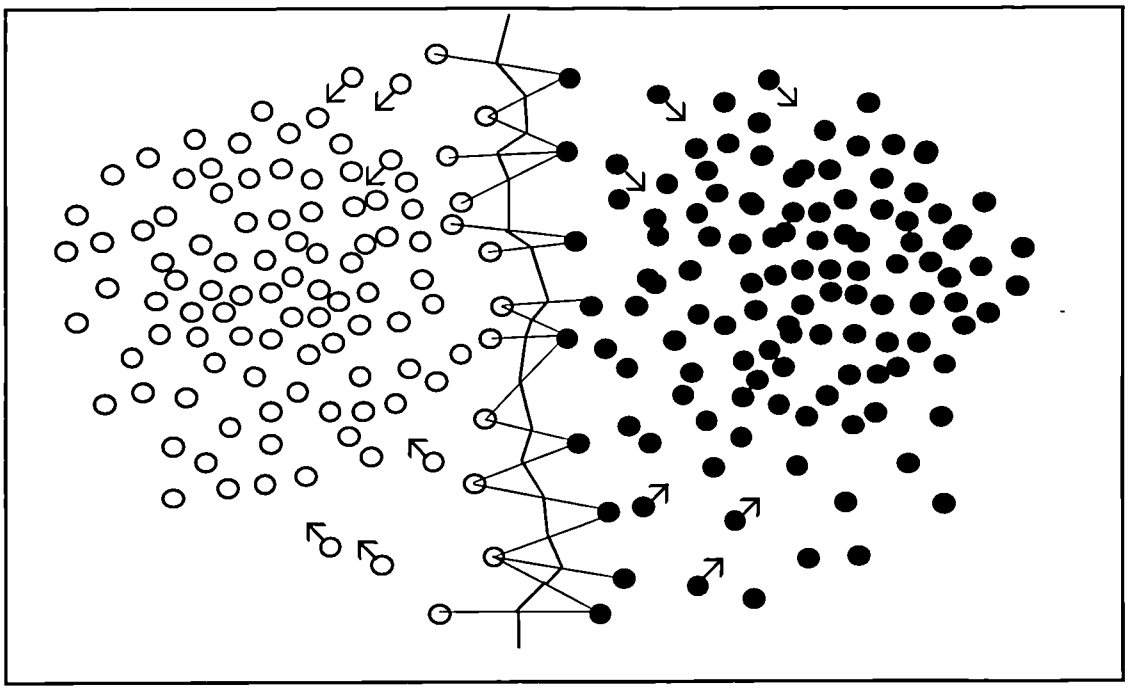


FIGURE 2.25
k-Nearest Neighbour Segmentation

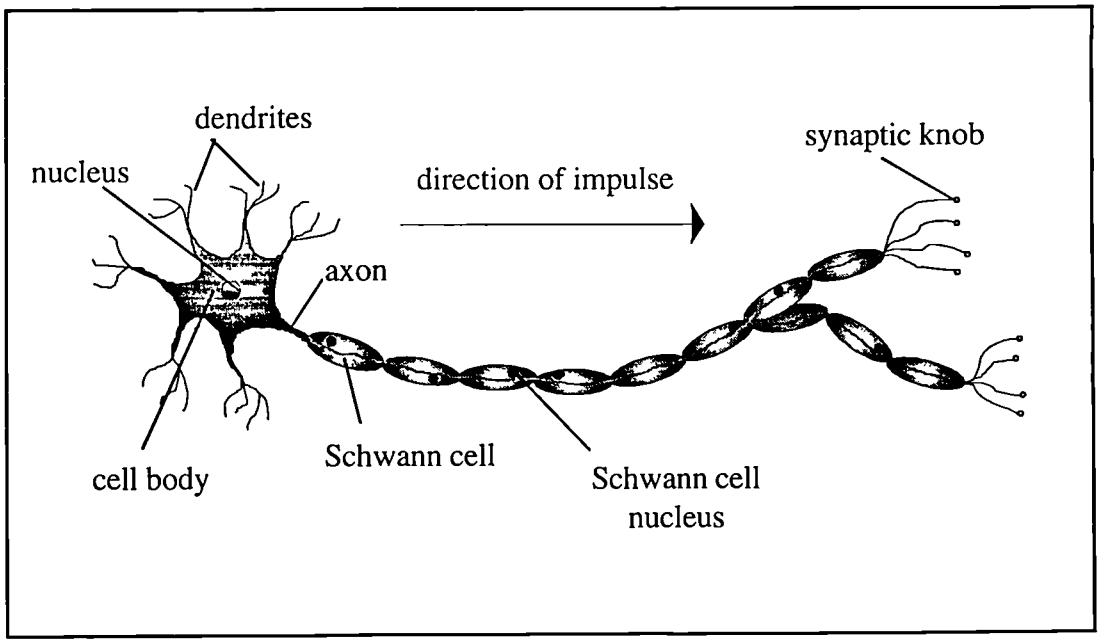


FIGURE 2.26
Individual Biological Neuron

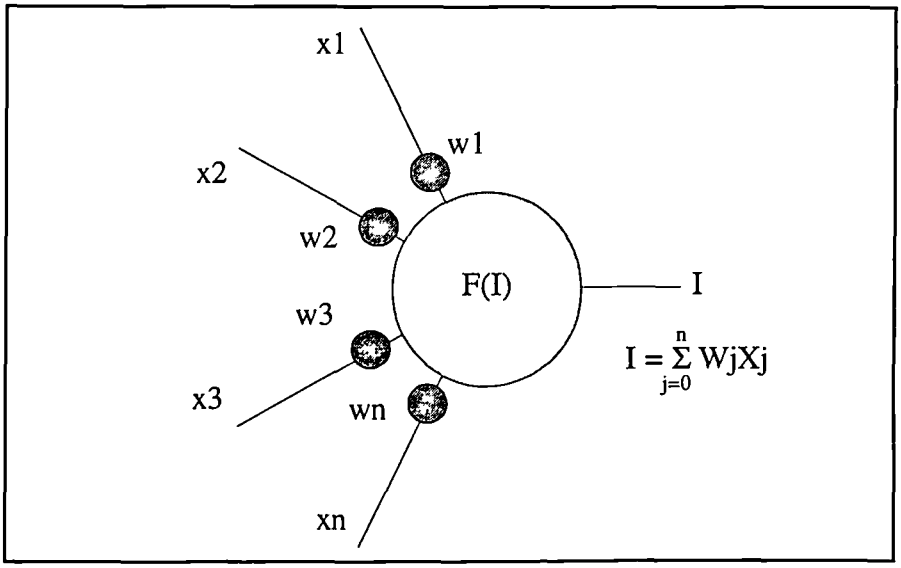


FIGURE 2.27
The Artificial Mathematical Neuron

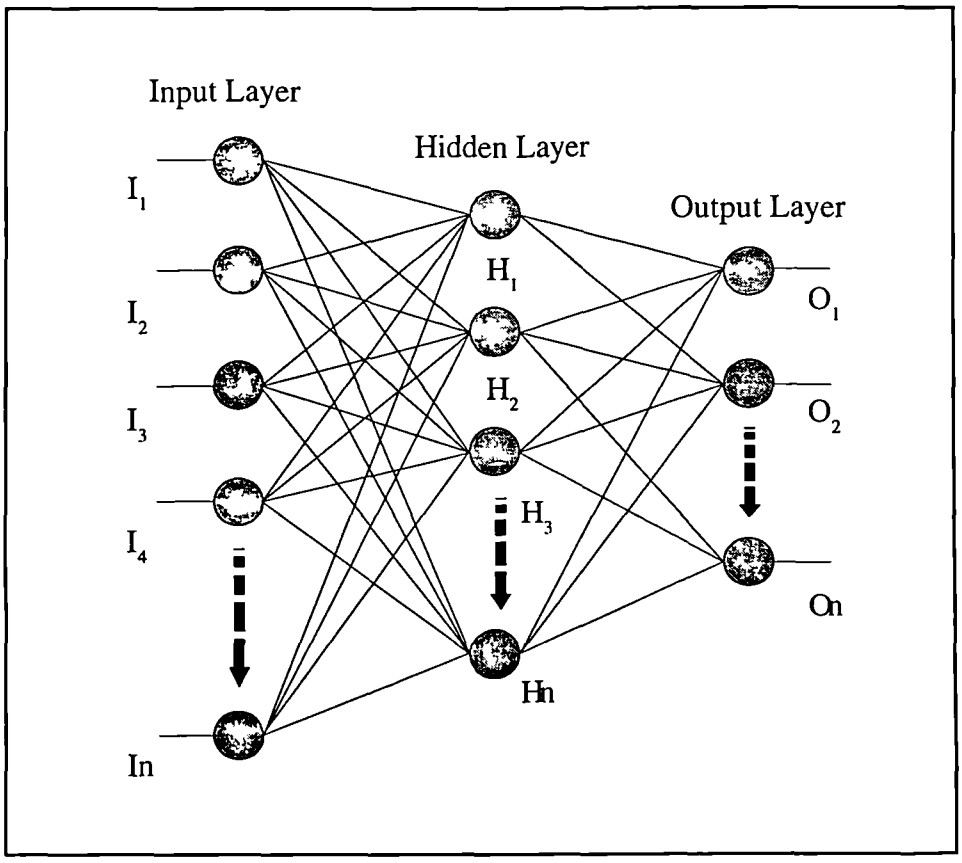


FIGURE 2.28
The Multi-Layer Perceptron

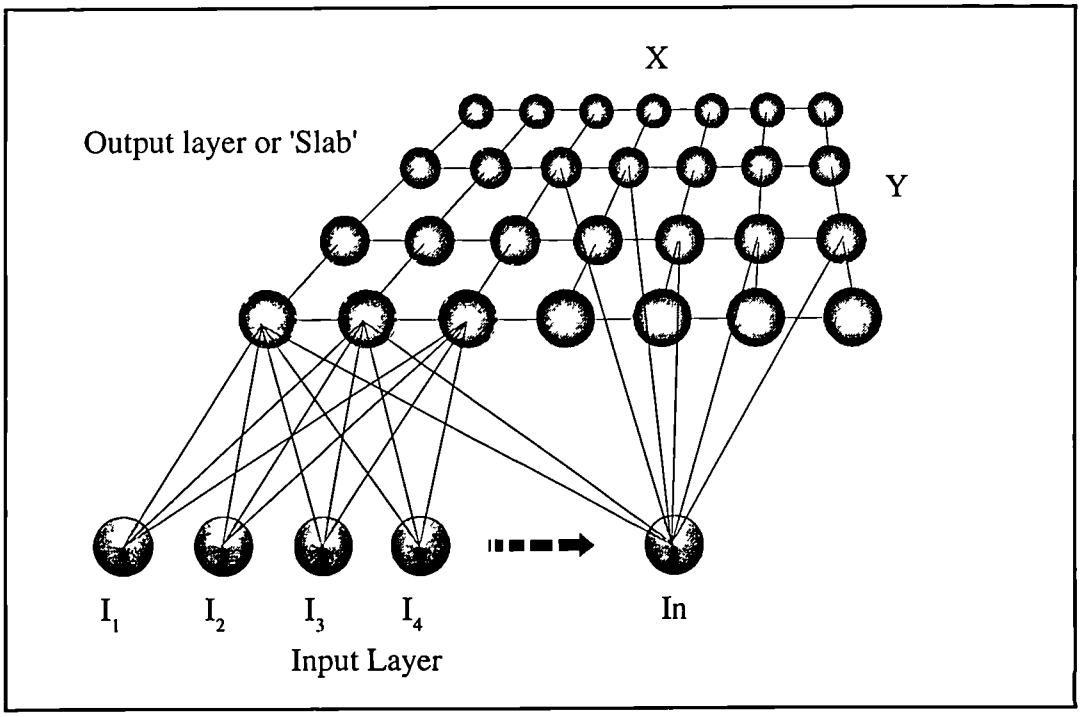


FIGURE 2.29
Self Organising Feature Map Architecture

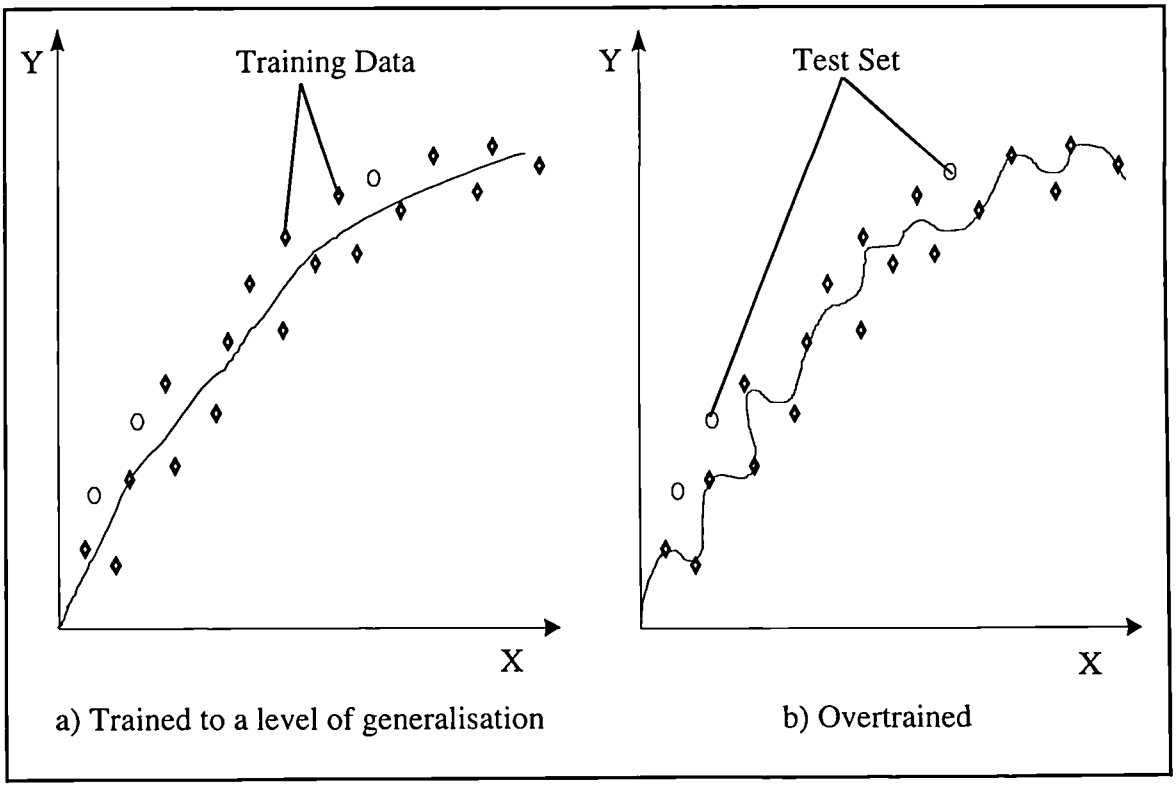


FIGURE 2.30a-b
The Affects of Overtraining a Network

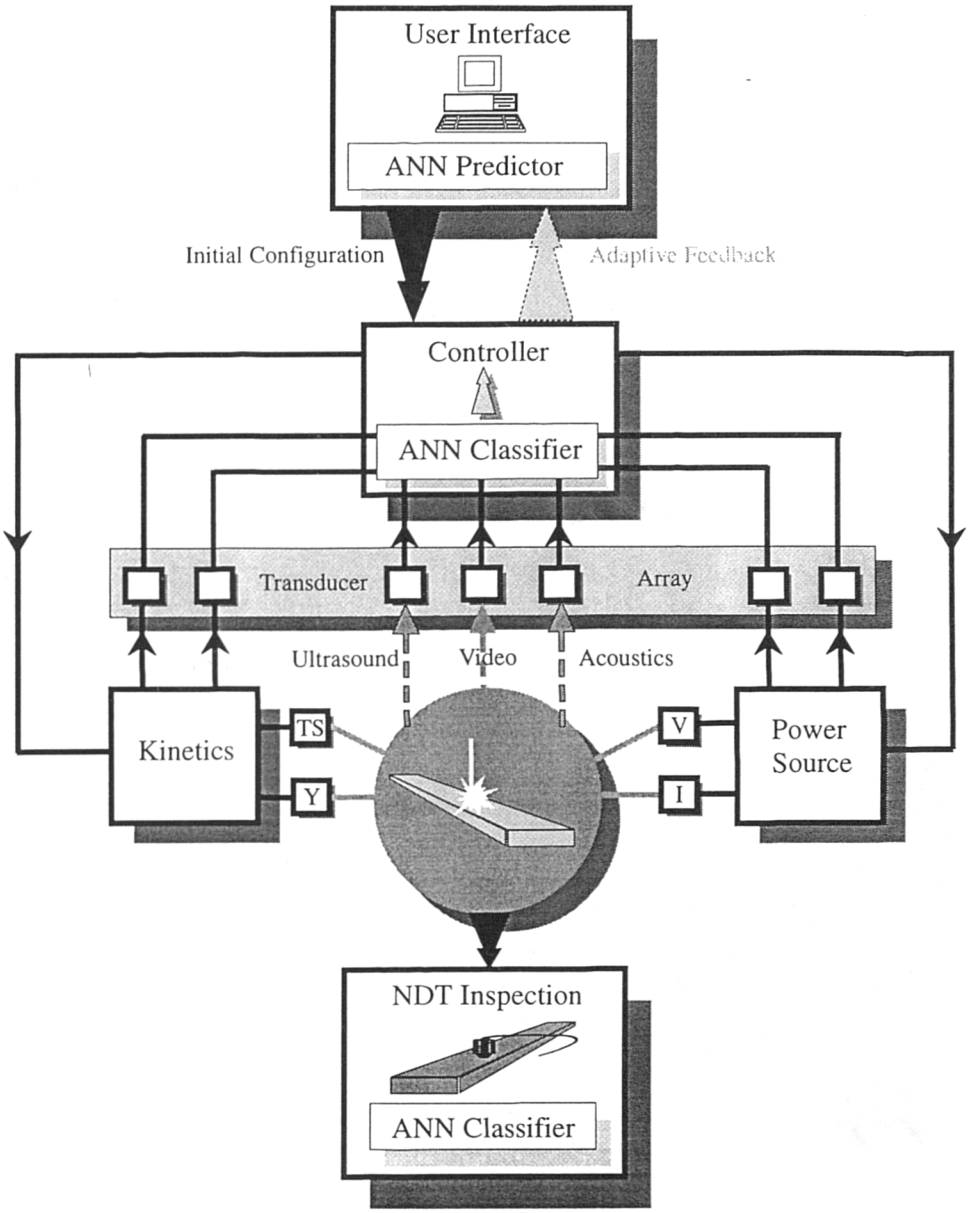


FIGURE 2.31
ANNs and the Production Weld Cycle

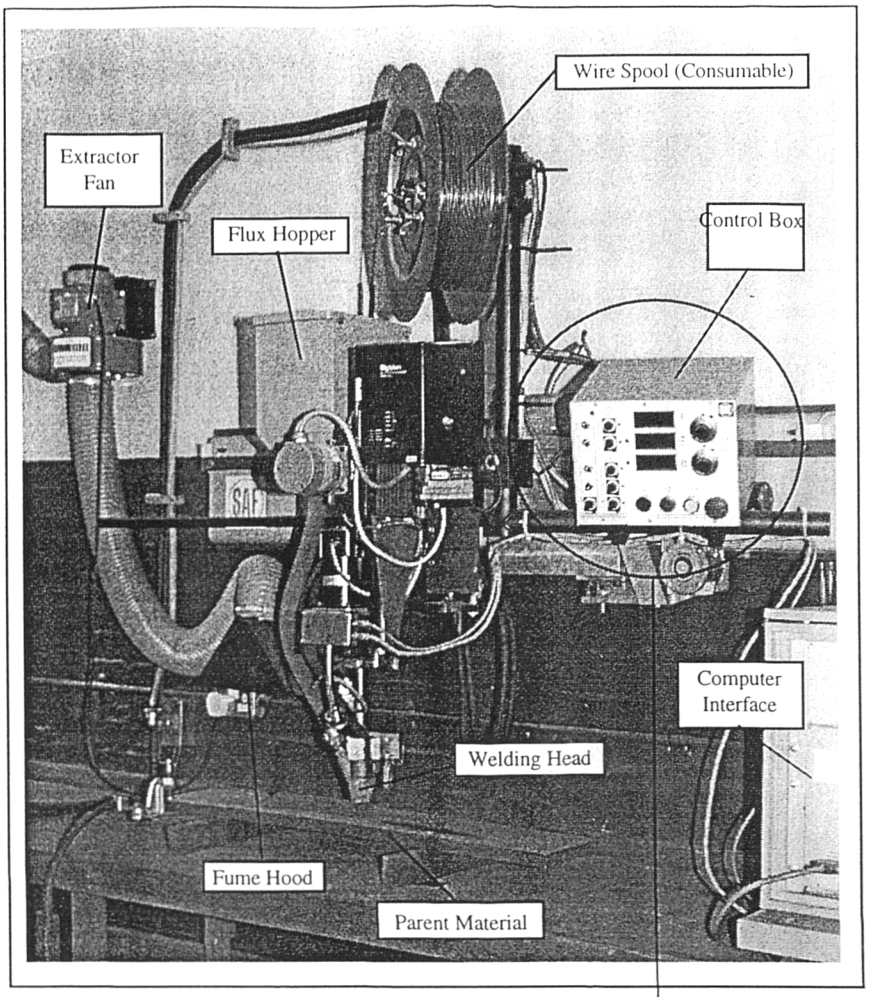


FIGURE 3.1
The Submerged Arc Welding Rig

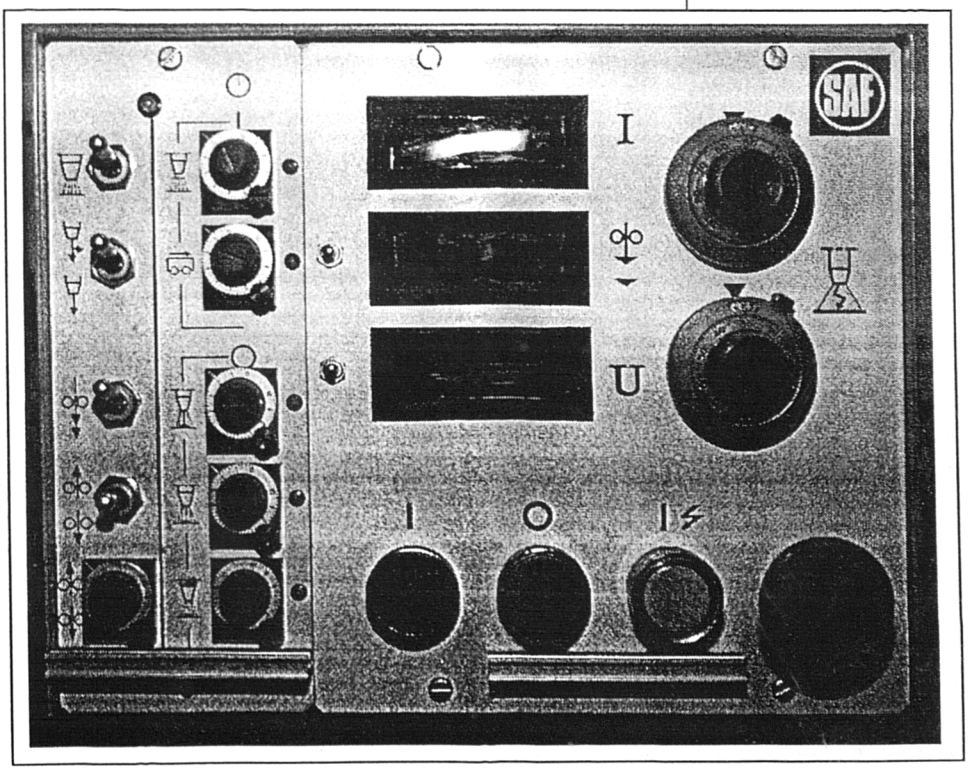


FIGURE 3.2
Weld Controller

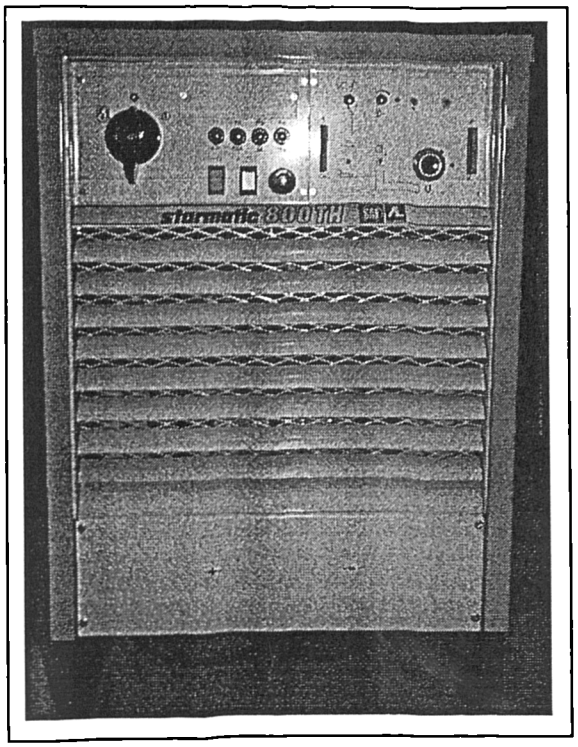


FIGURE 3.3
SAF Welding Power Source

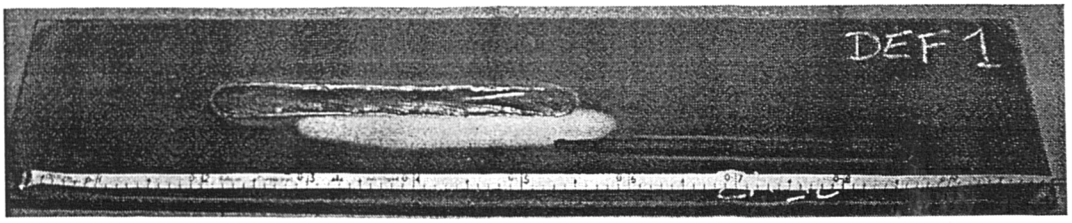


FIGURE 3.4
Typical Short Duration Welds

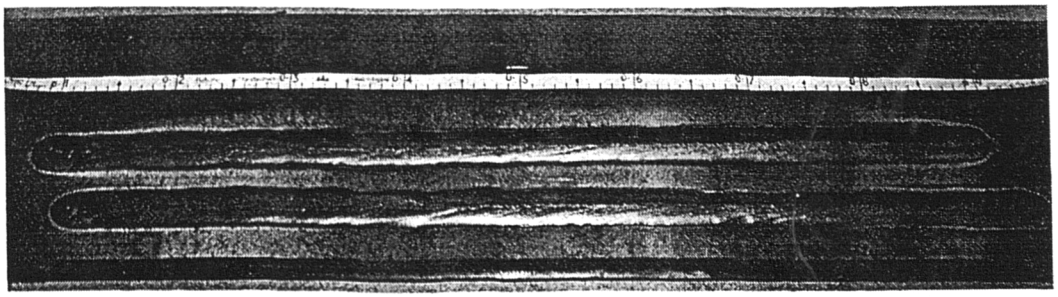


FIGURE 3.5
Typical Full Length Experimental Welds

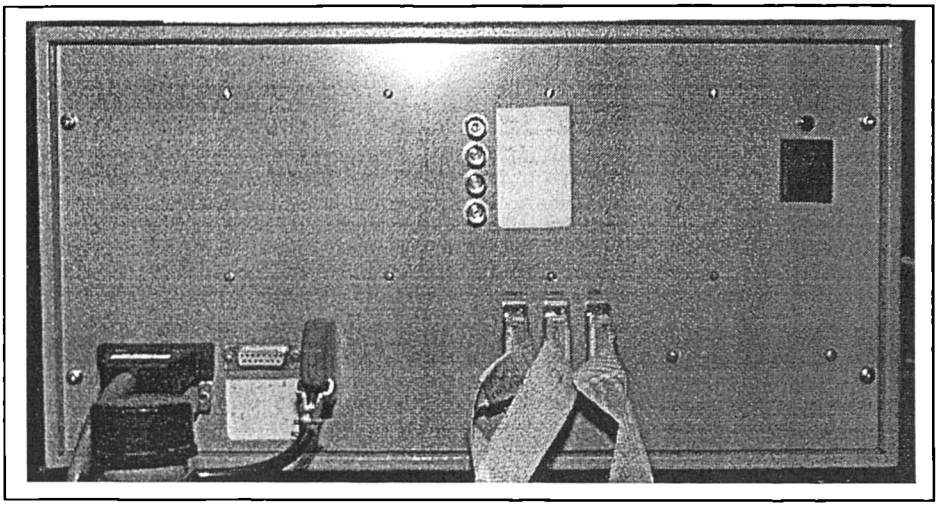


FIGURE 3.6
Ultravis Control Box

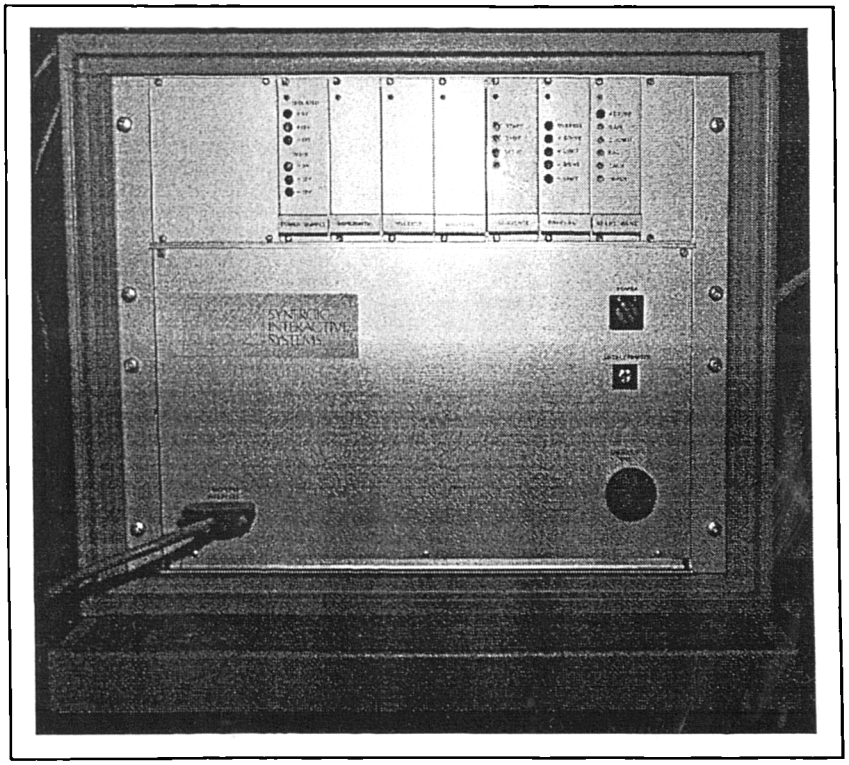
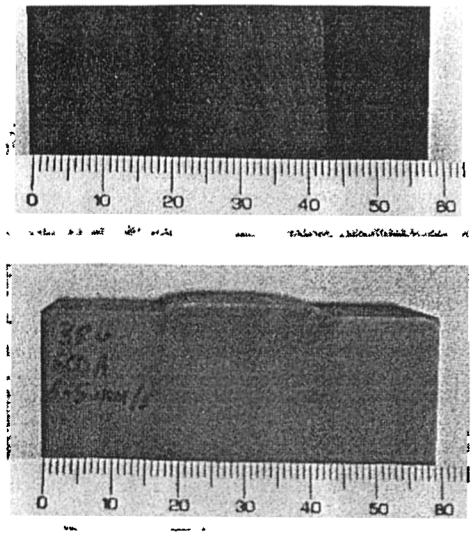
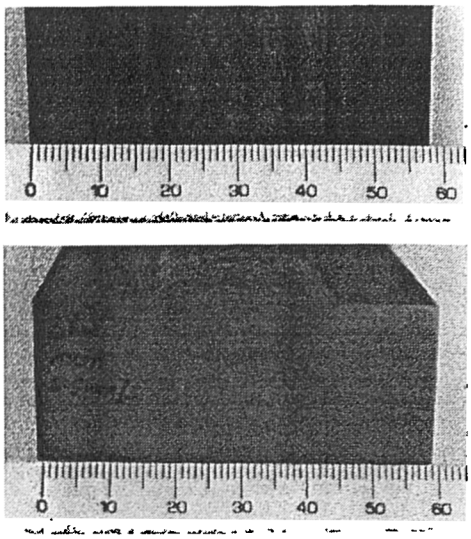


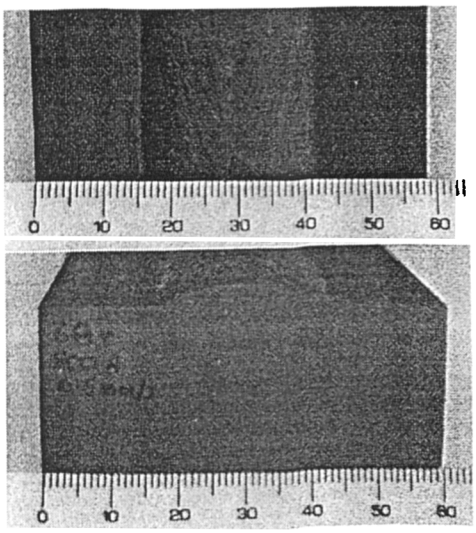
FIGURE 3.7
Synergic Systems Control Interface



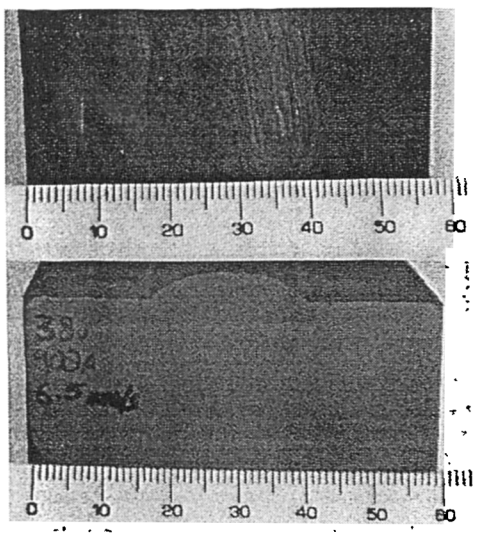
a) 38V, 600A, 6.5mm/s



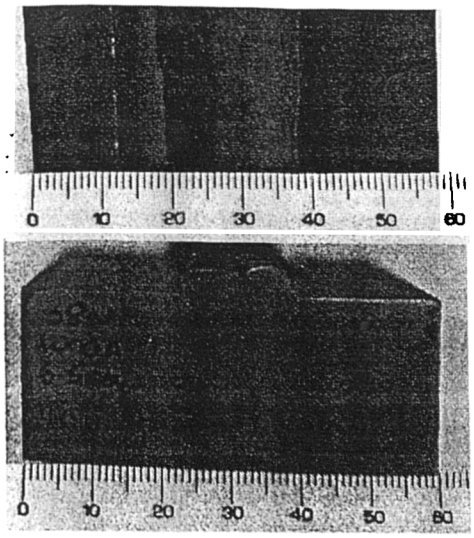
b) 38V, 700A, 6.5mm/s



c) 38V, 800A, 6.5mm/s

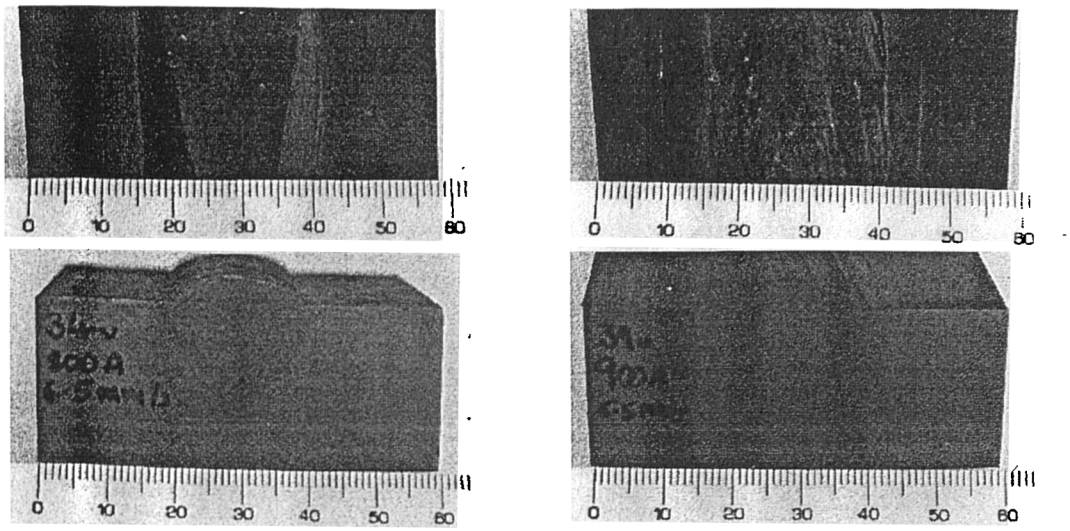


d) 38V, 900A, 6.5mm/s



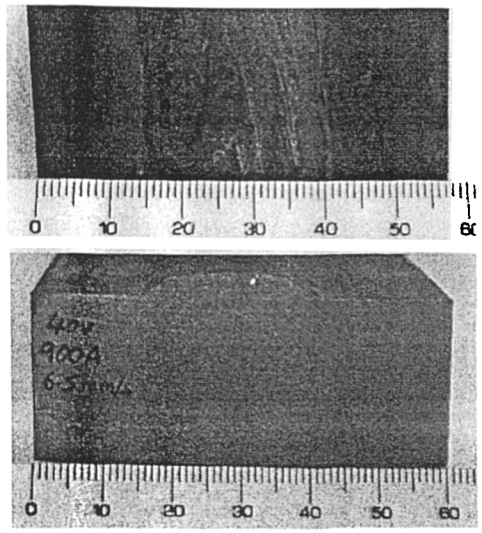
e) 38V, 1000A, 6.5mm/s

FIGURE 3.8 a - e
Weld Cross Sections - Bead Geometry for Varying Current

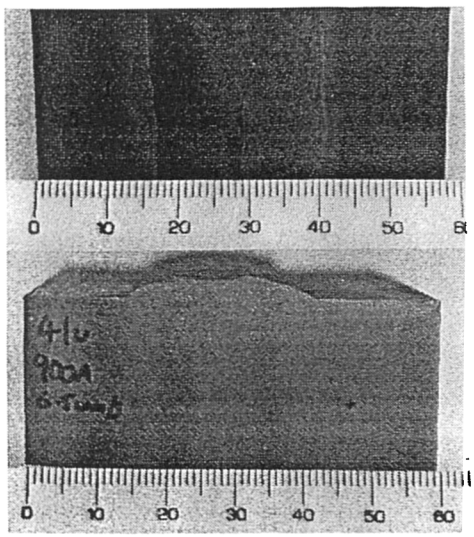


a) 34V, 800A, 6.5mm/s

b) 34V, 900A, 6.5mm/s



c) 40V, 900A, 6.5mm/s



d) 41V, 900A, 6.5mm/s

FIGURE 3.9 a - d
Weld Cross Sections - Bead Geometry for Varying Voltage

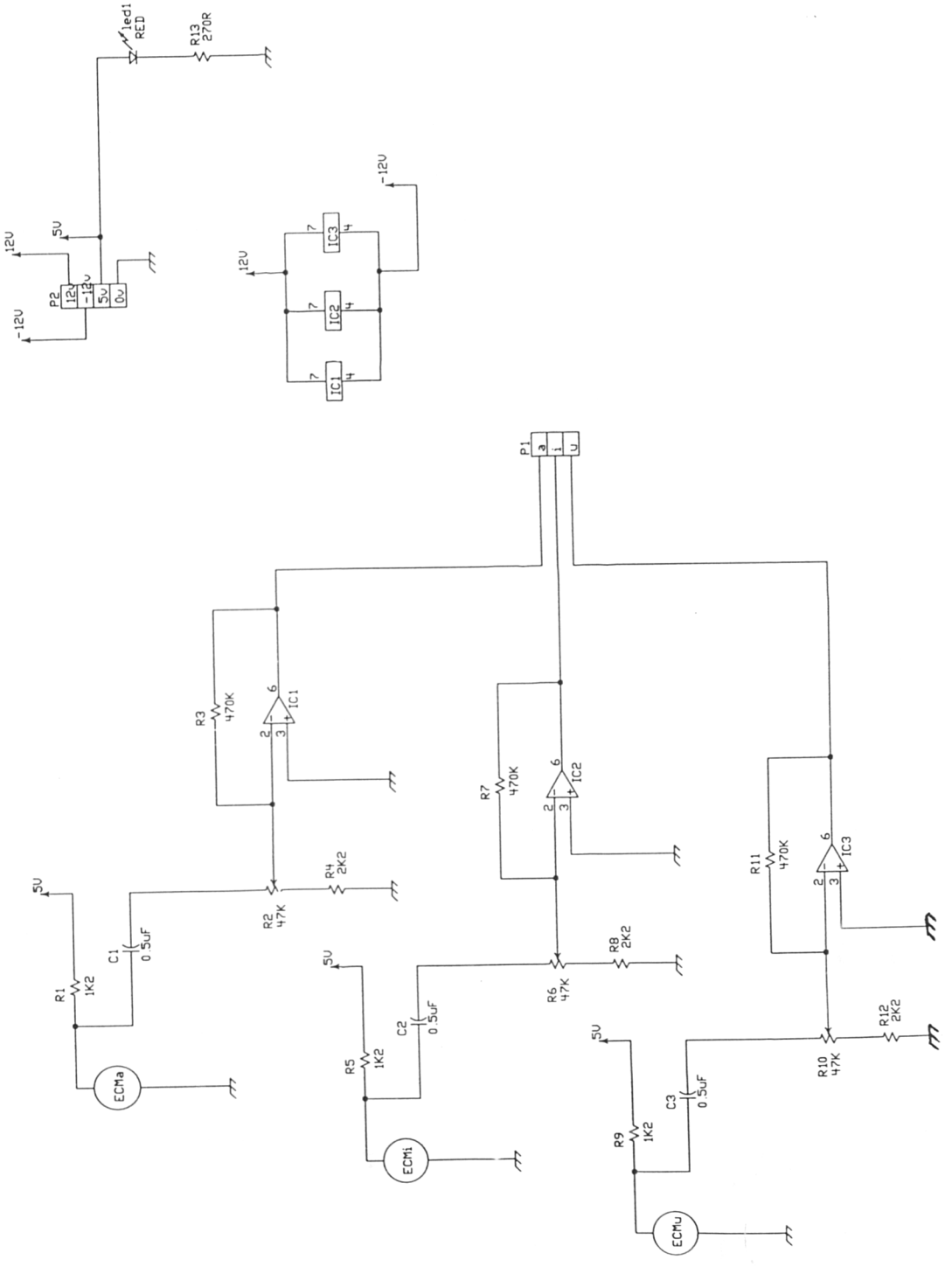


FIGURE 3.10
Microphone Assembly Circuit Schematic

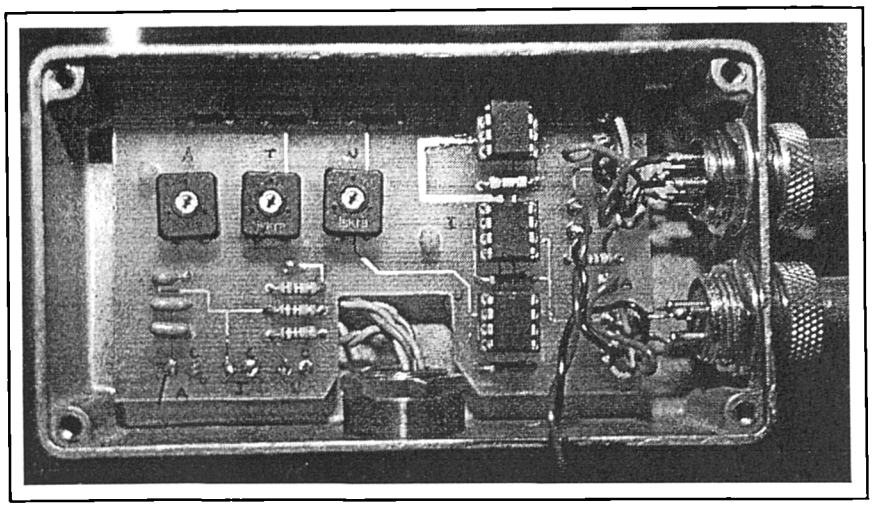


FIGURE 3.11
Microphone Pre-amplifier

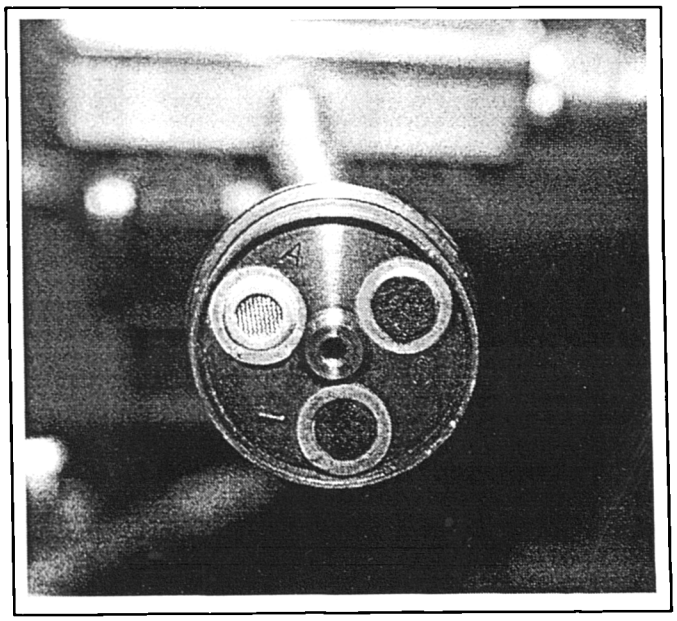


FIGURE 3.12
Transducer Head Arrangement

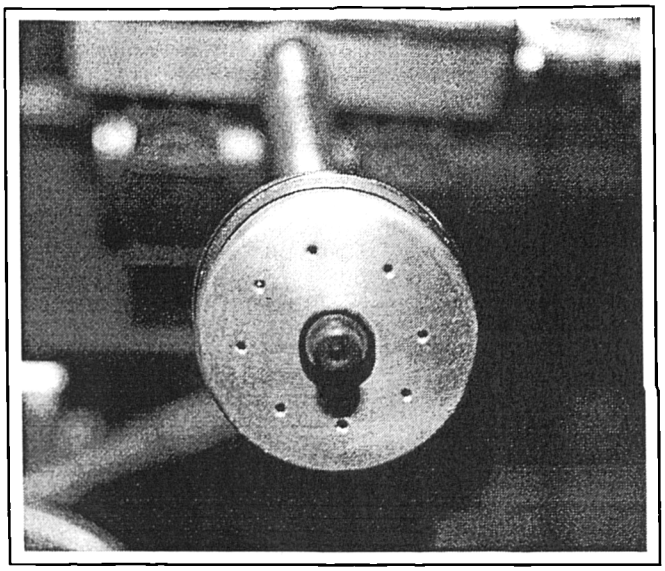


FIGURE 3.13
Transducer Head with Guard

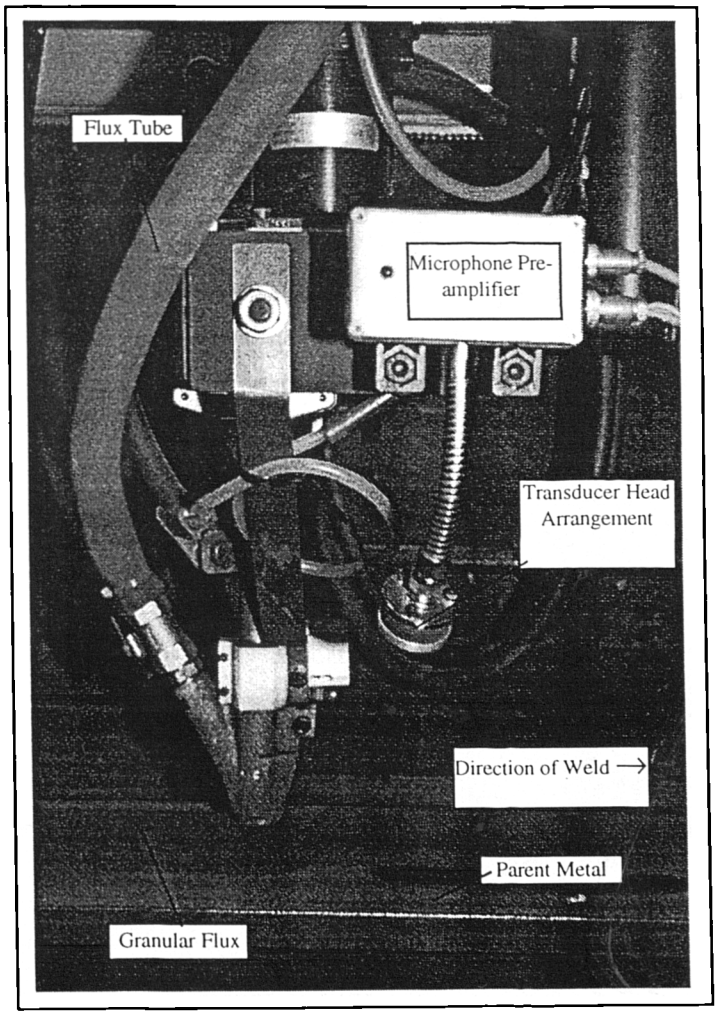


FIGURE 3.14
Position of Transducer Arrangement at Welding Head

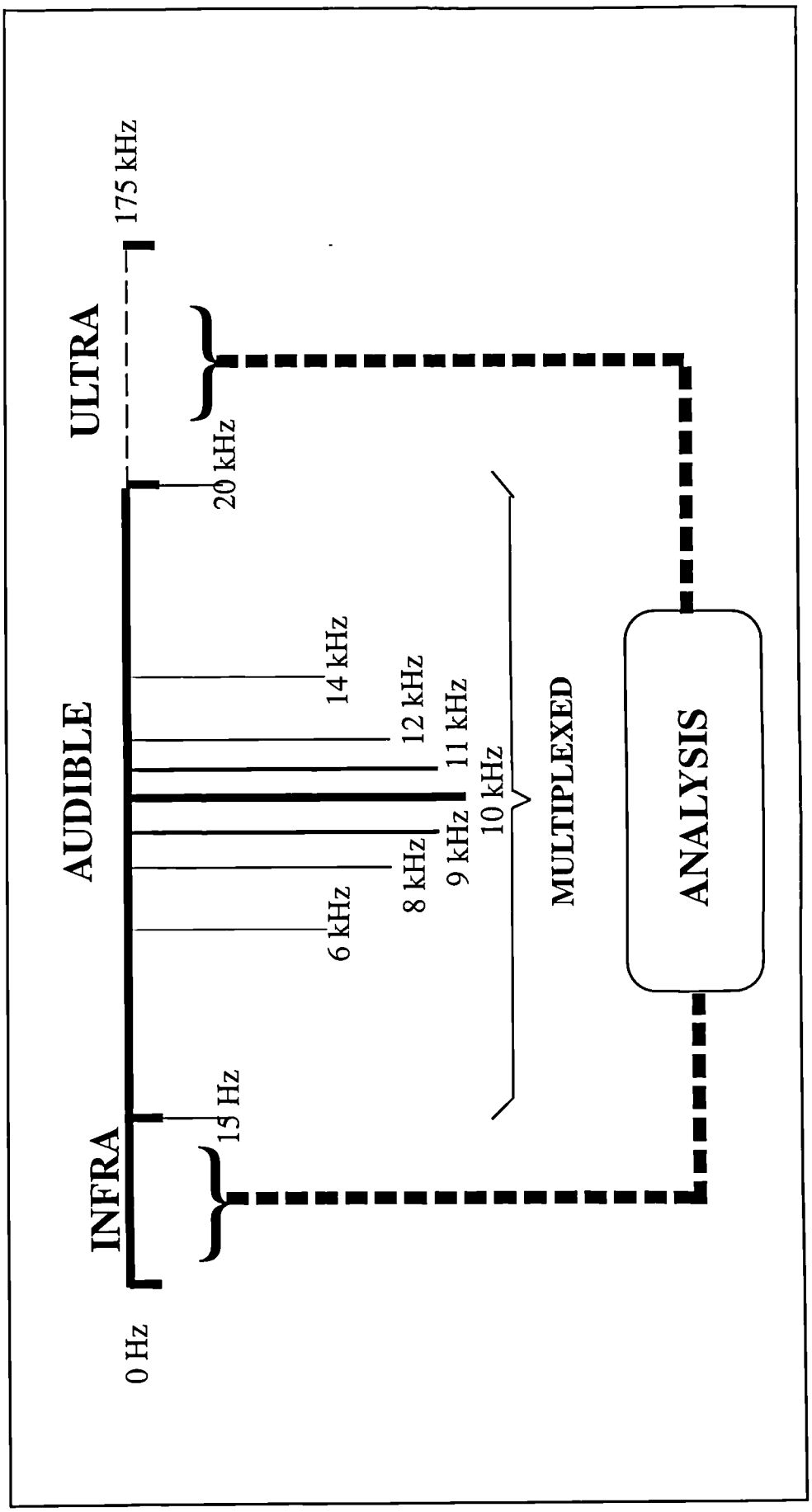


FIGURE 3.15
Schematic of Bandwidth Design

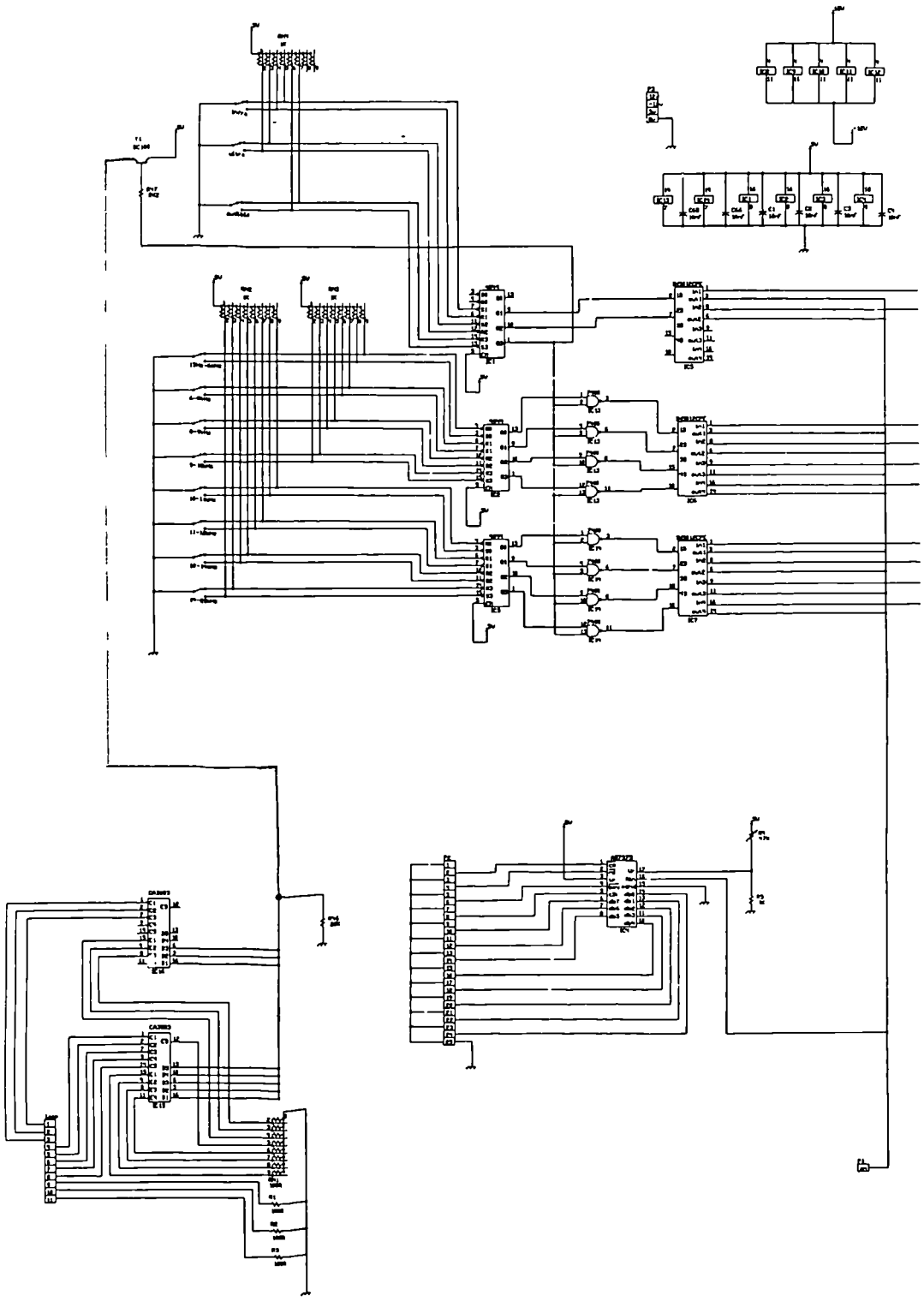


FIGURE 3.16
Audio Filter System Circuit Schematic

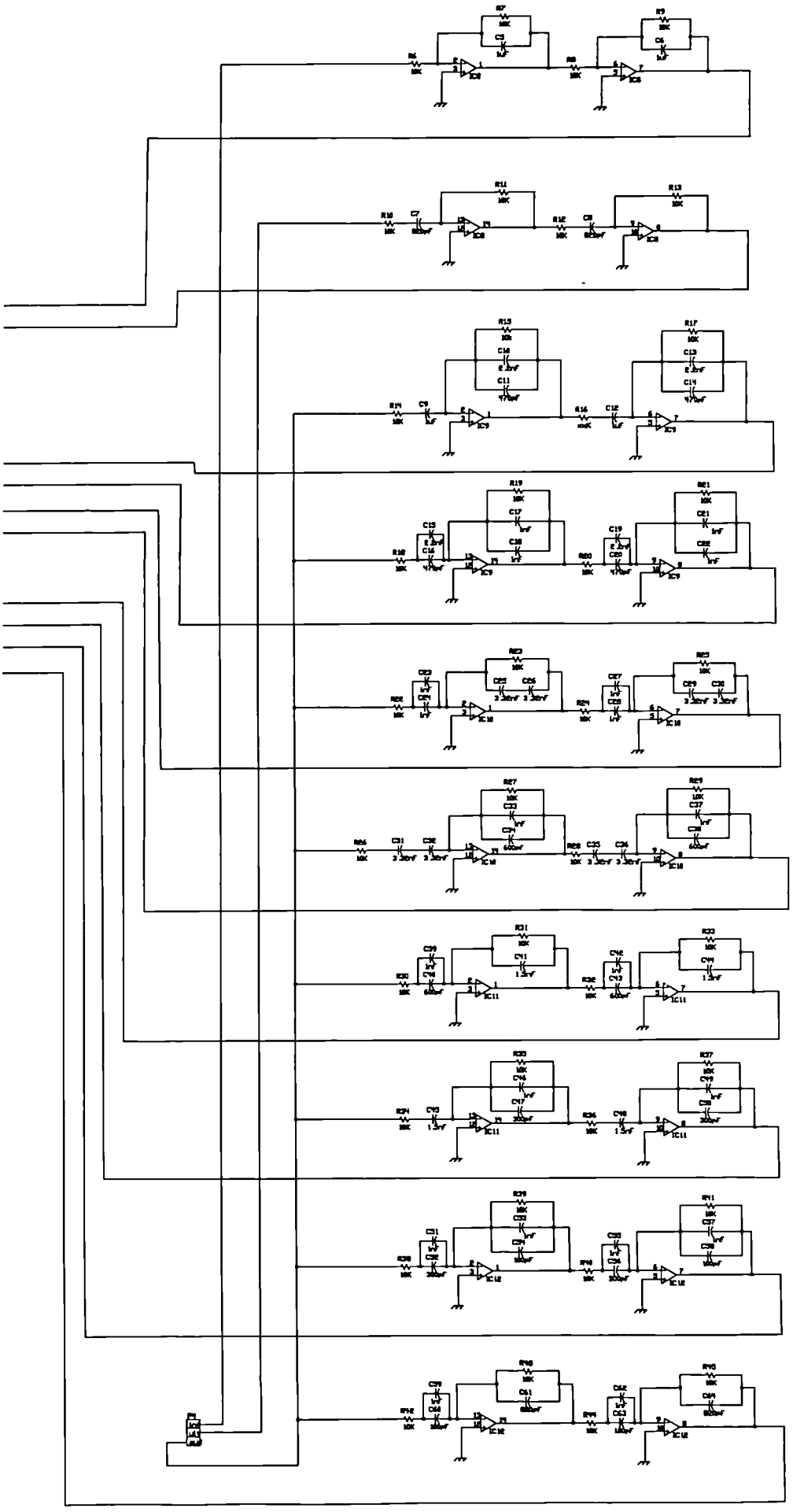


FIGURE 3.16
Audio Filter System Circuit Schematic

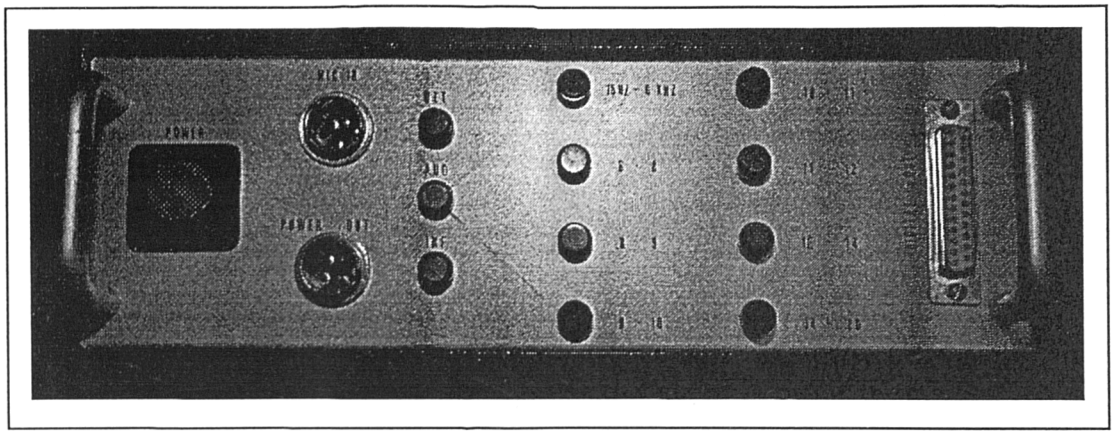


FIGURE 3.17
Audio Filter Box (External)

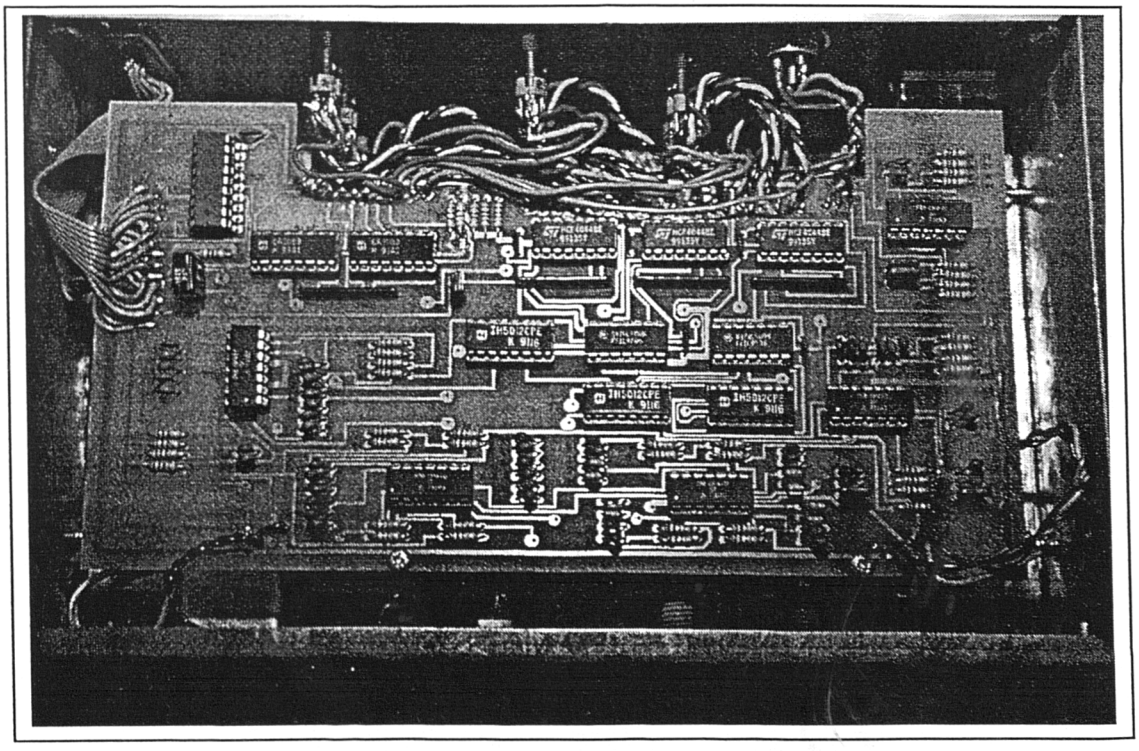


FIGURE 3.18
Audio Filter Box (Internal)

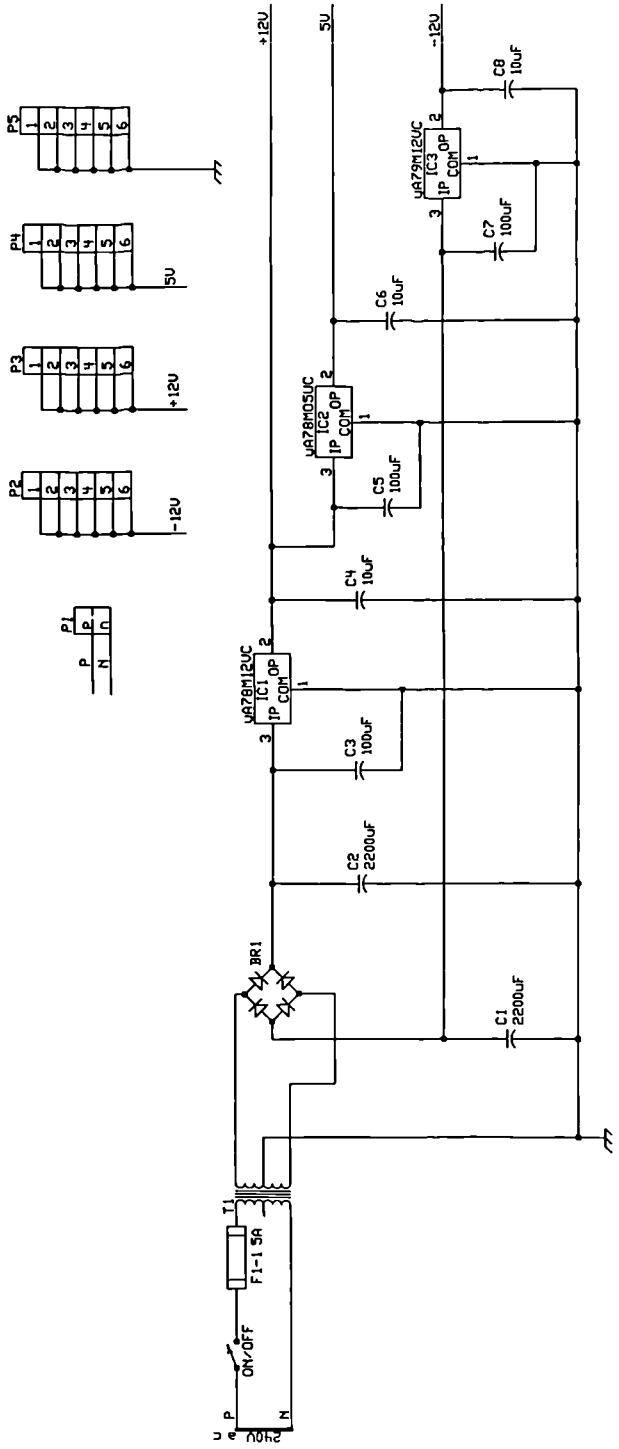


FIGURE 3.19
Syatem Power Supply Unit - Circuit Schematic

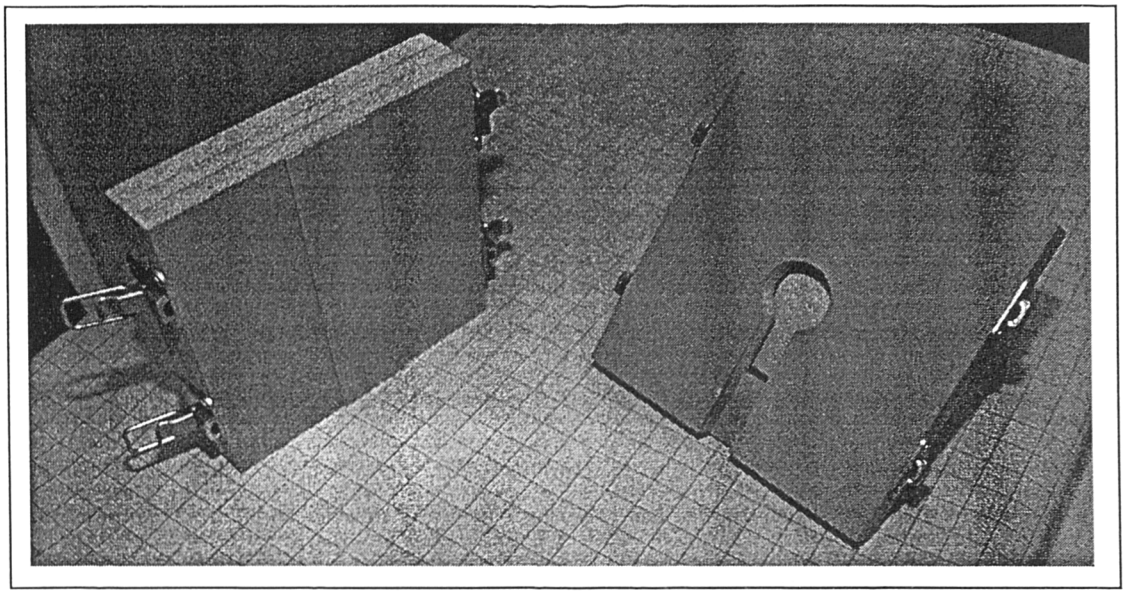


FIGURE 3.20
Purpose Built Anechoic Chamber (Courtesy of Francis Hughes)

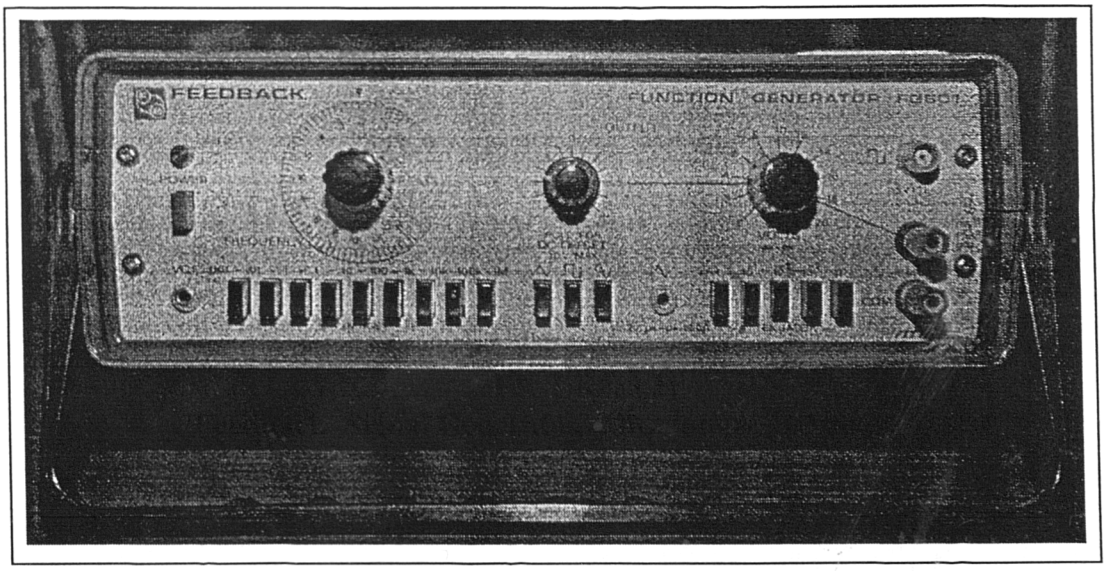


FIGURE 3.21
'Feedback' Signal Generator (Microphone Response Tests)

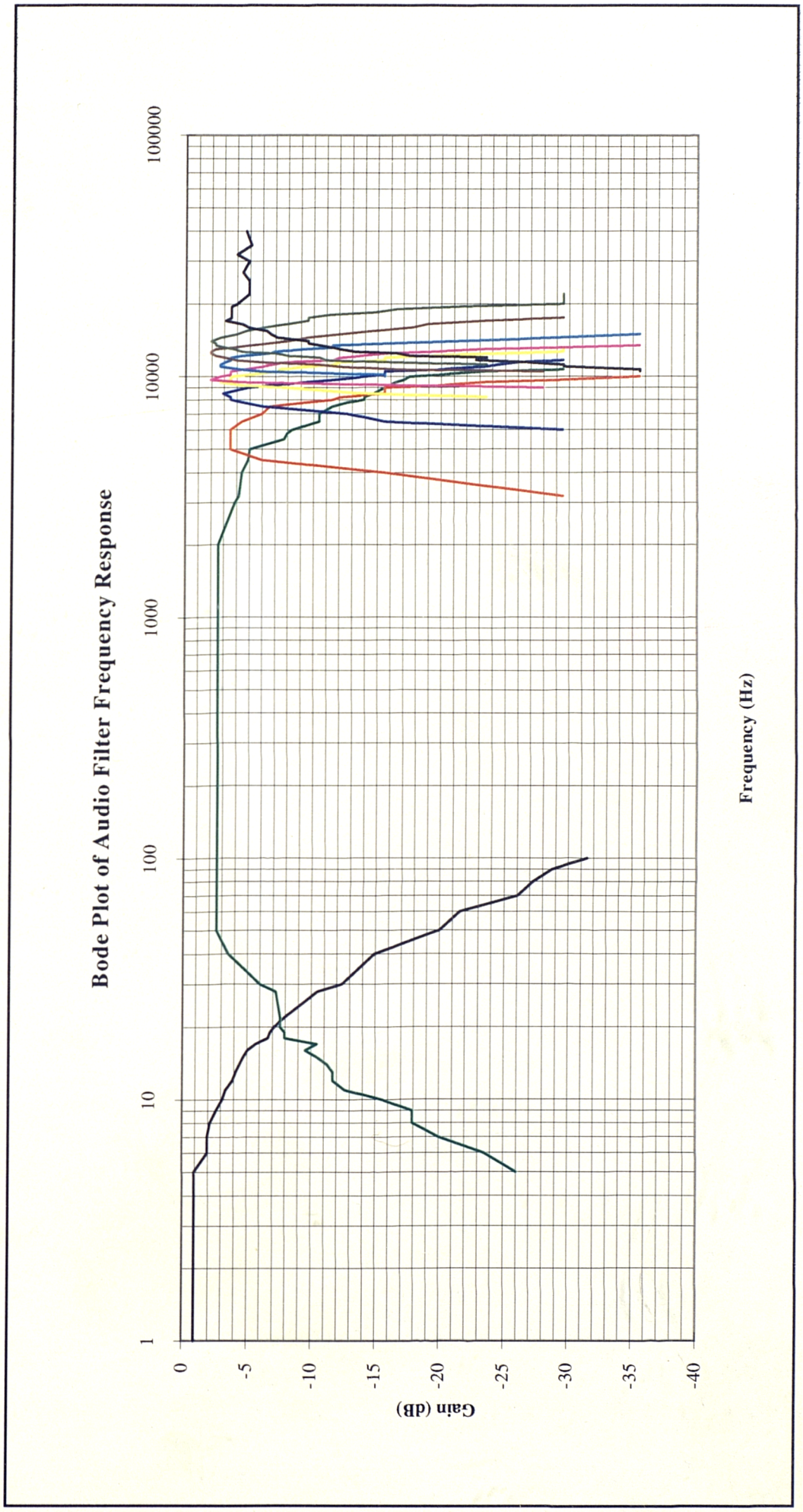


FIGURE 3.22
Audio Filter System Frequency Response (Log Scale)

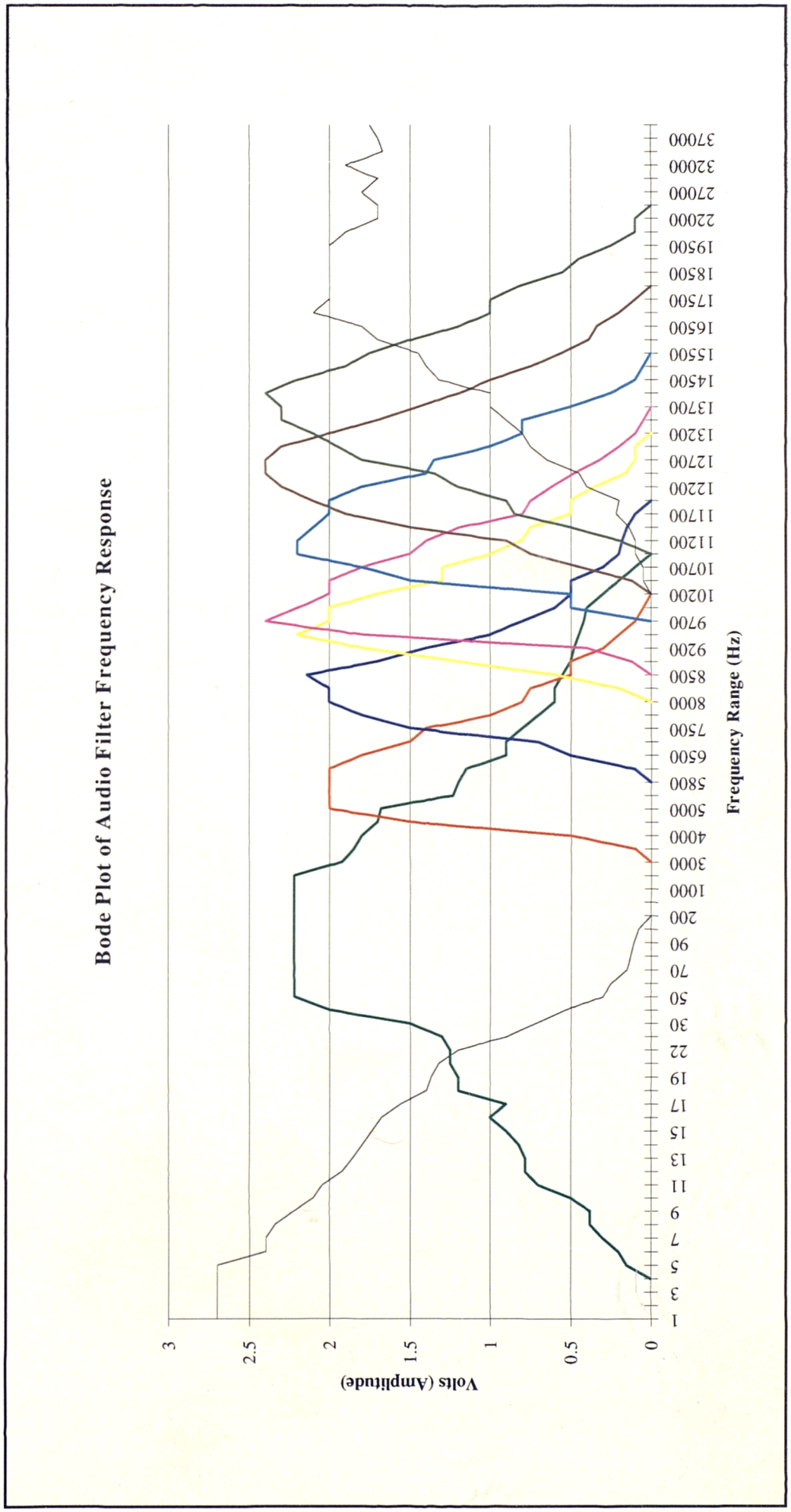


FIGURE 3.22a
Audio Filter System Frequency Response (Linear Scale)

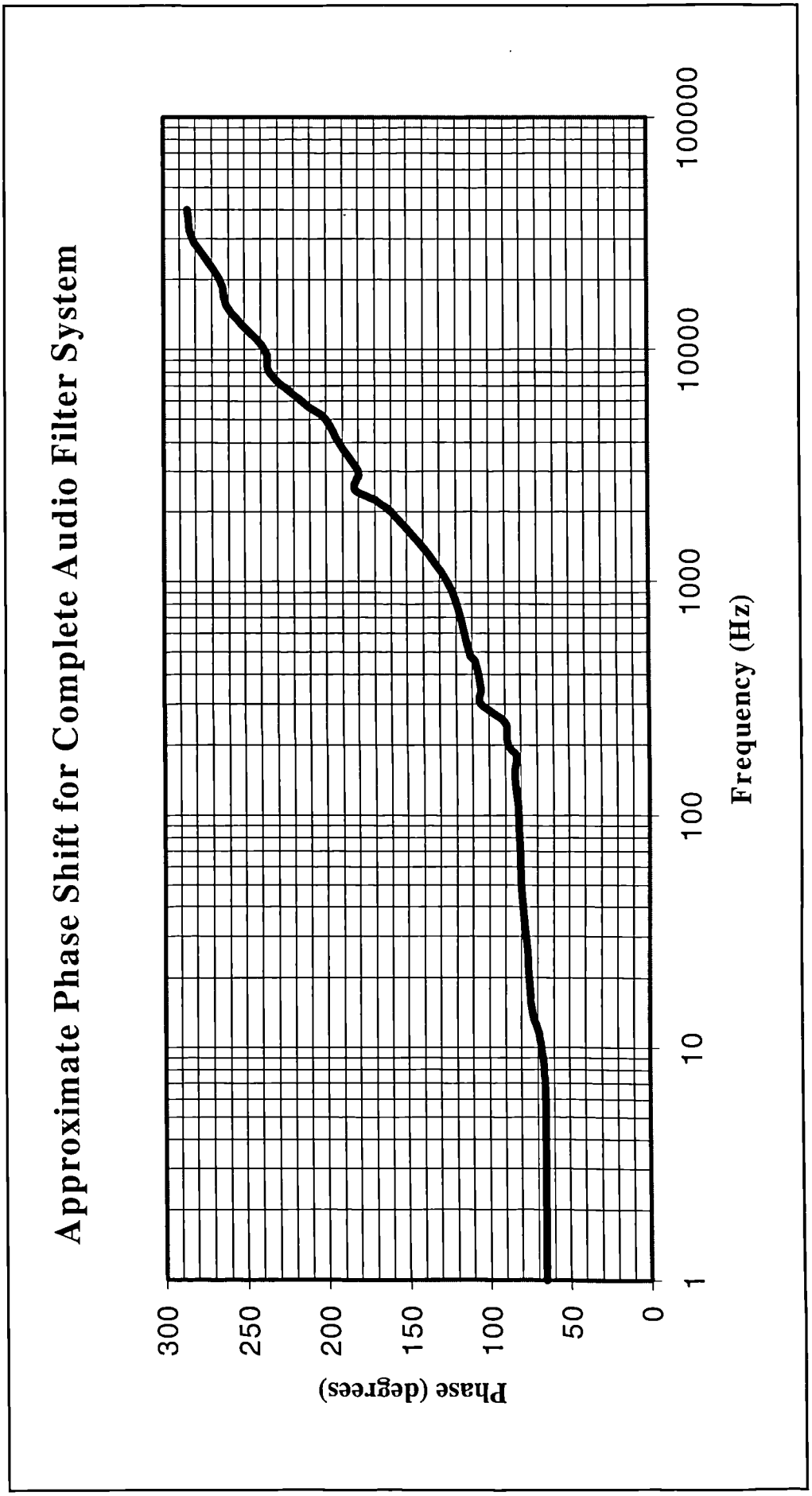


FIGURE 3.23
Audio Filter System Phase Response

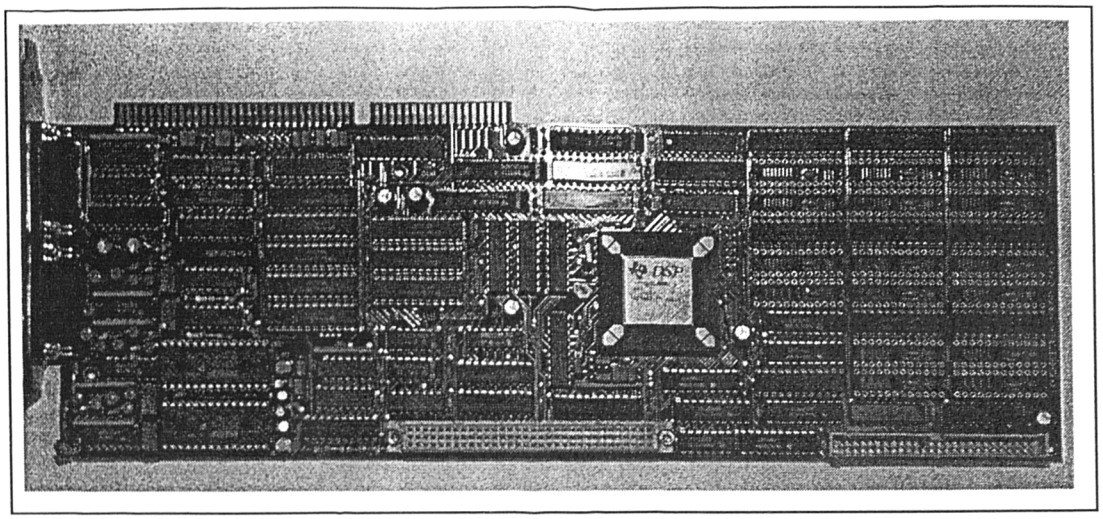


FIGURE 3.24
TMS320C30 DSP Card (Motherboard Development System)

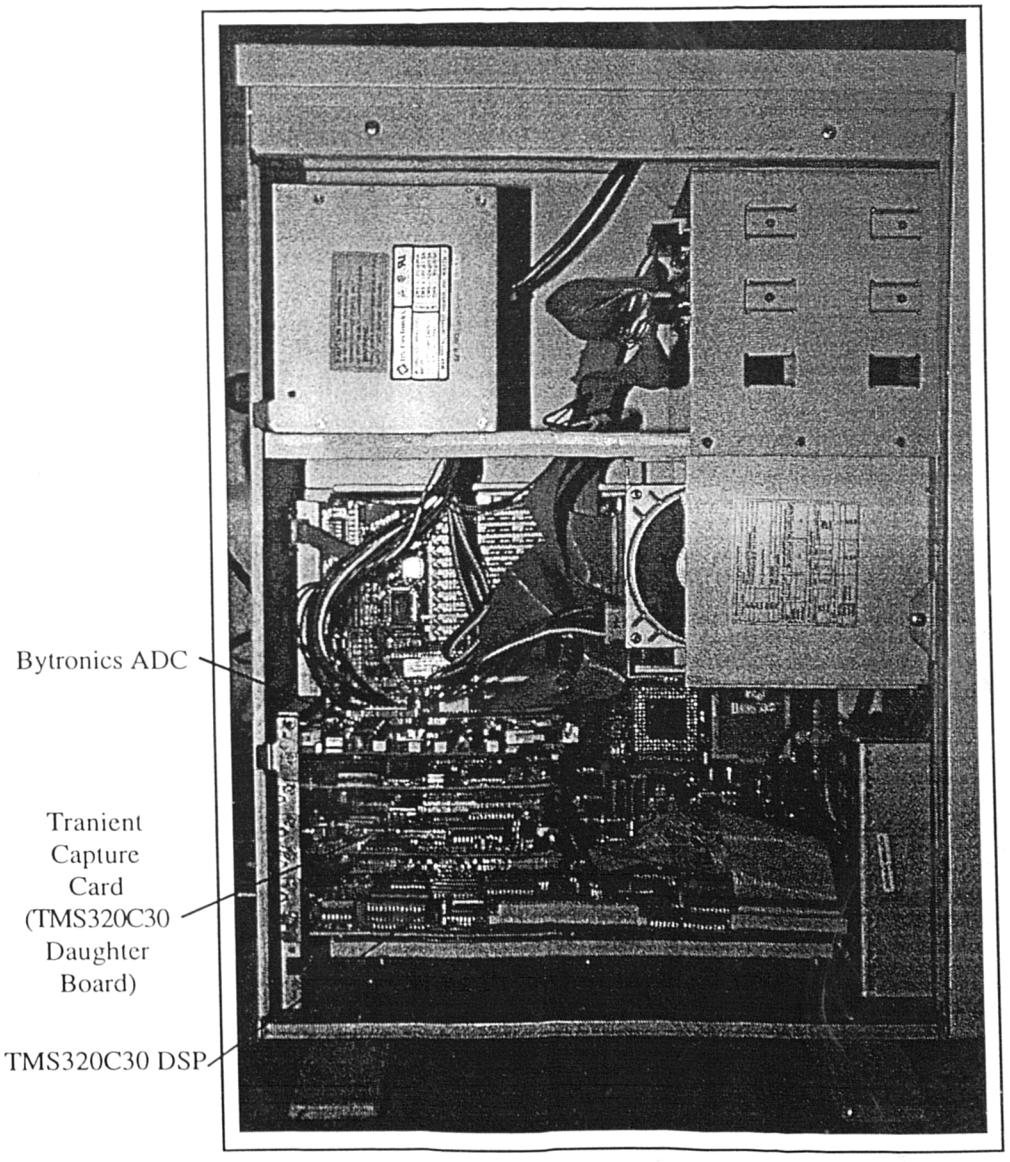


FIGURE 3.25
Host PC (Monitoring / DSP)

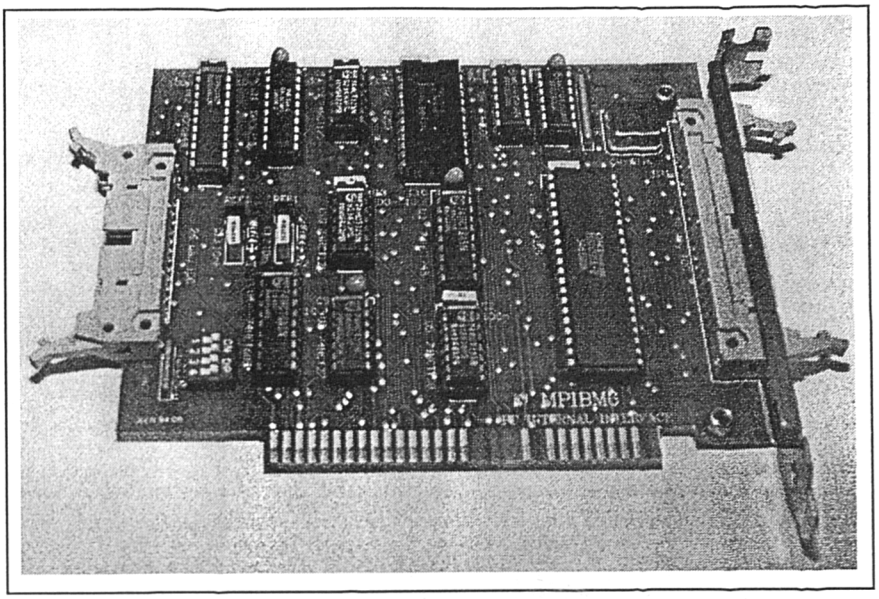


FIGURE 3.26
Weld Monitoring ADC Card

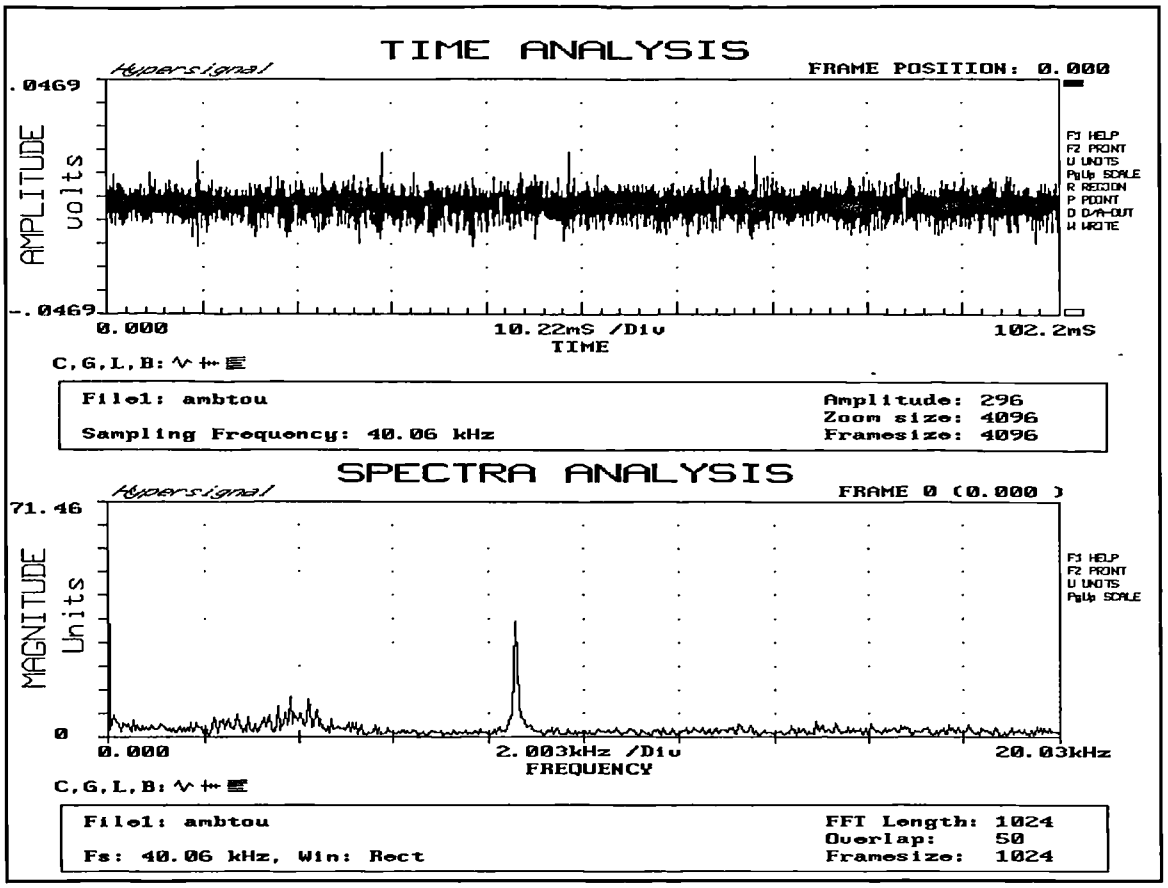


FIGURE 3.27
Time / Amplitude & FFT - Power Source Noise

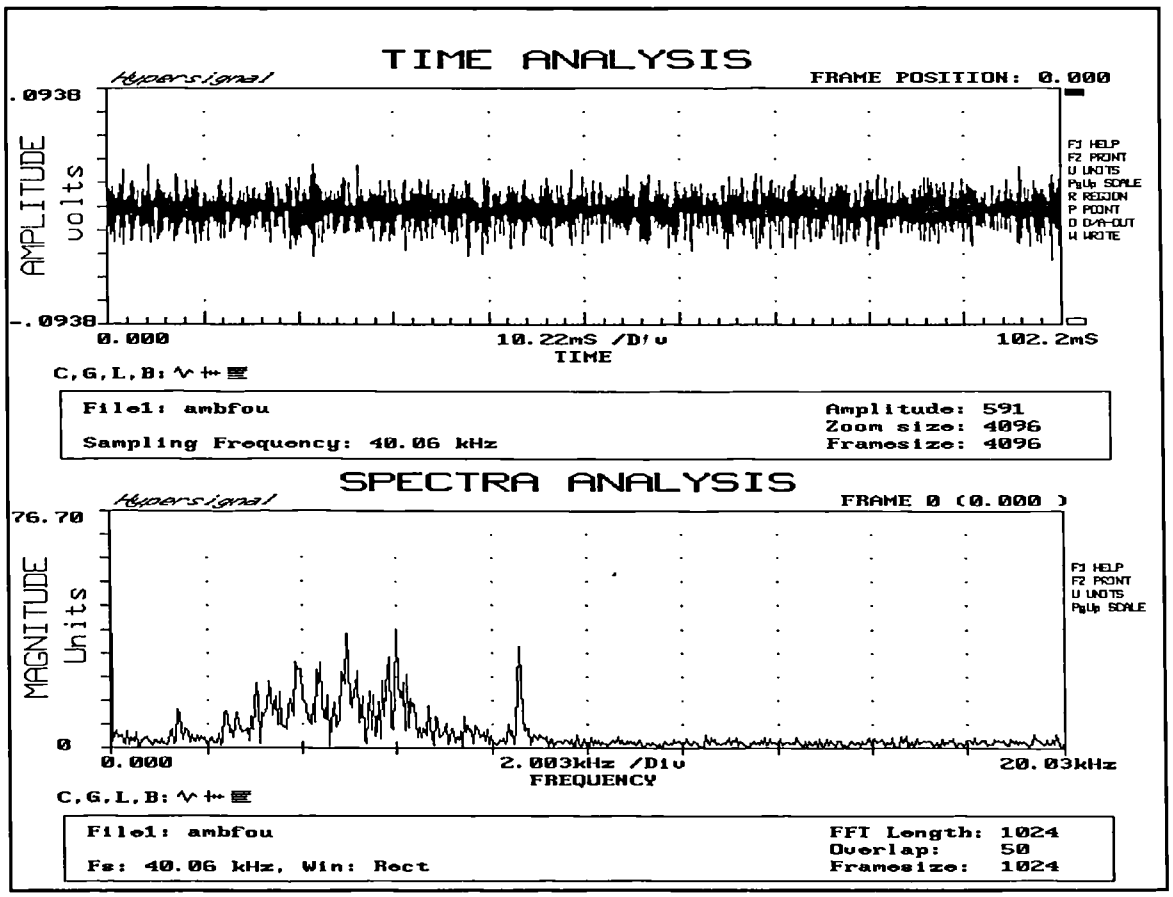


FIGURE 3.28
Time / Amplitude & FFT - Extractor Fan Noise

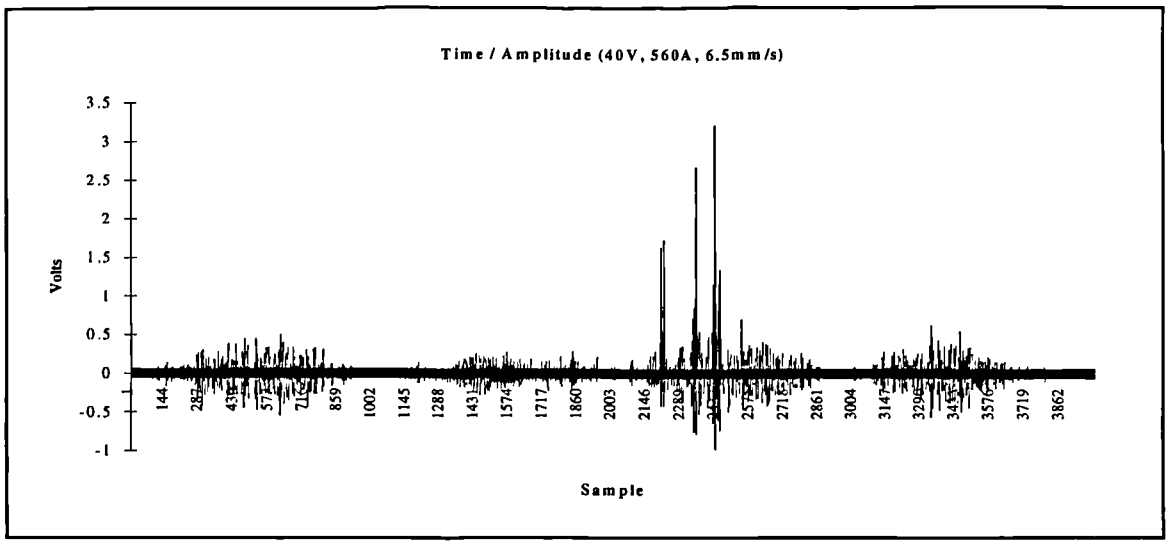


FIGURE 3.29
Time / Amplitude Trace - Optimum Weld Voltage

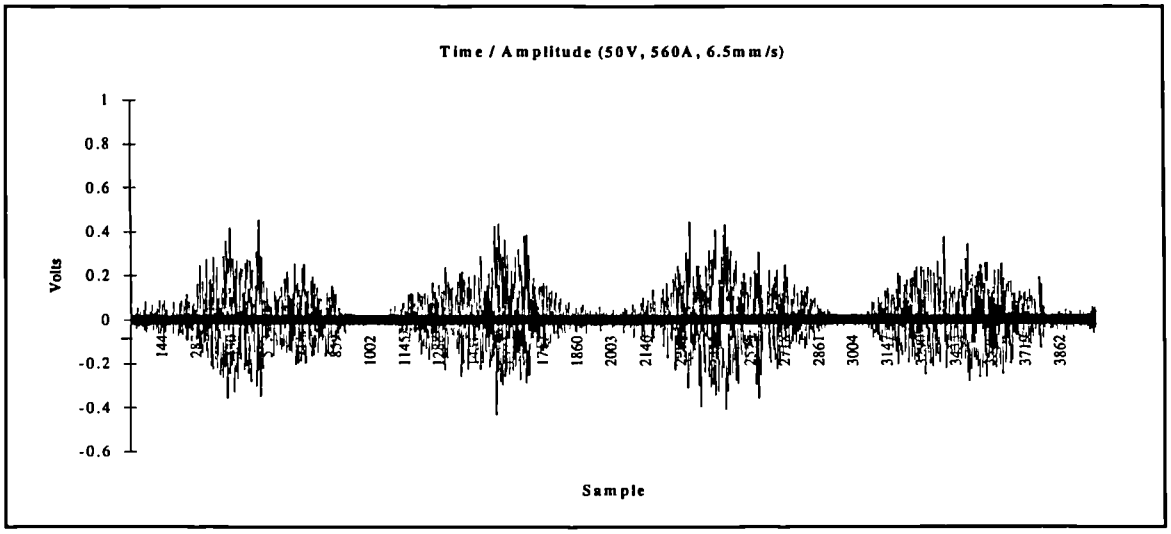


FIGURE 3.30
Time / Amplitude Trace - High Weld Voltage

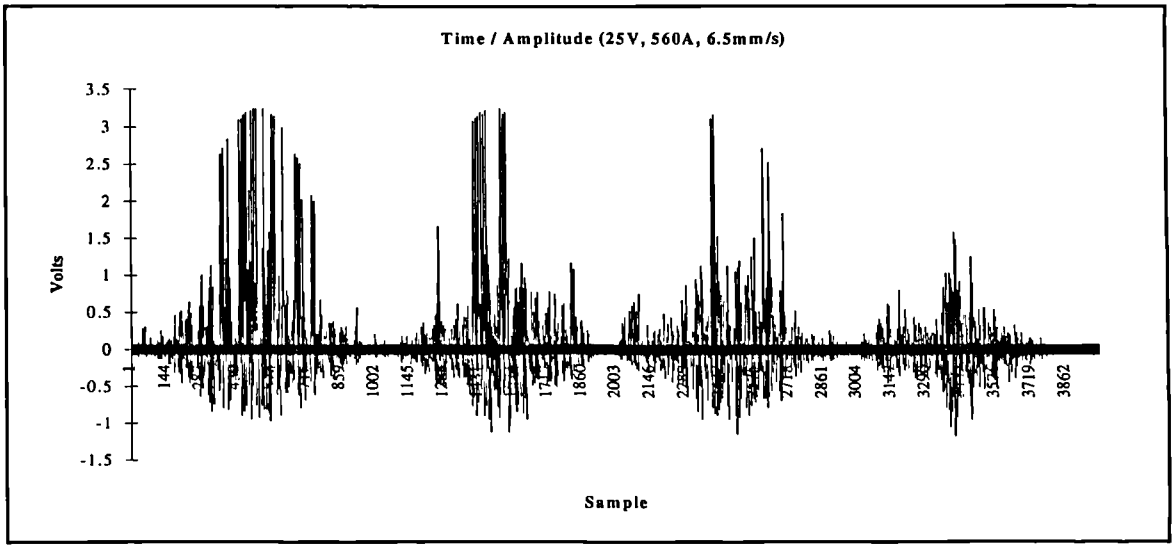


FIGURE 3.31
Time / Amplitude Trace - Low Weld Voltage

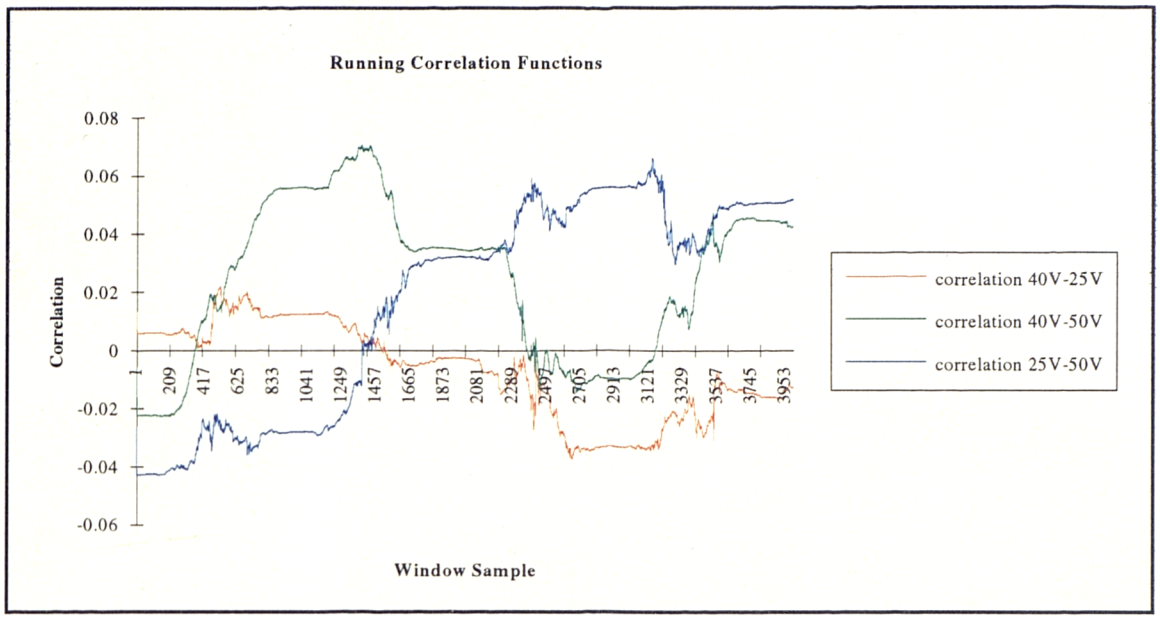


FIGURE 3.32
Voltage Correlations (Raw Signals)

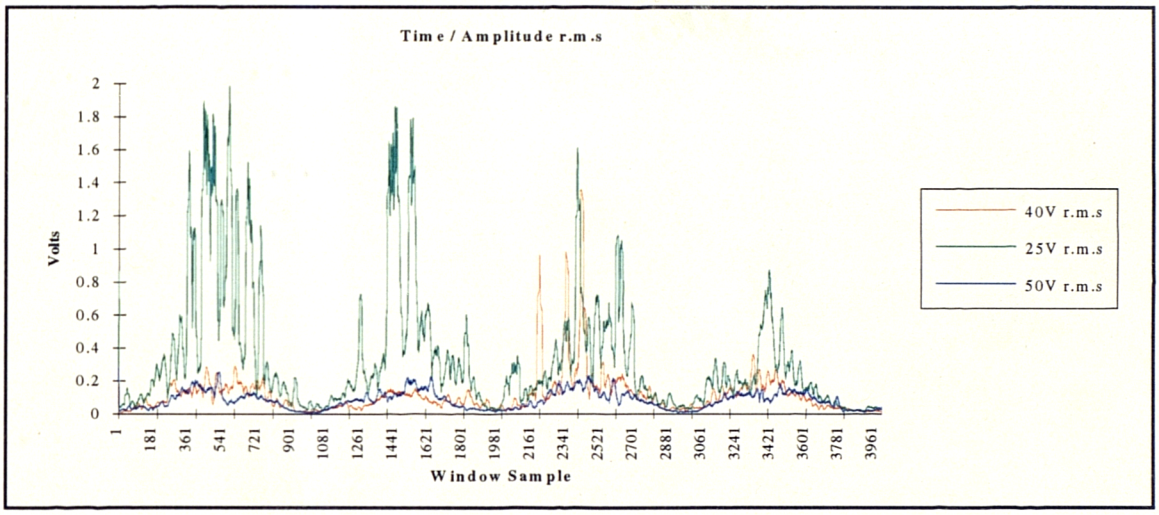


FIGURE 3.33
Voltage r.m.s Levels

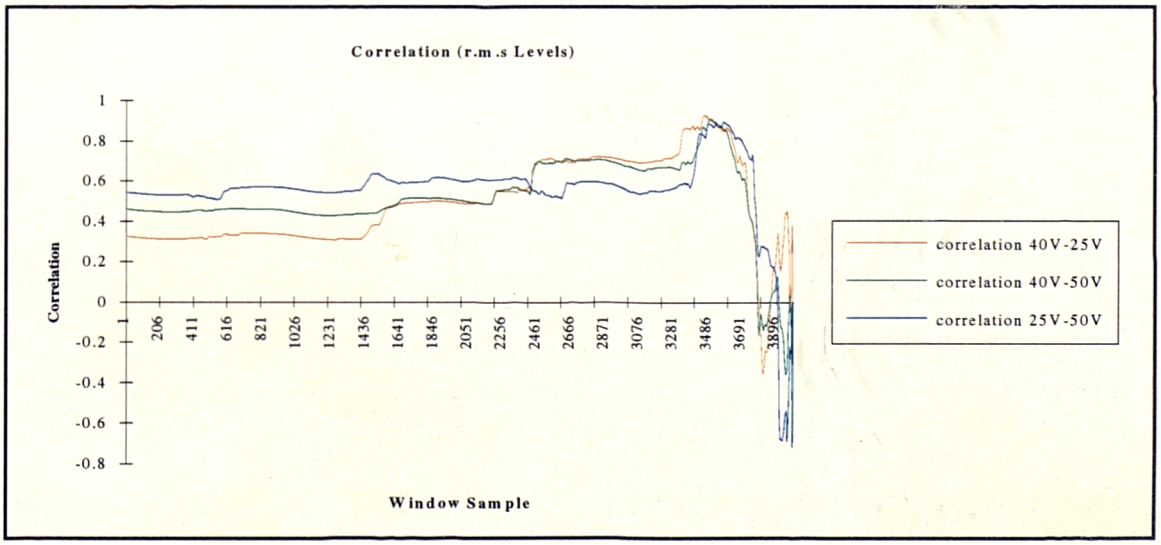


FIGURE 3.34
Voltage Correlations (r.m.s Levels)

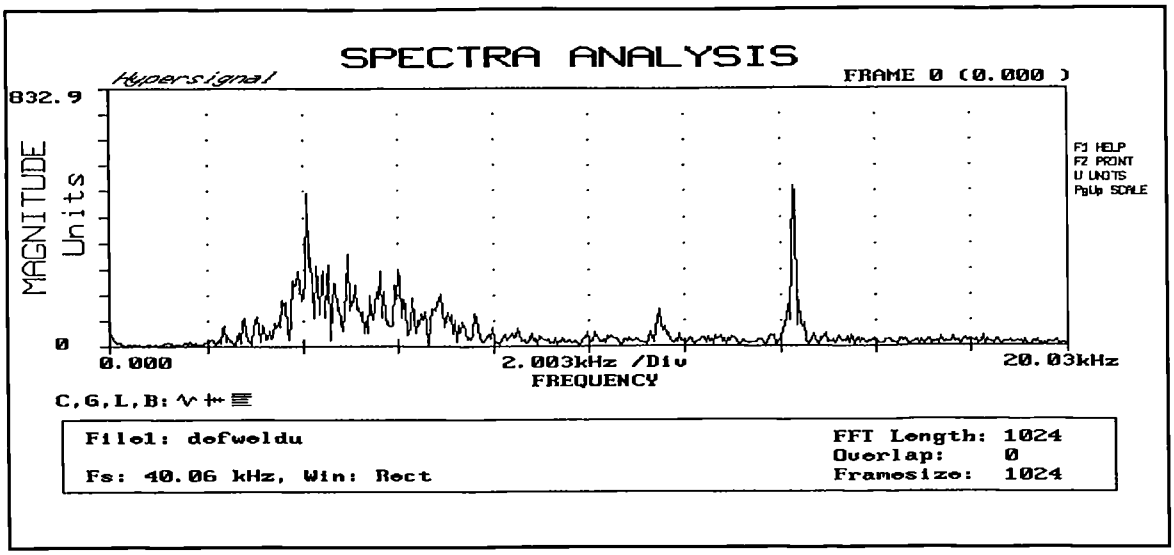


FIGURE 3.35
1024 Point FFT - Optimum Weld Voltage

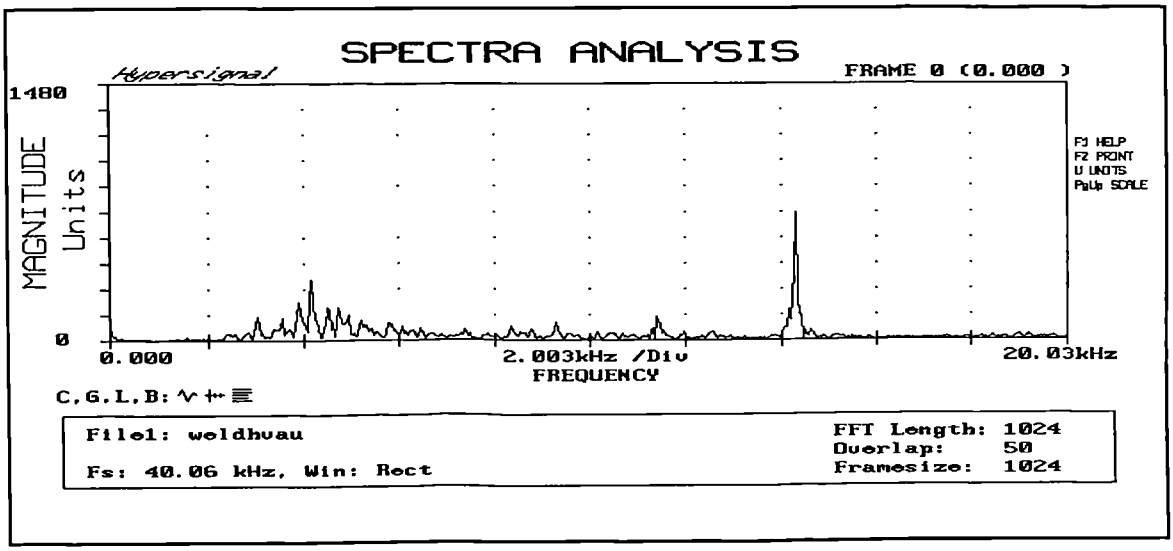


FIGURE 3.36
1024 Point FFT - High Weld Voltage

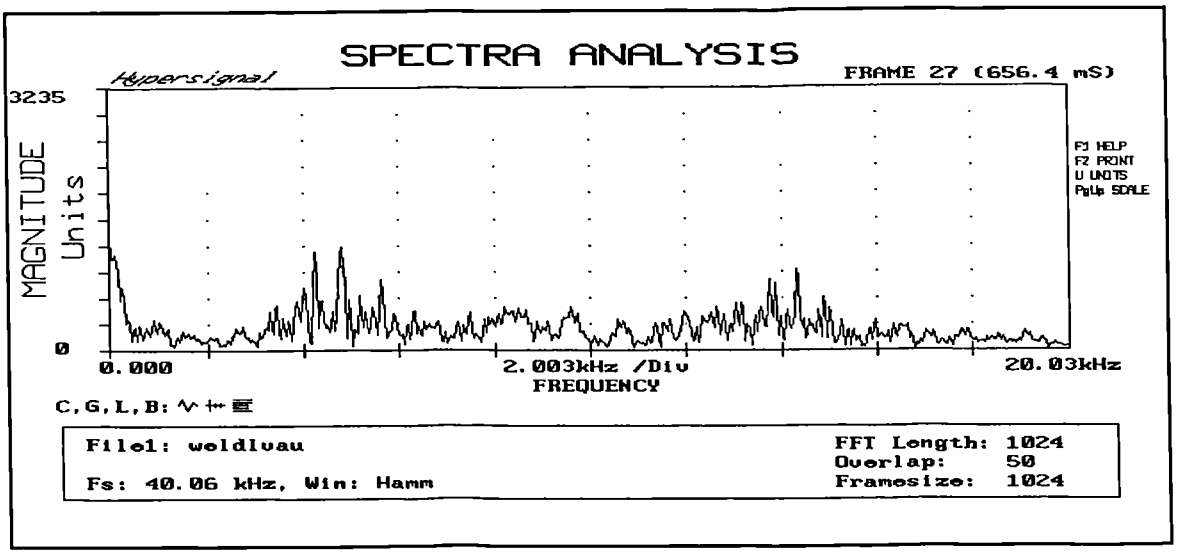


FIGURE 3.37
1024 Point FFT - Low Weld Voltage

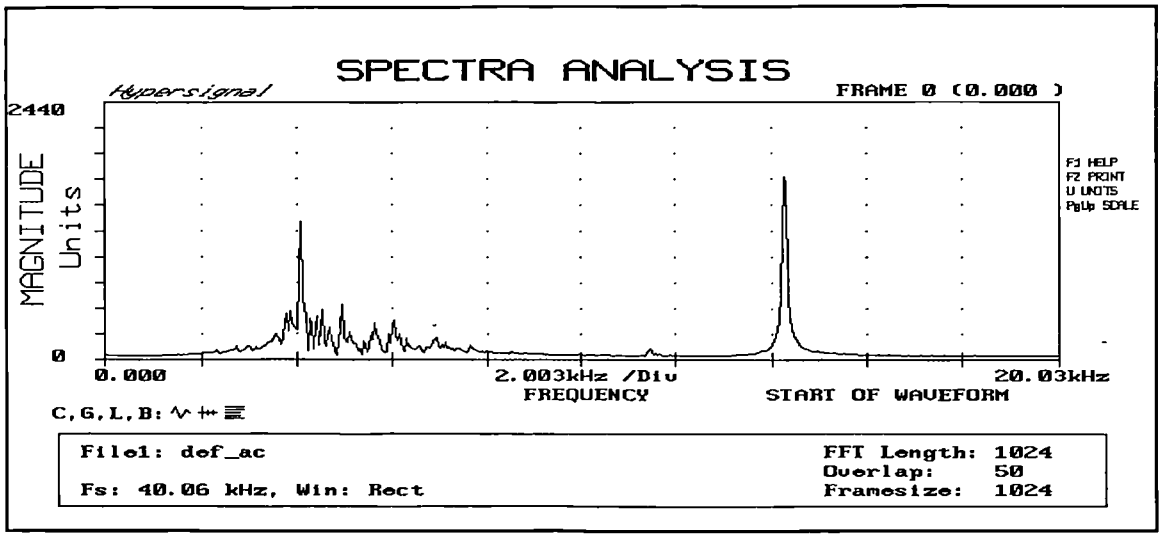


FIGURE 3.38
Auto-Correlation (1024Point FFT) - Optimum Weld Voltage

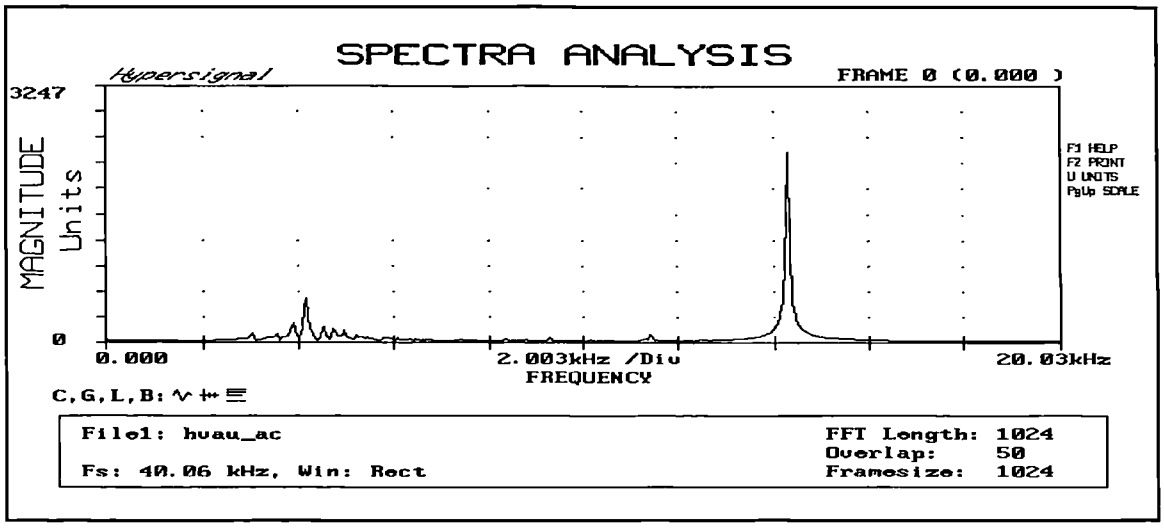


FIGURE 3.39
Auto-Correlation (1024Point FFT) - High Weld Voltage

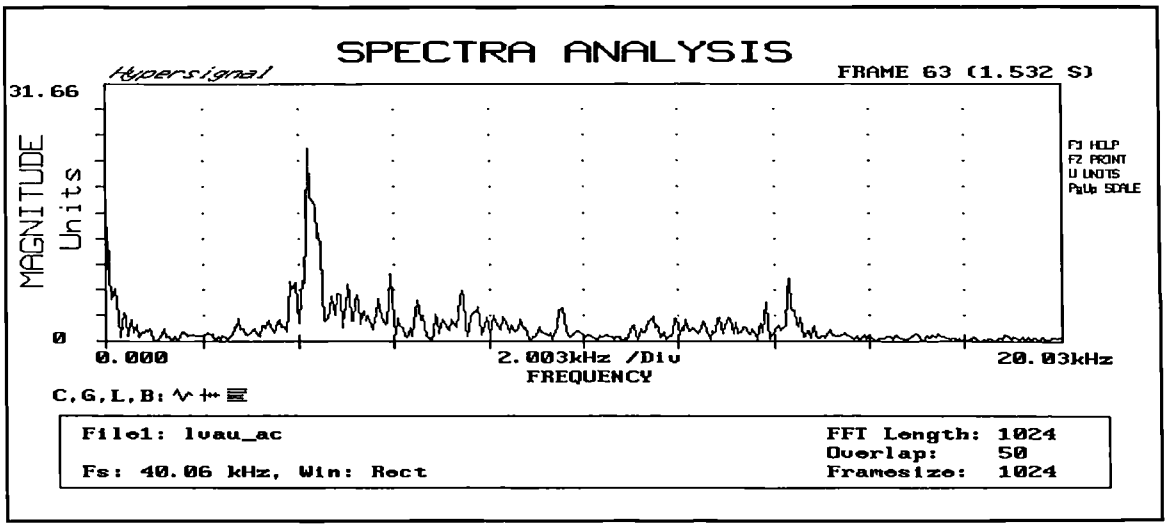


FIGURE 3.40
Auto-Correlation (1024Point FFT) - Low Weld Voltage

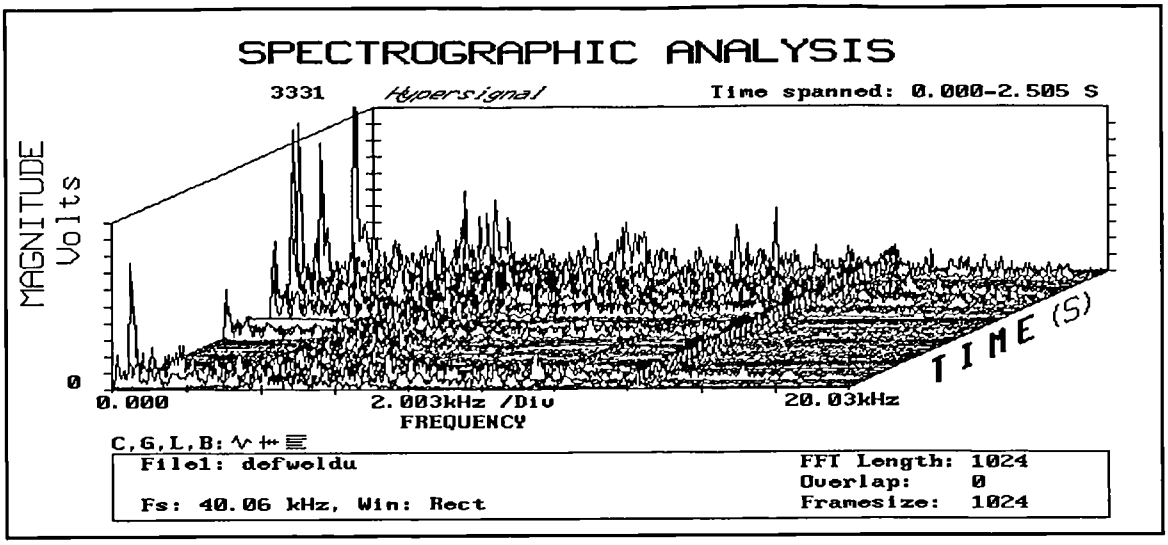


FIGURE 3.41
Time Variant 1024 Point FFT - Optimum Weld Voltage

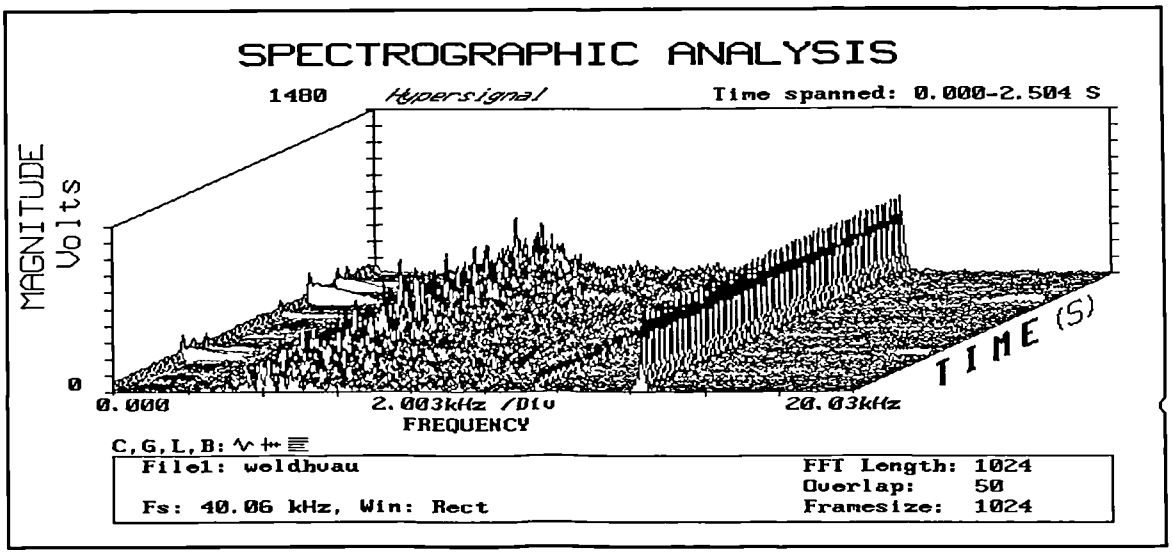


FIGURE 3.42
Time Variant 1024 Point FFT - High Weld Voltage

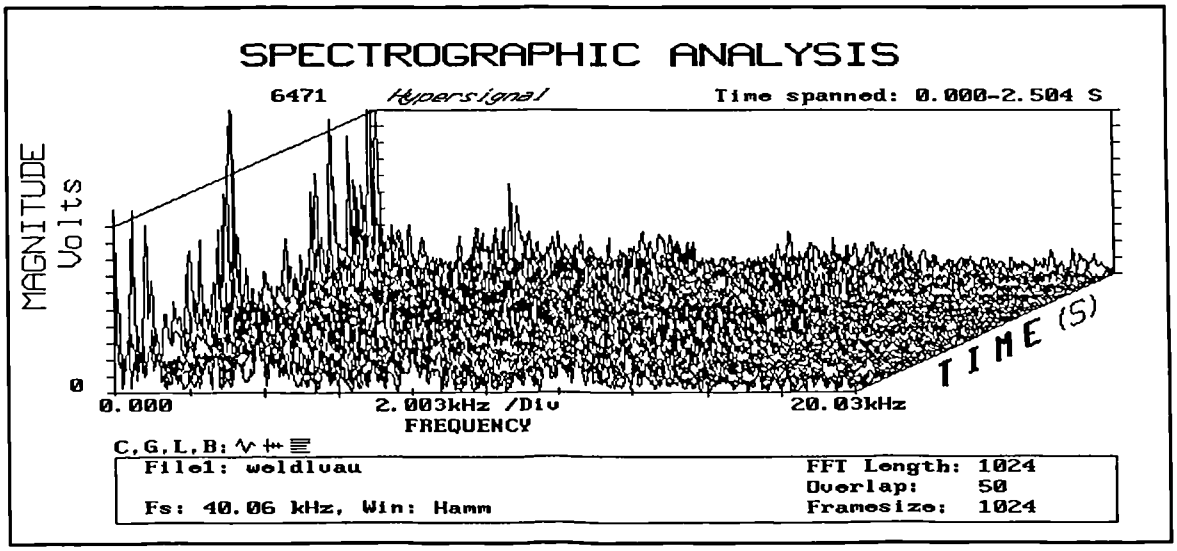


FIGURE 3.43
Time Variant 1024 Point FFT - Low Weld Voltage

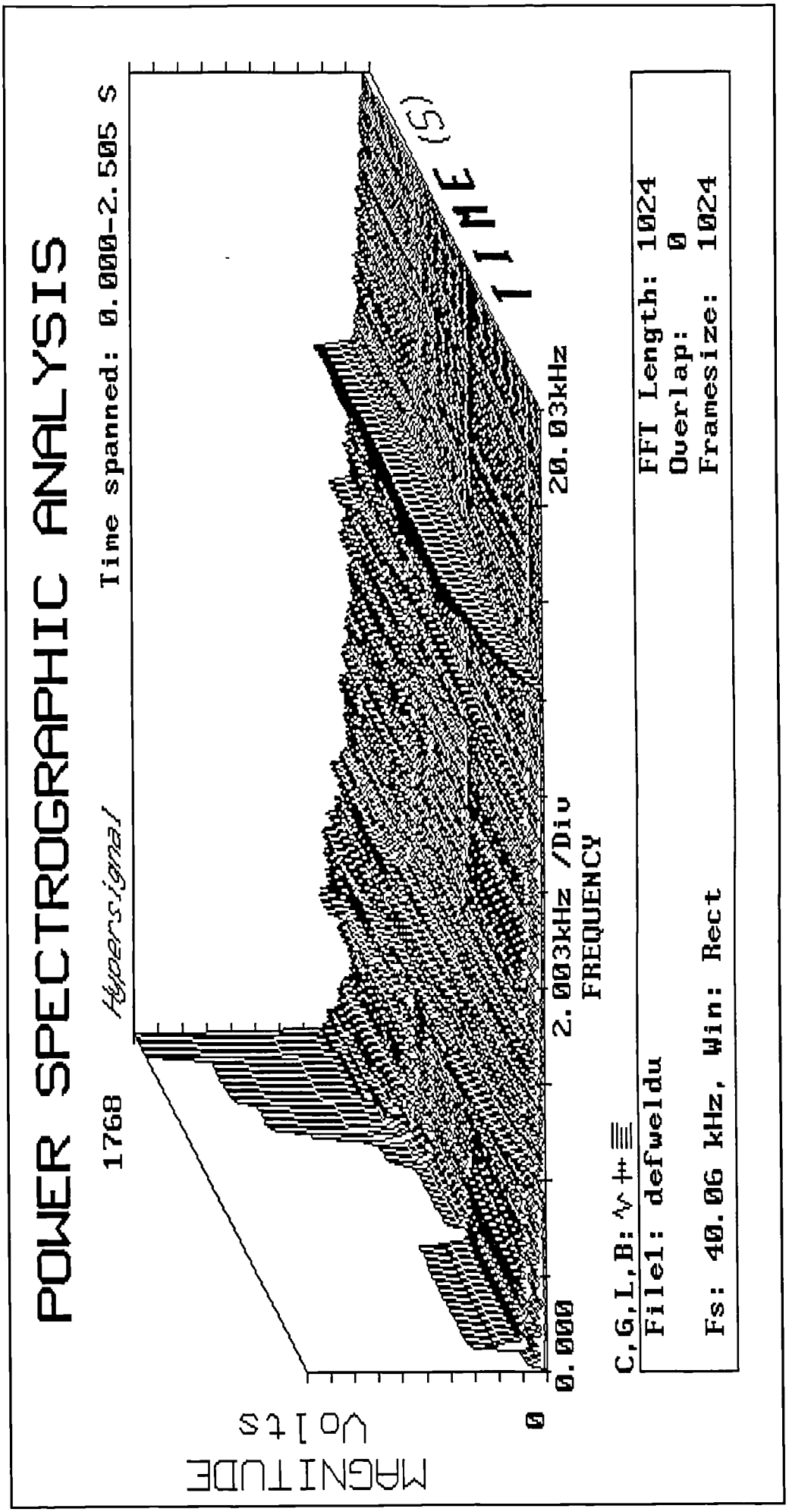


FIGURE 3.44
Power Spectra - Optimum Voltage (running average method)

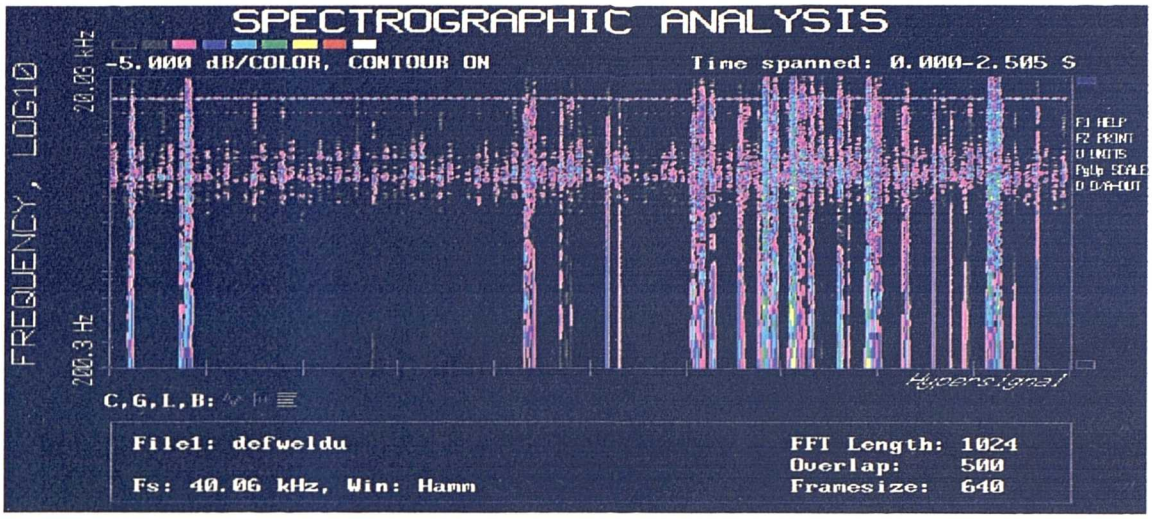


FIGURE 3.45
Spectral Contour Map - Optimum Weld Voltage

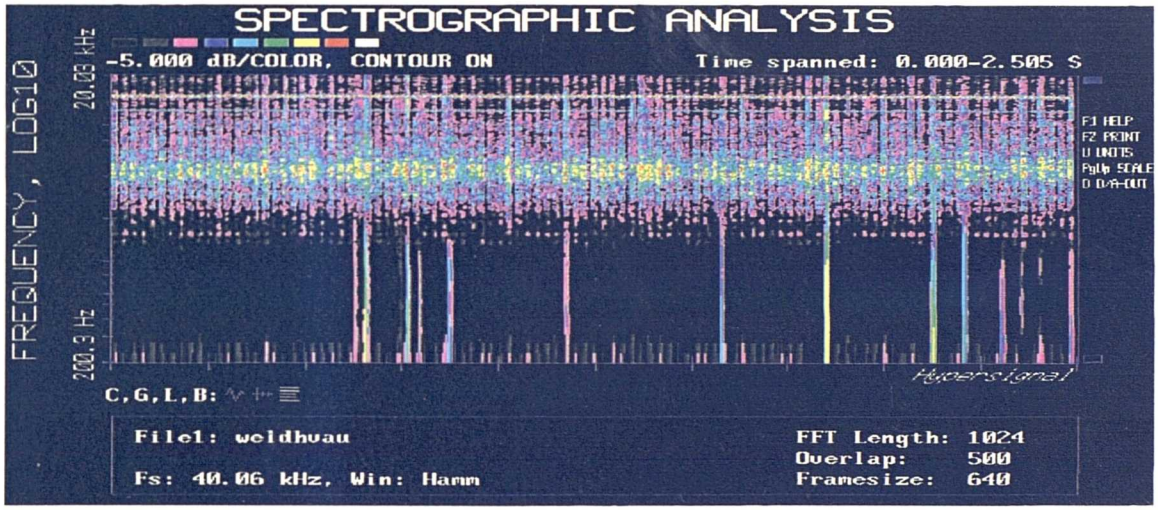


FIGURE 3.46
Spectral Contour Map - High Weld Voltage

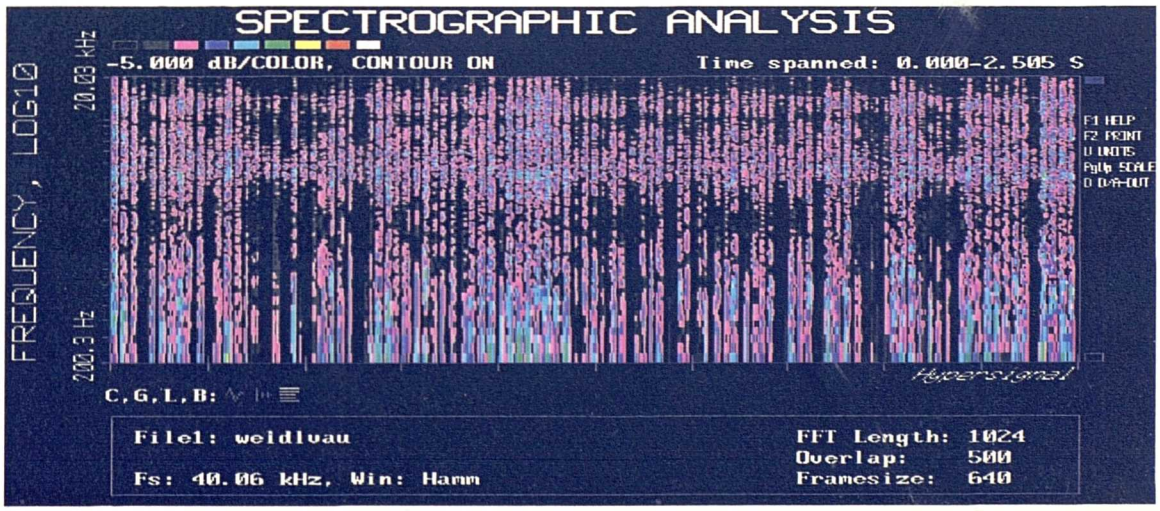
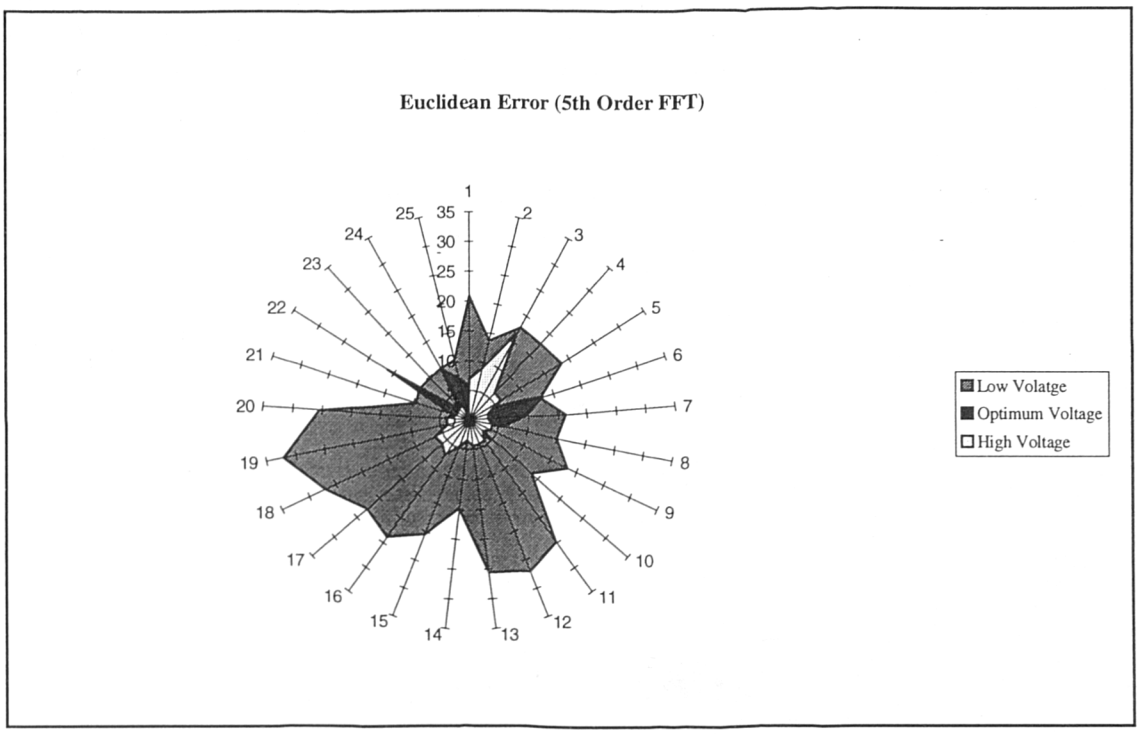
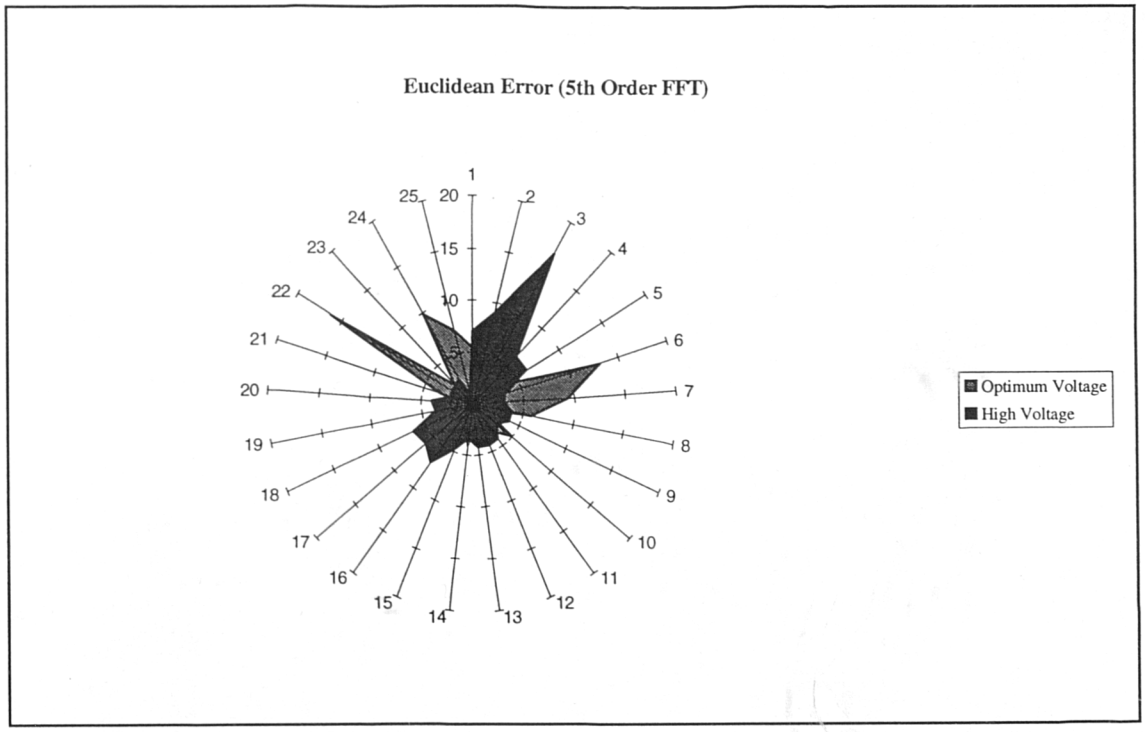


FIGURE 3.47
Spectral Contour Map - Low Weld Voltage

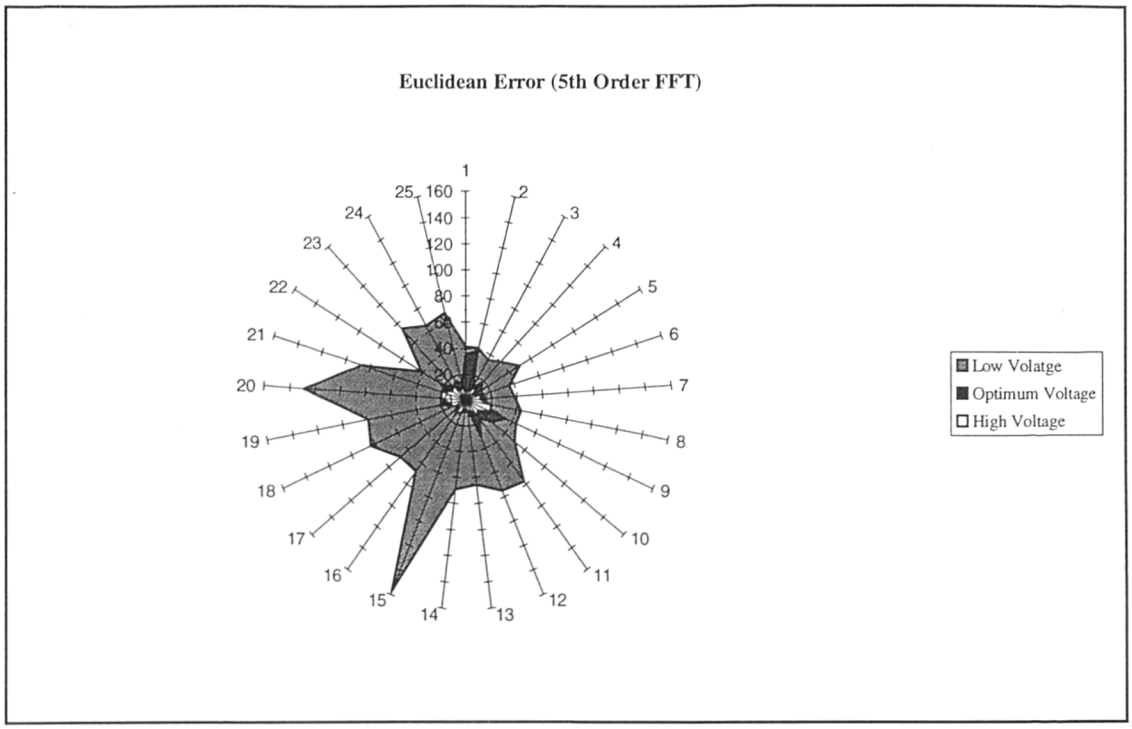


a) All data sets (Optimum Weld Level Training Data)

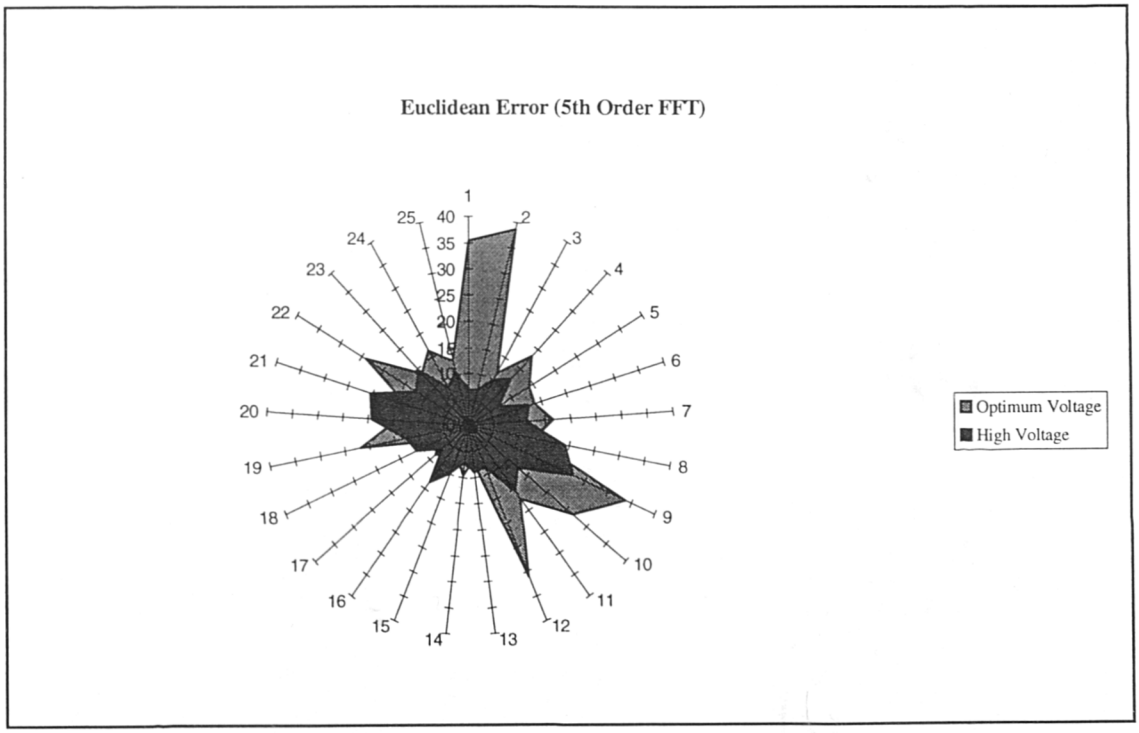


b) Optimum & High voltage Data (Optimum Weld Level Training Data)

FIGURE 3.48 a-b
Euclidean Error Graphs (Table 3.5)

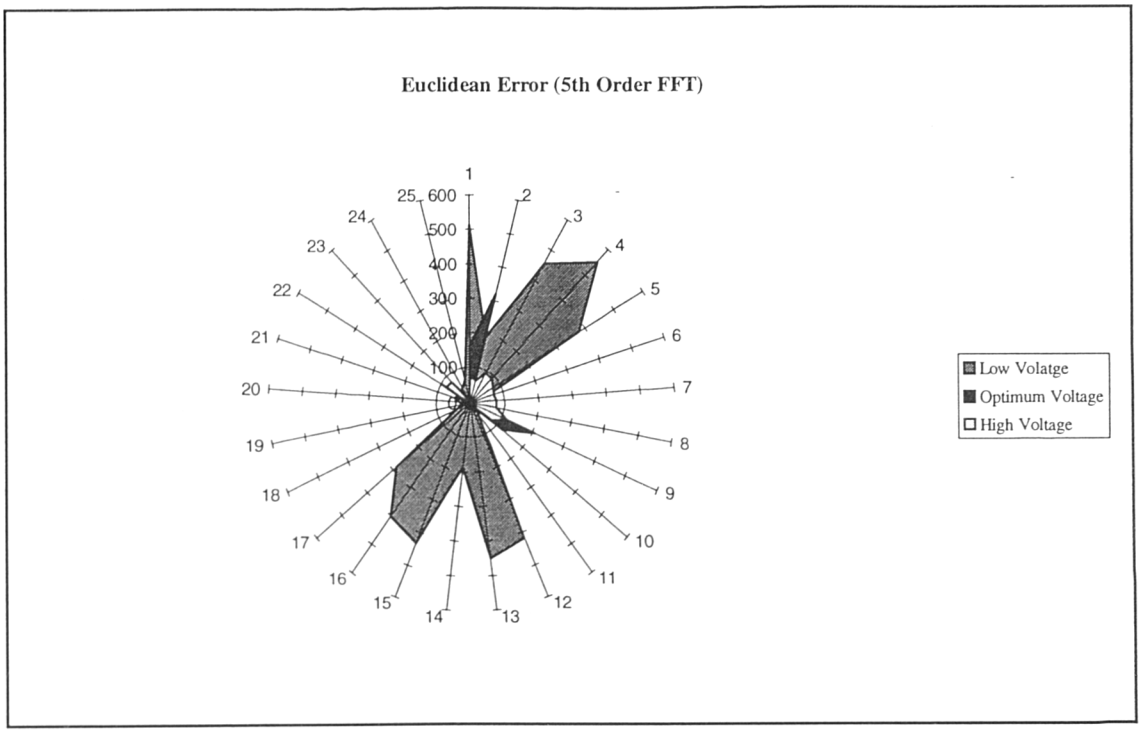


c) All data sets (High Voltage Level Training Data)

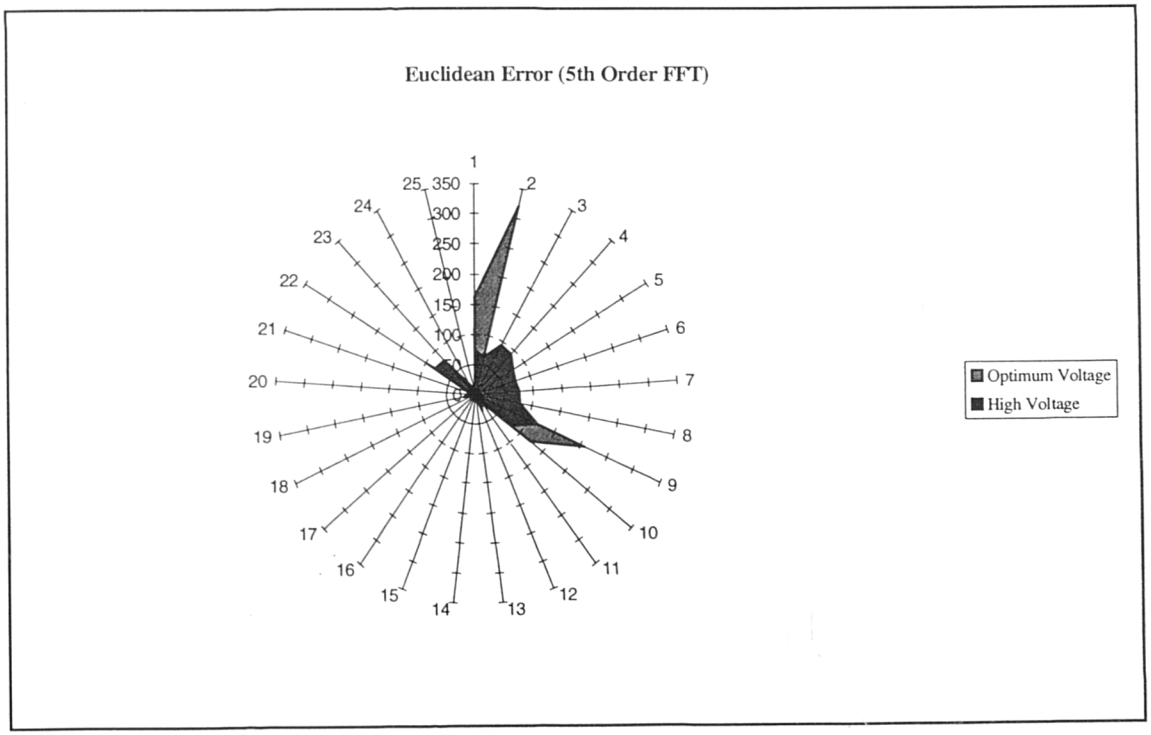


d) Optimum & High voltage Data (High Voltage Level Training Data)

FIGURE 3.48 c-d
Euclidean Error Graphs (Table 3.5)



e) All data sets (Low Voltage Level Training Data)



f) Optimum & High voltage Data (Low Weld Level Training Data)

FIGURE 3.48 e-f
Euclidean Error Graphs (Table 3.5)

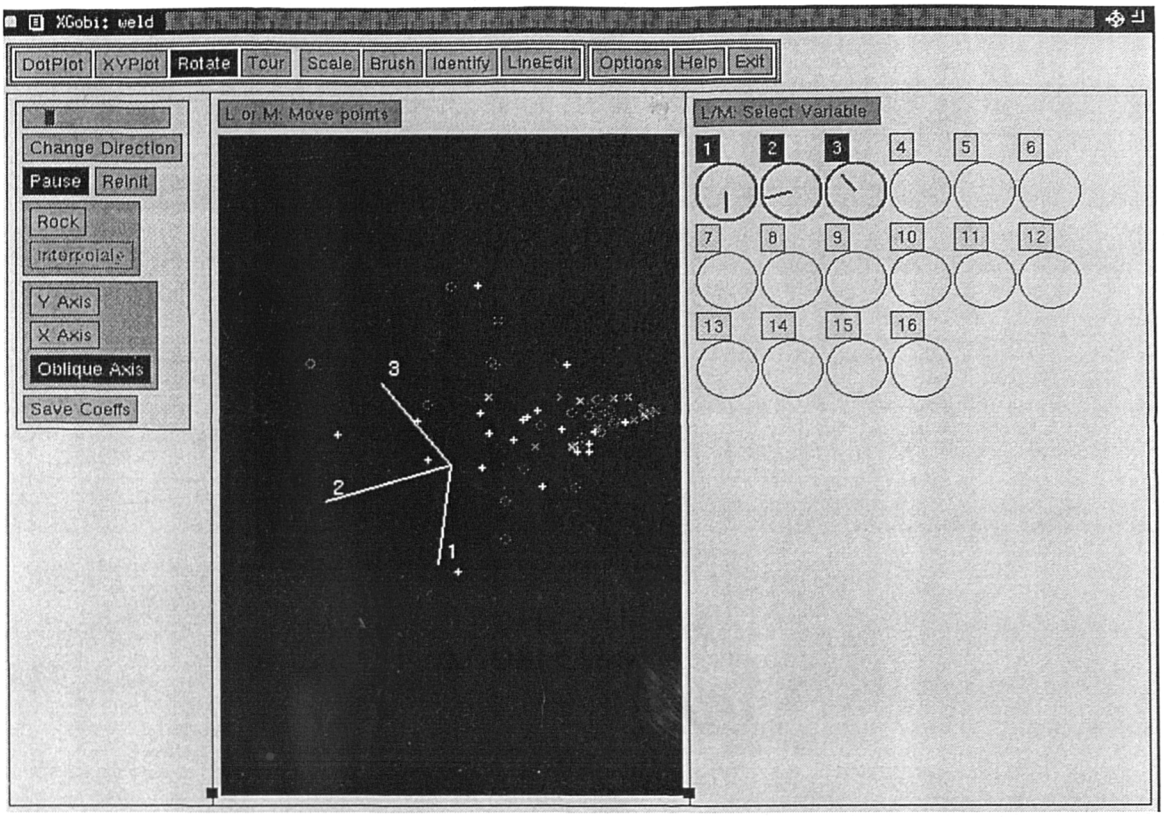


FIGURE 3.49
XGOBI Cluster Analysis

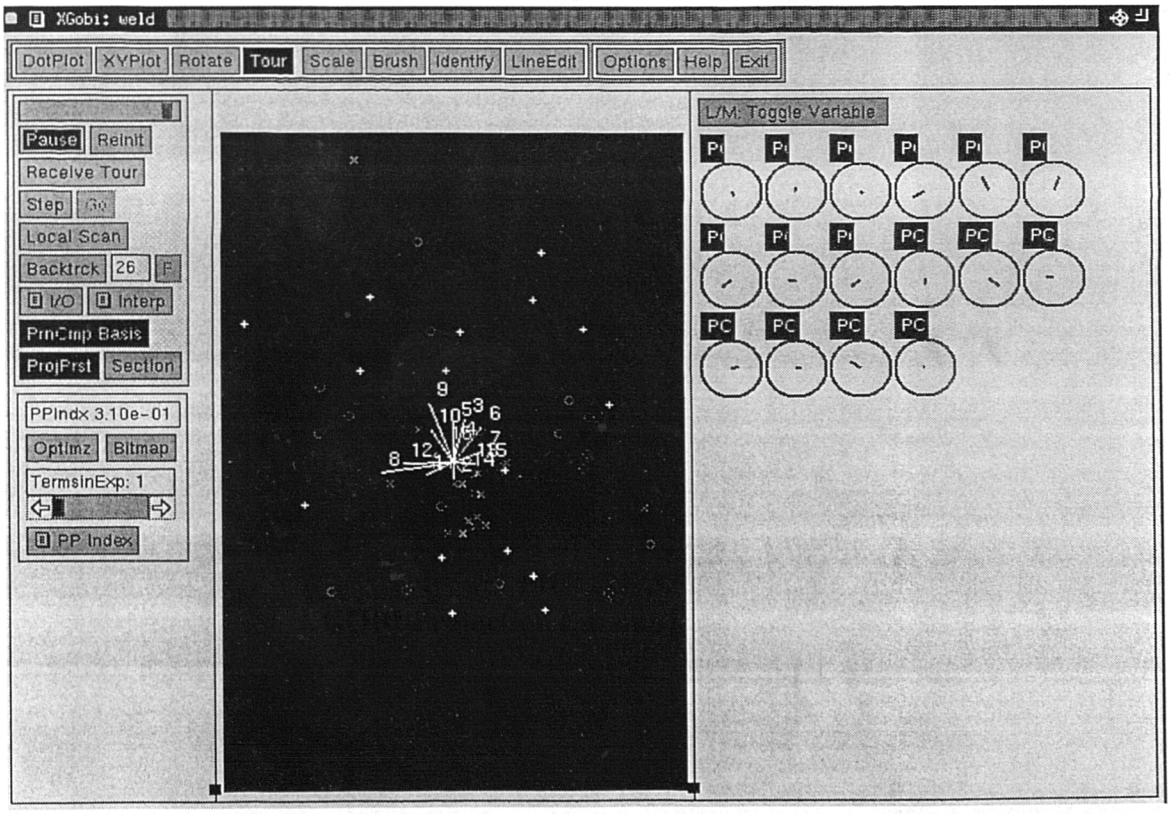


FIGURE 3.50
XGOBI Projection Pursuit Data Manipulation

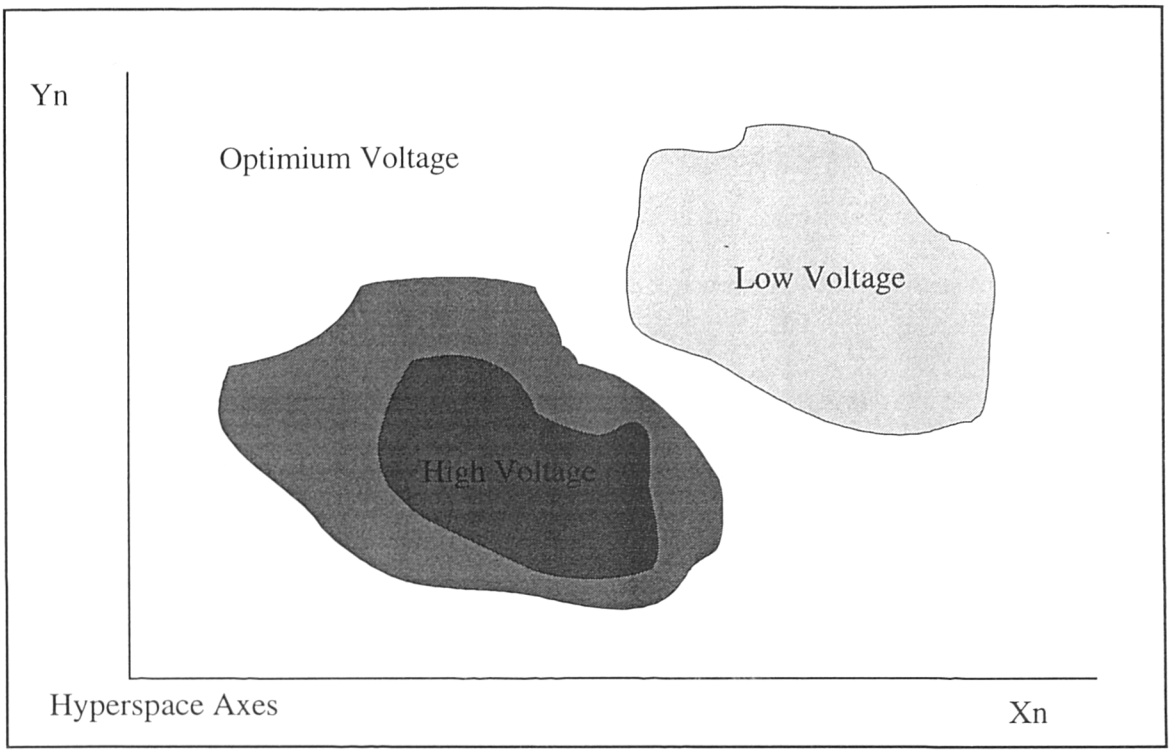


FIGURE 3.51
Voltage Data Set in Hyperspace - High Volts as a Subset of the Optimum

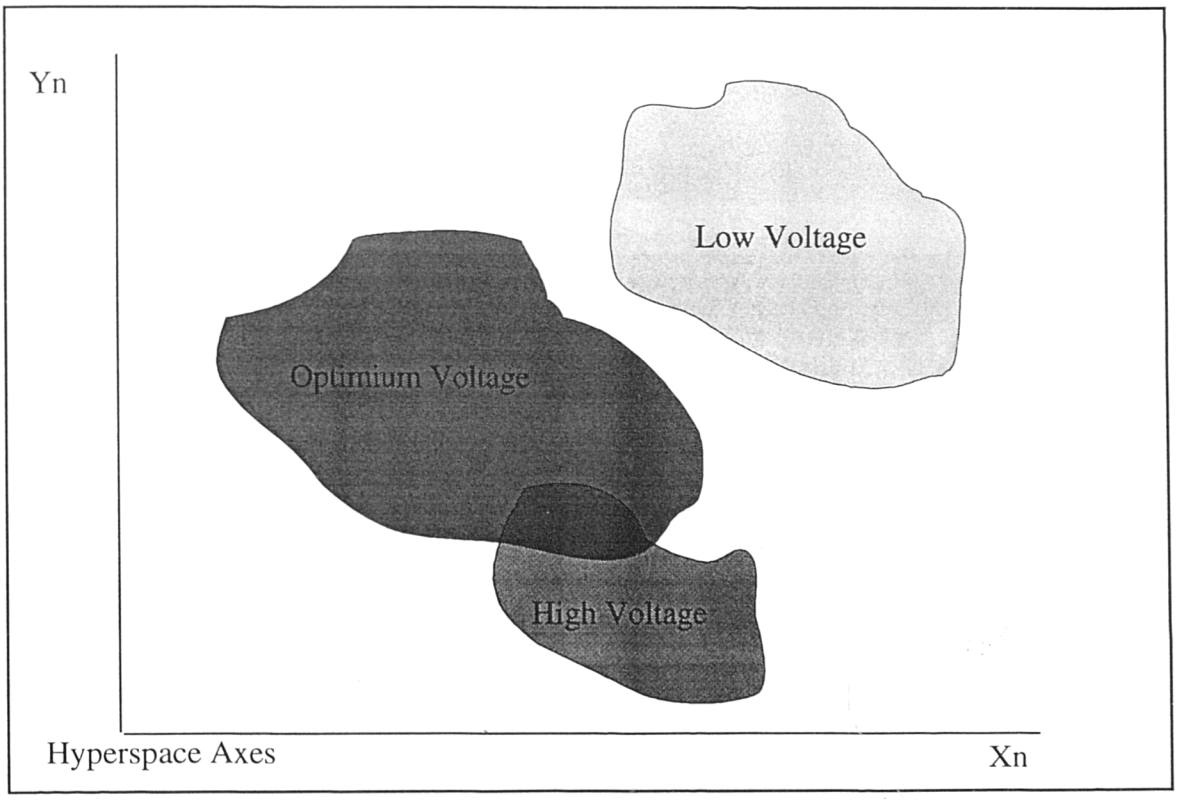


FIGURE 3.52
Separation of Data in Hyperspace - High Volts Yields a Minor Union

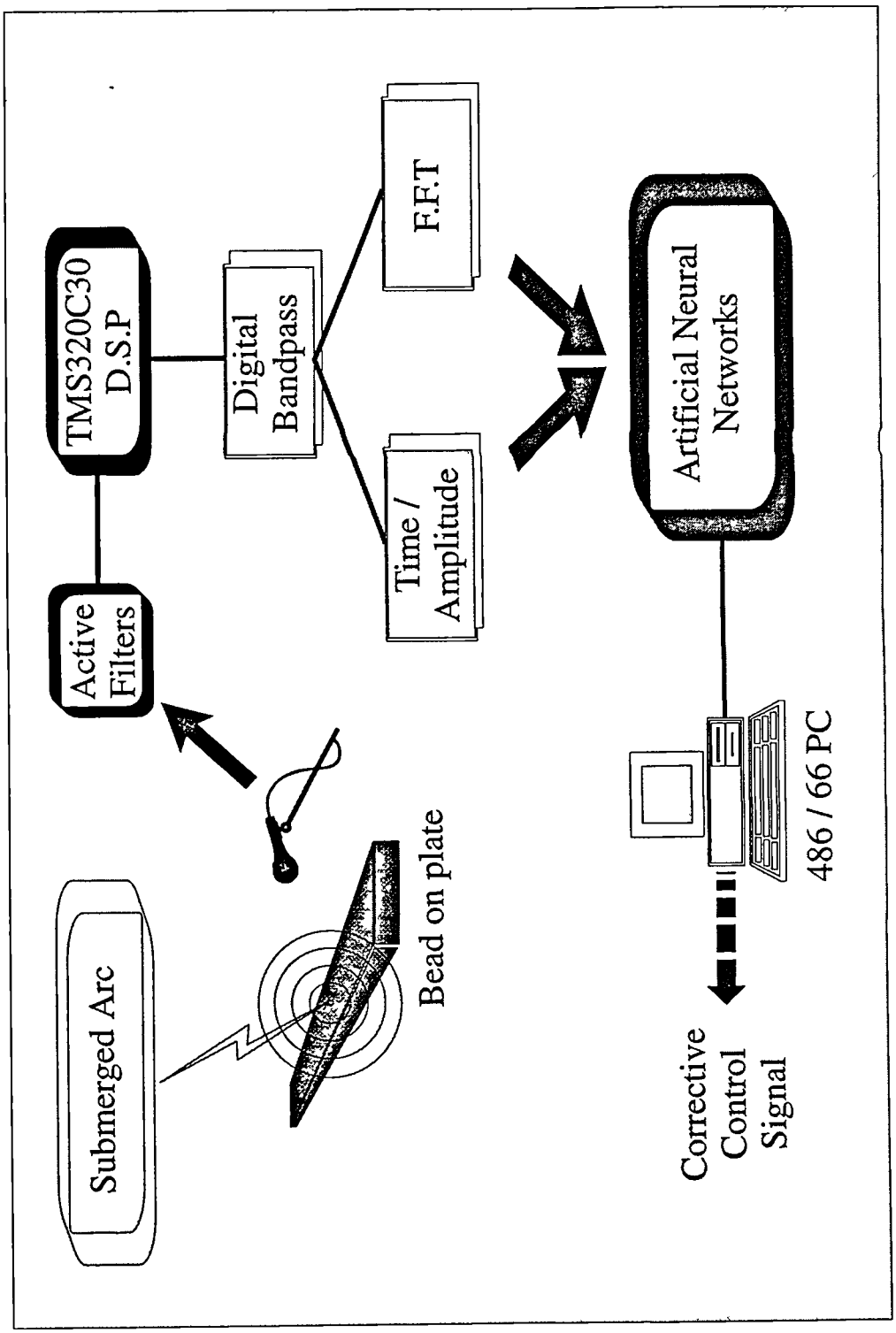
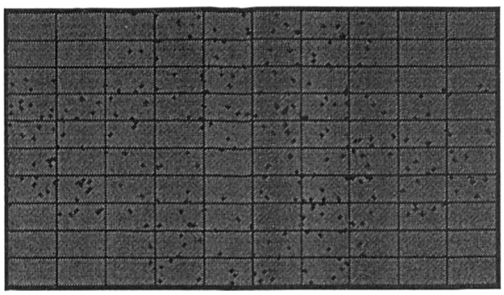
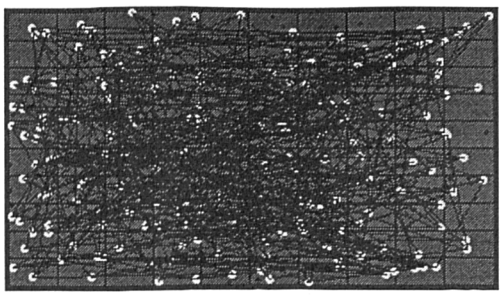


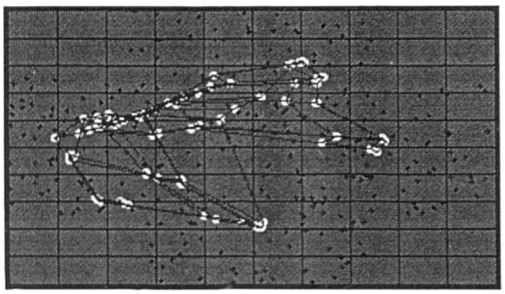
FIGURE 3.53
Experimental System Set-up



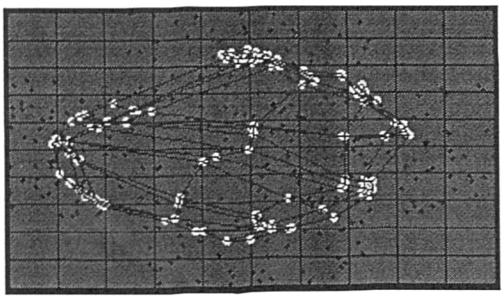
a) Training Data Set - Cross



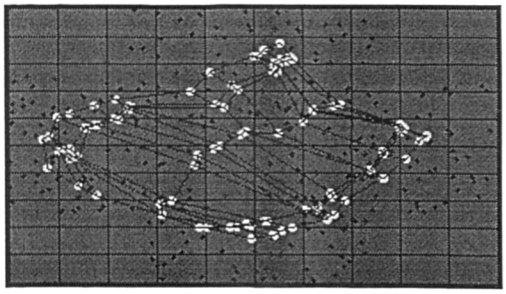
b) Random Weight Initiation



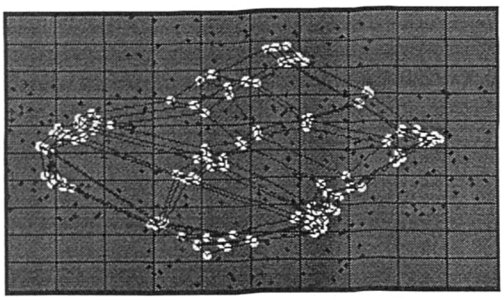
c) 100 Iterations



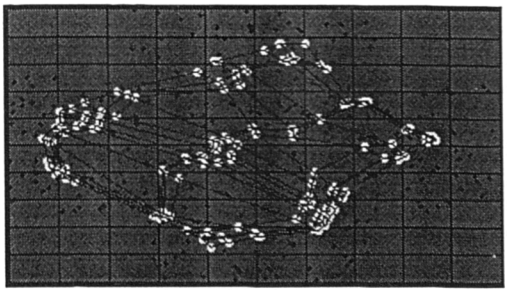
d) 200 Iterations



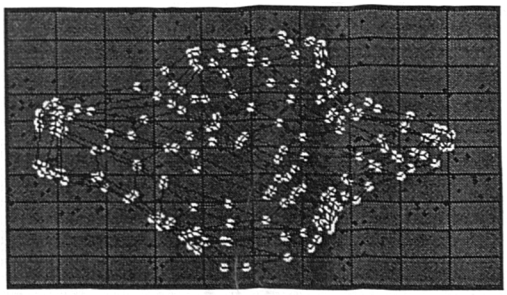
d) 300 Iterations



e) 400 Iterations

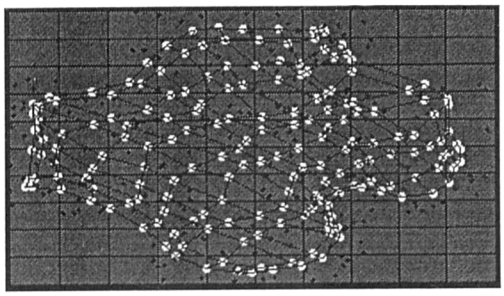


f) 500 Iterations

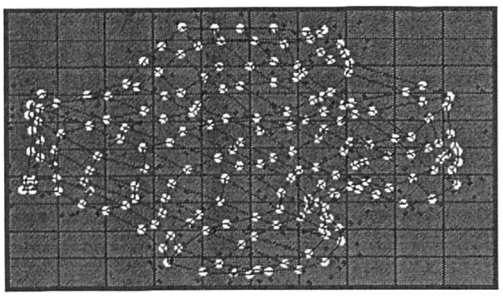


g) 600 Iterations

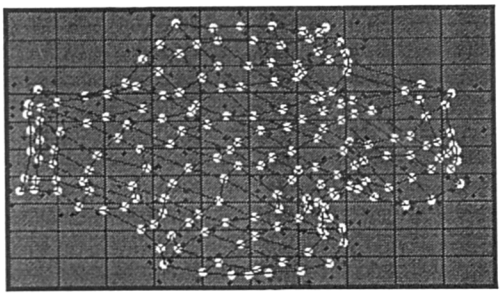
FIGURE 3.54
Learning Progression of Two Dimensional Cross Data (Pt1)



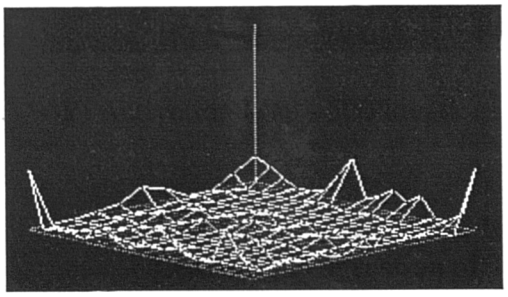
h) 700 Iterations



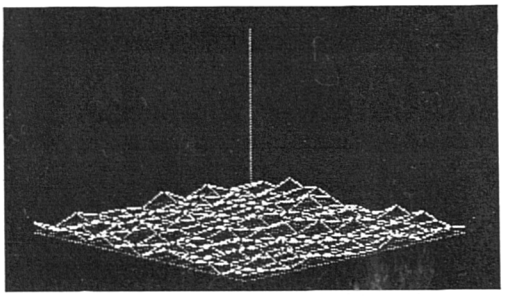
i) 800 Iterations



j) 900 Iterations

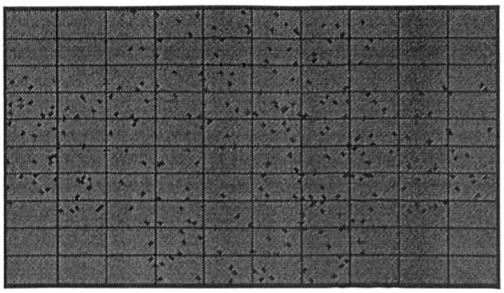


k) Activation Map - 500 Iterations

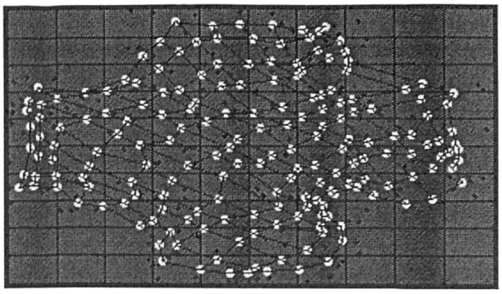


l) Activation Map - 900 Iterations

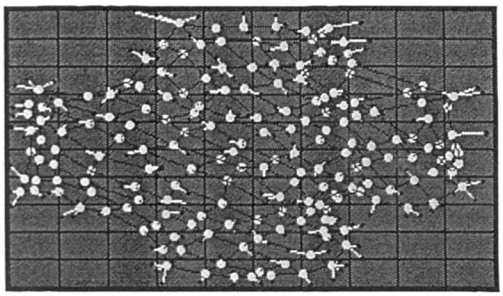
FIGURE 3.54
Learning Progression of Two Dimensional Cross Data (Pt2)



1) Test Pattern (Cross Configuration)



2) Trained Weights



3) Test Run & Error Calculation

FIGURE 3.55 a
Output layer on Test Set of Similar Classification

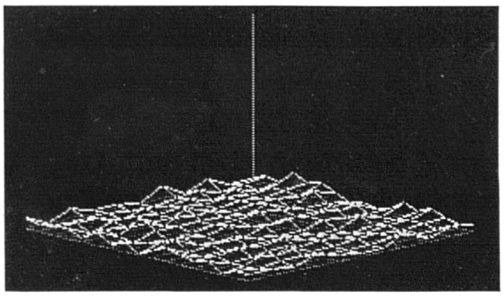
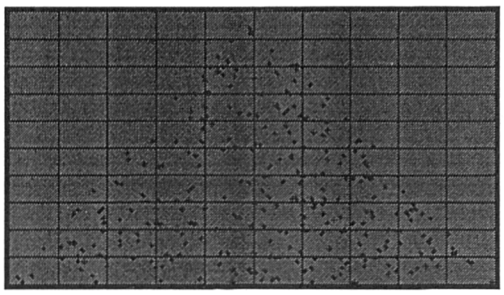
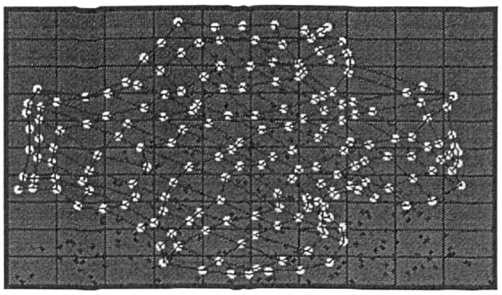


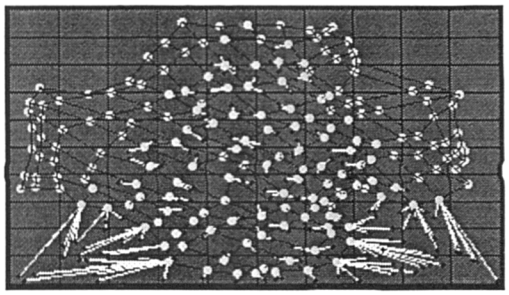
FIGURE 3.55 b
Output layer Activation Map
on Test Set of Similar Classification



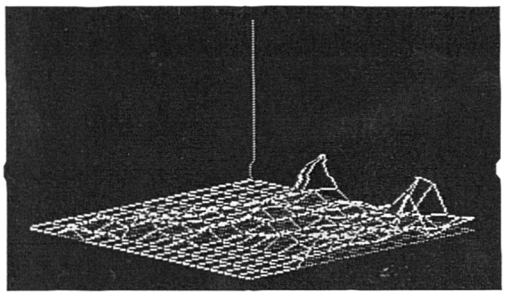
a) Test Data - Triangular Distribution



b) Trained Network

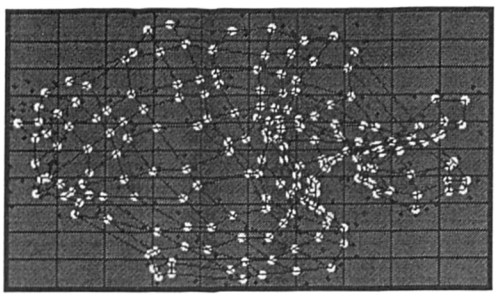


c) Test on Trained Network

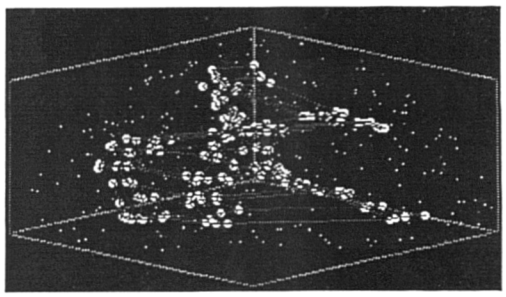


d) Resulting Activation Map

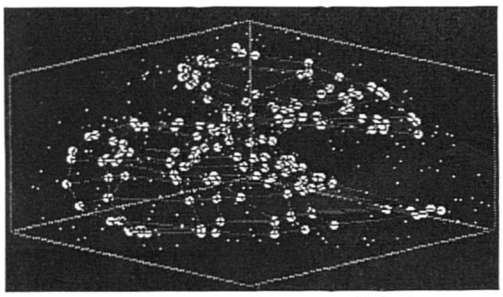
FIGURE 3.56
Output Layer Activation - Test Data Distribution Change



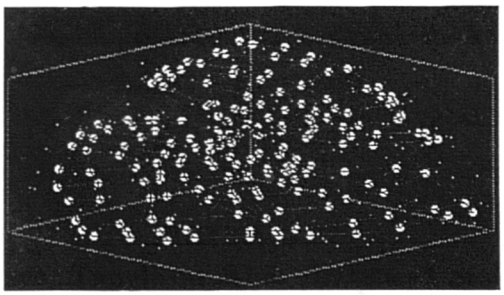
a) Two Dimensional Tangle



b) 3D Training - 500 Iterations

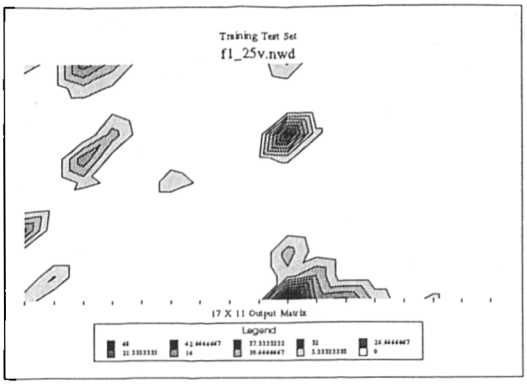


c) Tangle Development

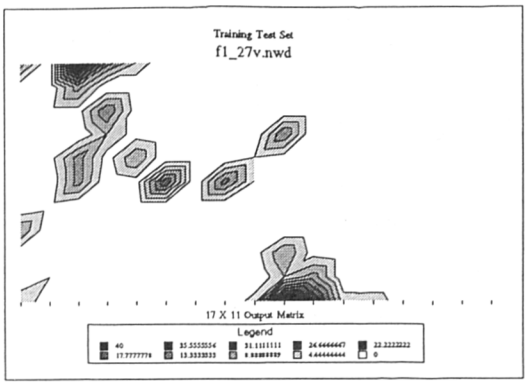


d) Full Tangle -(Local Minima)

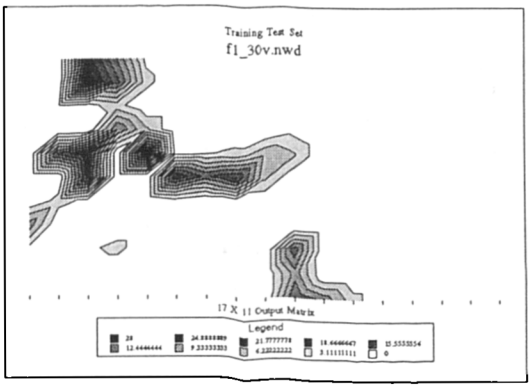
FIGURE 3.57
Output Layer Tangle on 2D & 3DData - Local Minima



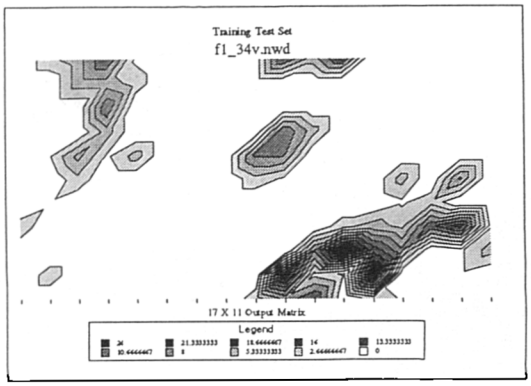
a)Activation Map- 25 Volts



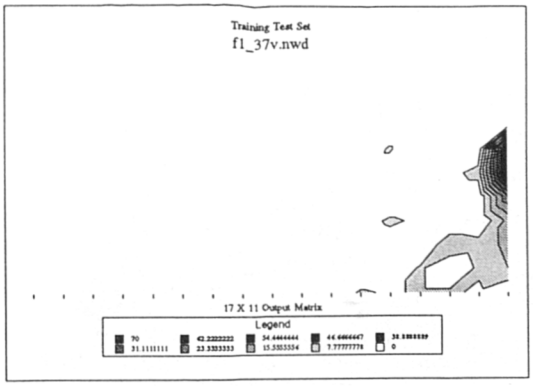
b)Activation Map- 27 Volts



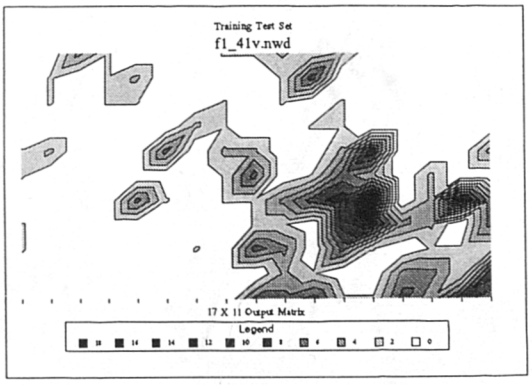
c)Activation Map- 30 Volts



d)Activation Map- 34 Volts

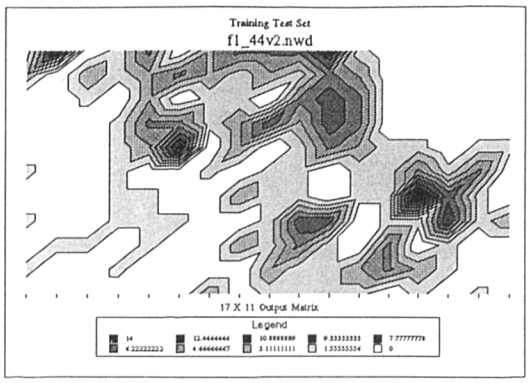


e)Activation Map- 37 Volts

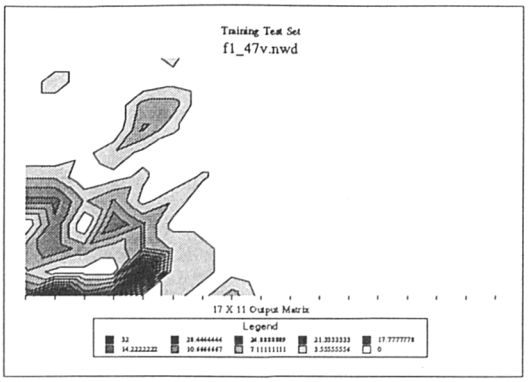


f)Activation Map- 41 Volts

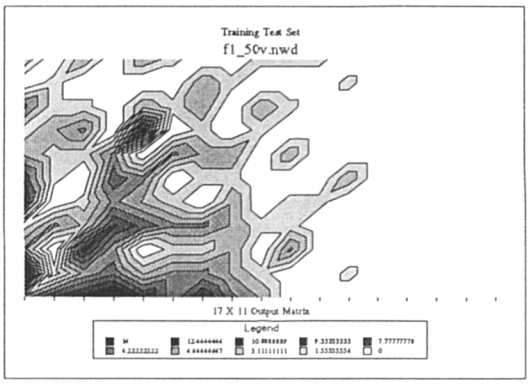
FIGURE 3.58 - Output Layer Acivation Map (pt1)



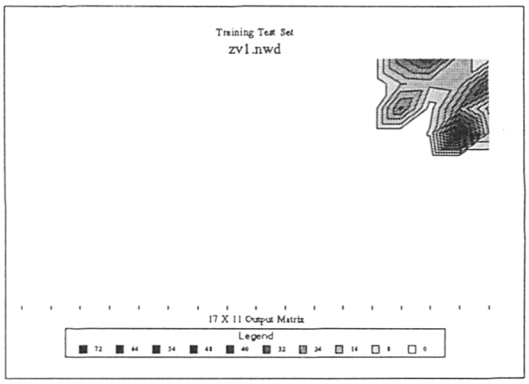
g)Activation Map- 44 Volts



h)Activation Map- 47 Volts



i)Activation Map- 50 Volts



j)Activation Map- Zero Volts

FIGURE 3.58 - Output Layer Acivation Map (pt2)

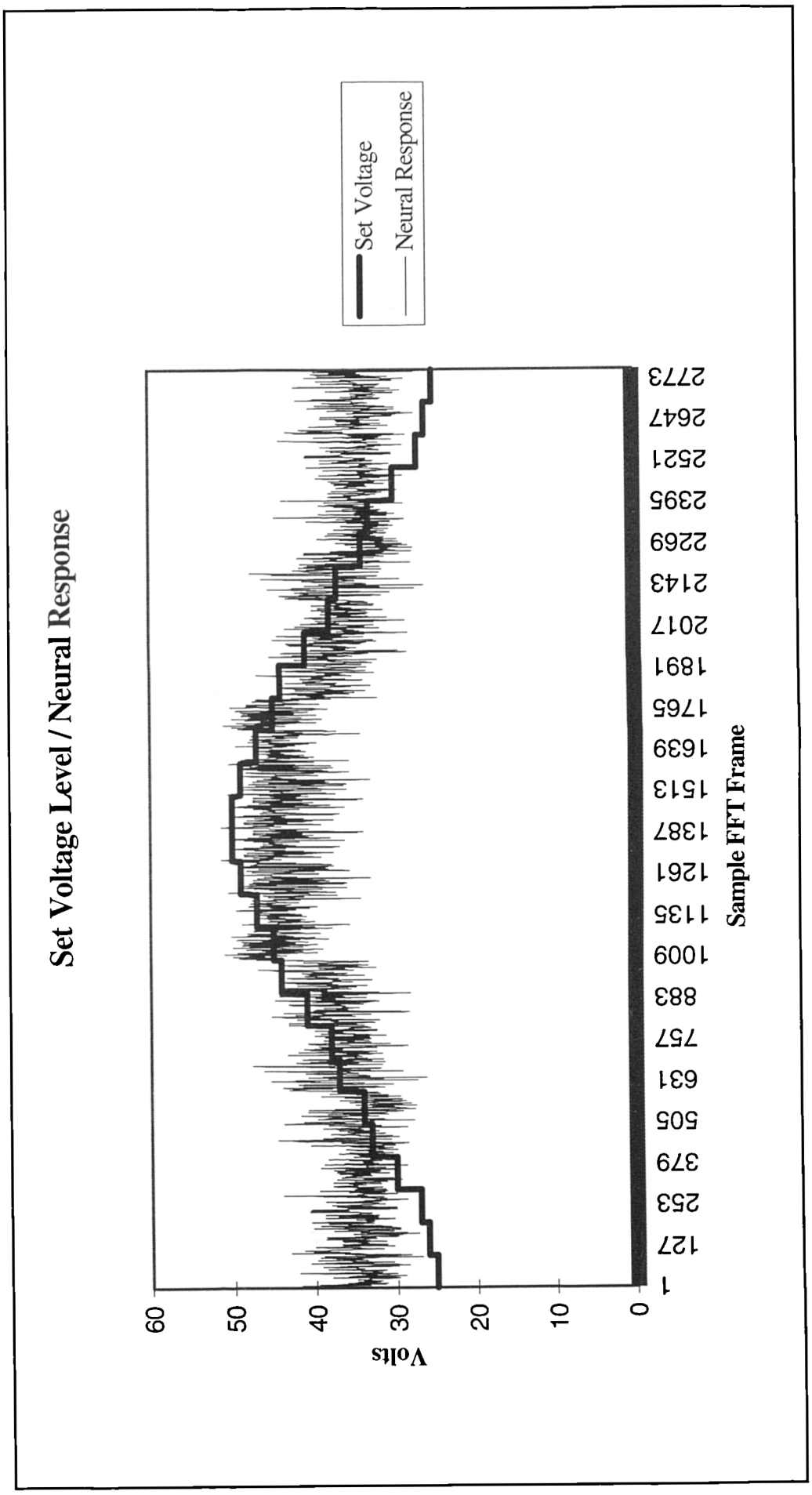


FIGURE 3.59
Monitored & ANN Predicted Voltage Trace

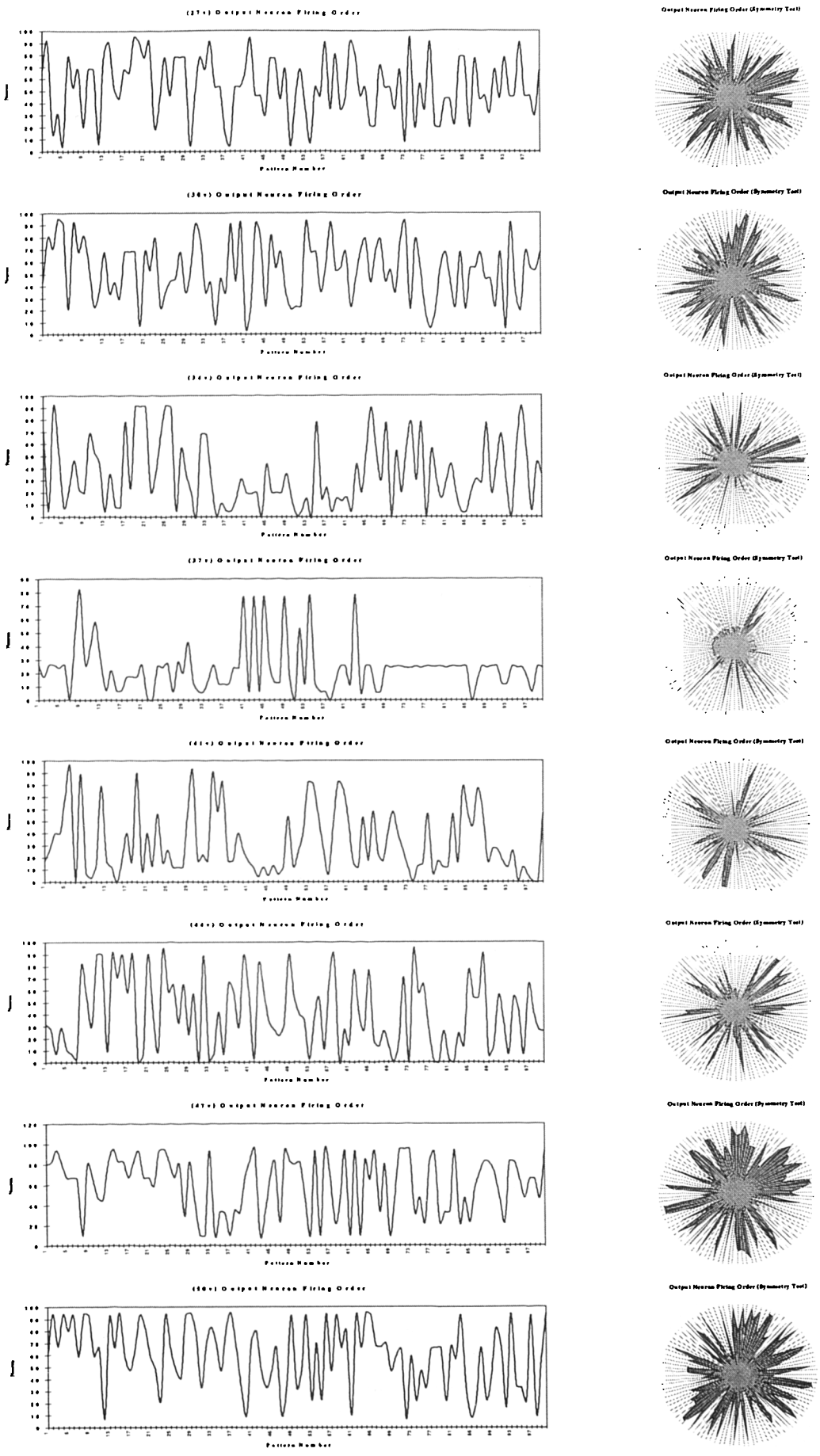


FIGURE 3.60
Output Node Firing Order for Series of Voltage Change Welds

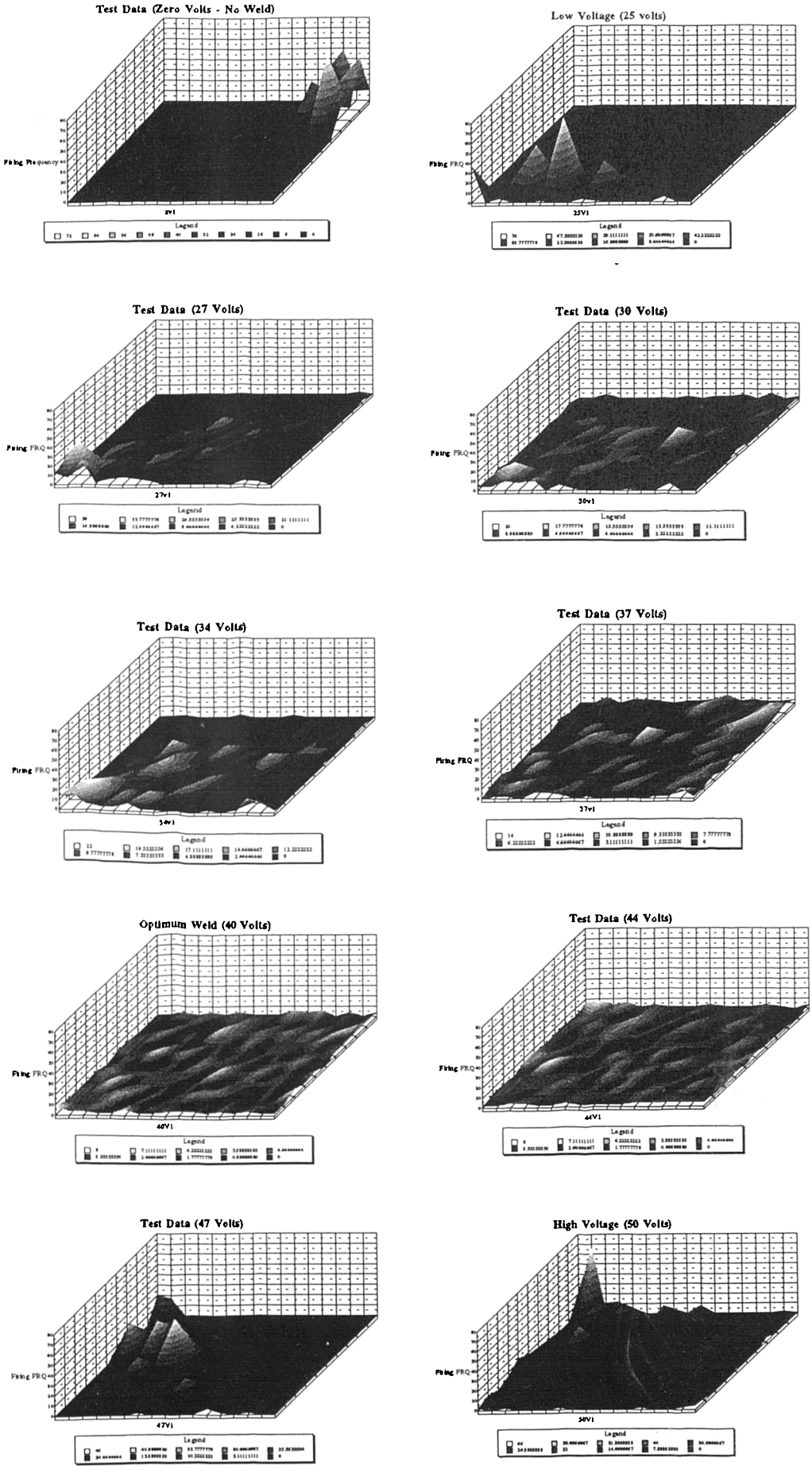


FIGURE 3.61

Activation Map Series - Voltage Level Identification

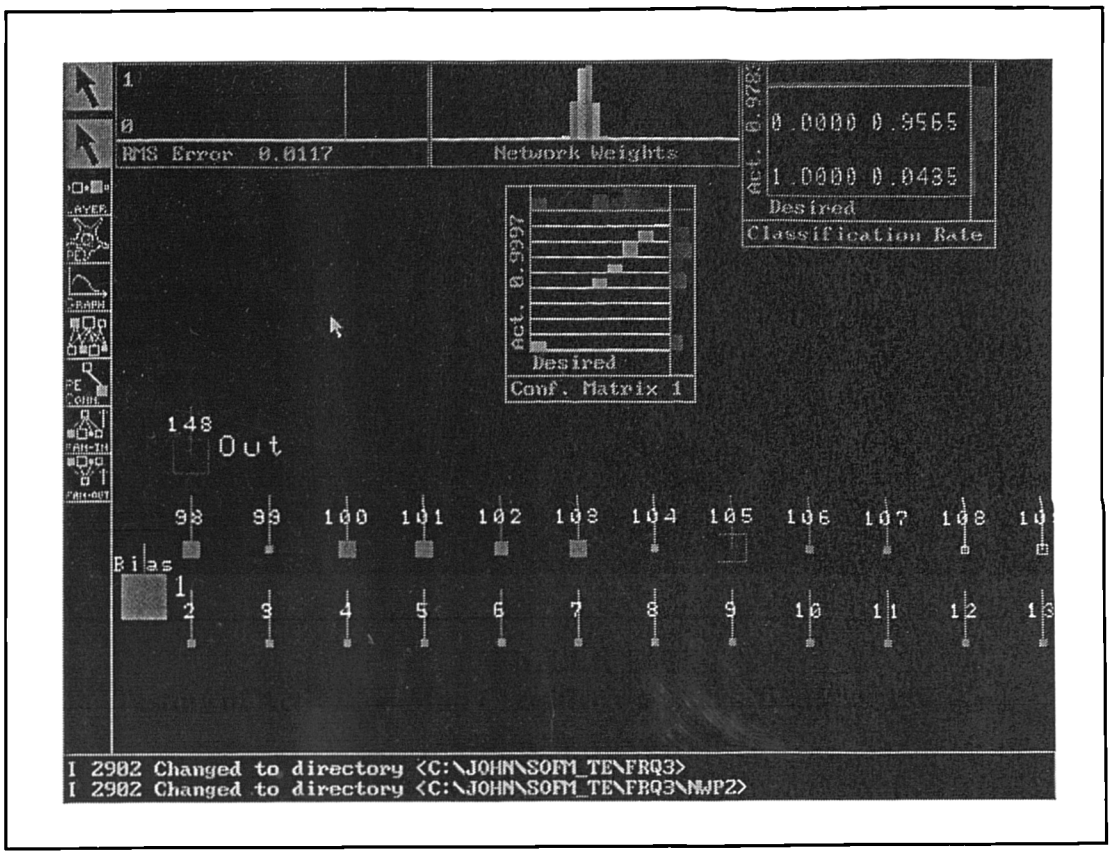


FIGURE 3.62
Testing of Activation Map Classifier via NeuralWorks Professional II Feedforward Network

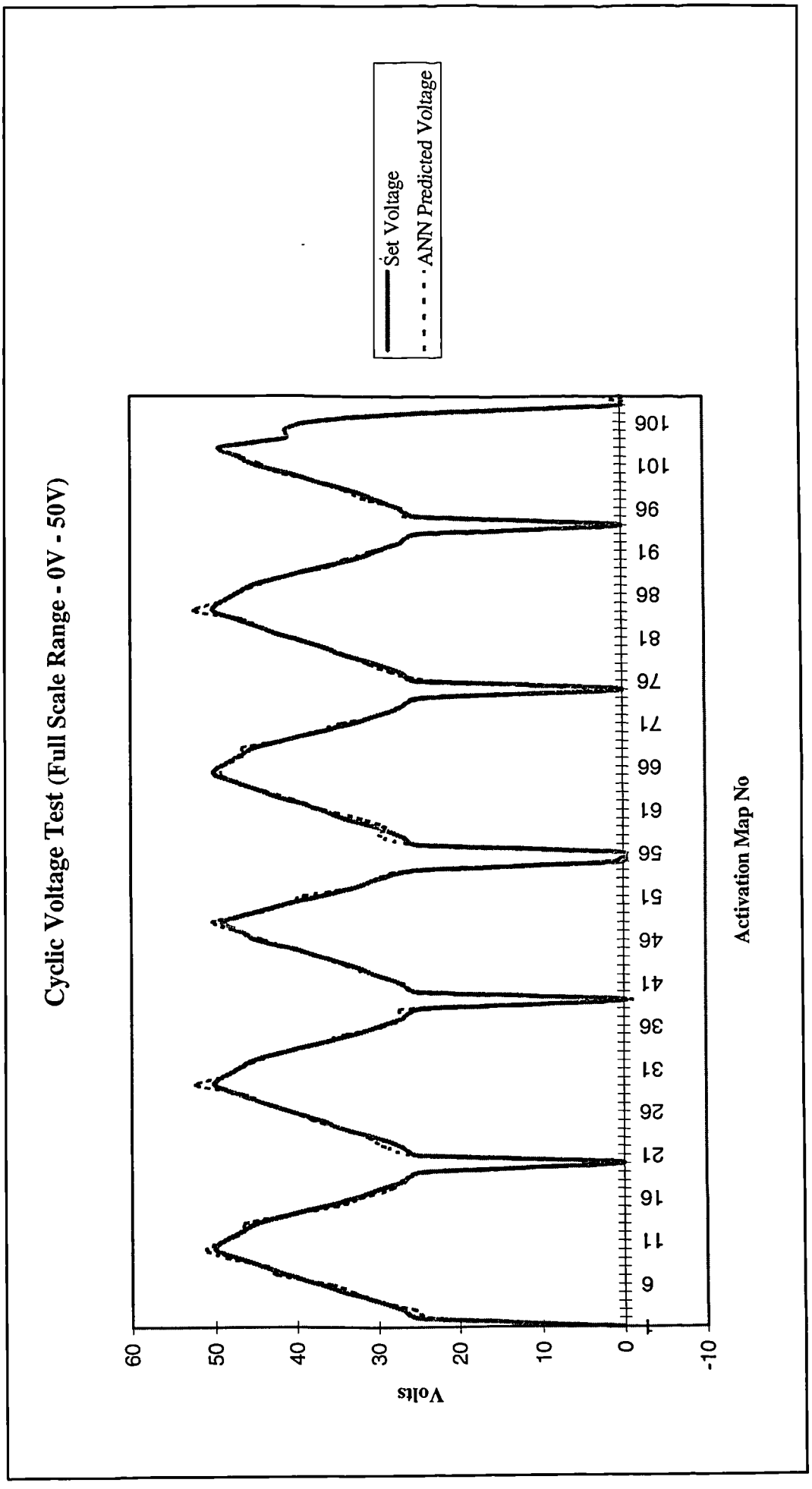


FIGURE 3.63
Pre-Set & ANN Predicted Voltage Levels On Cyclic Test

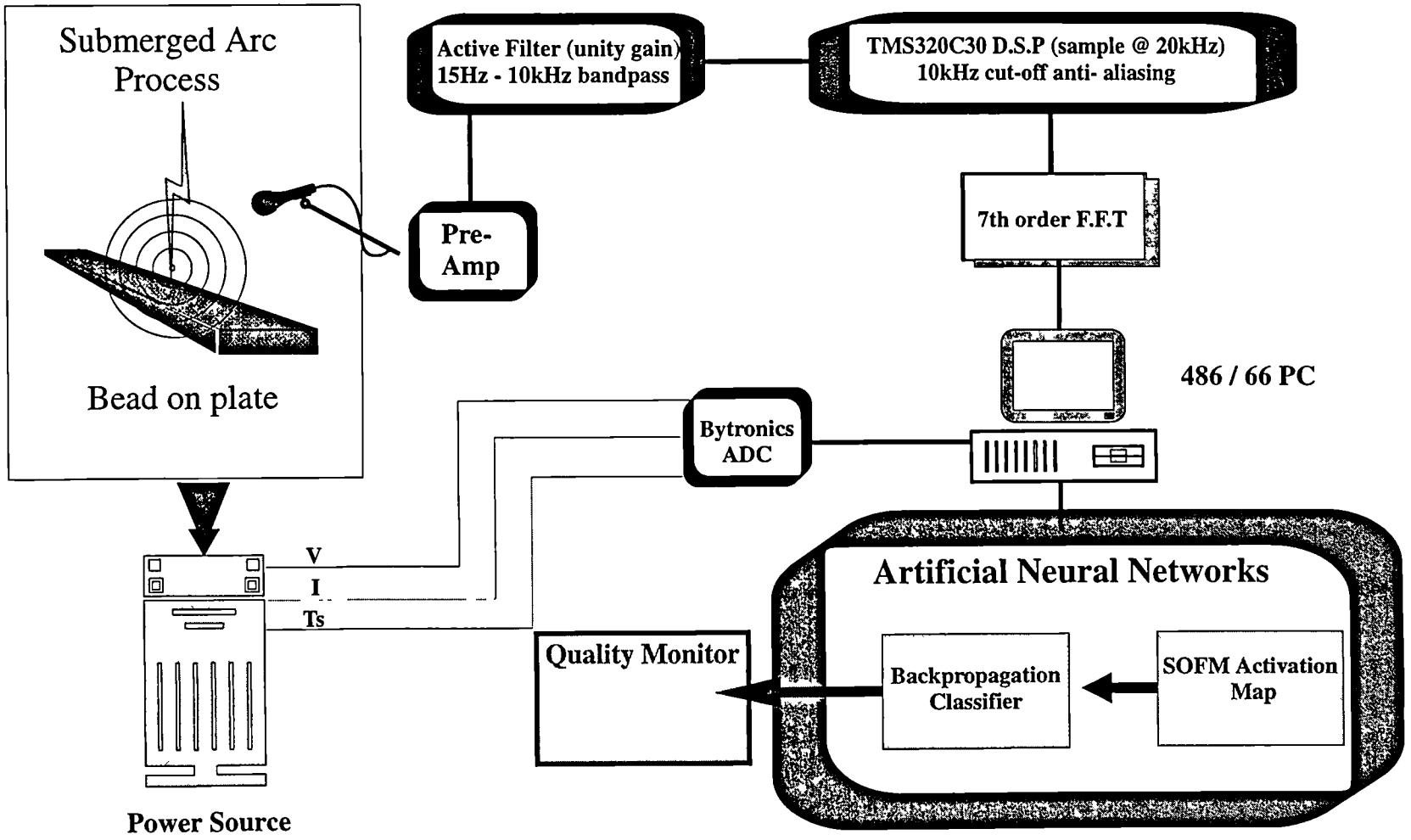


FIGURE 3.64
Full & Final Experimental System Set-up

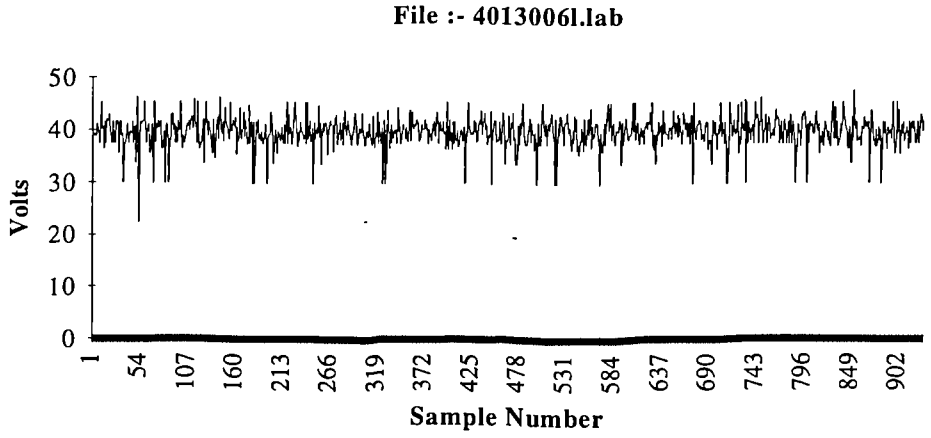


FIGURE 4.1
Typical Voltage Trace (40V - Nominal)

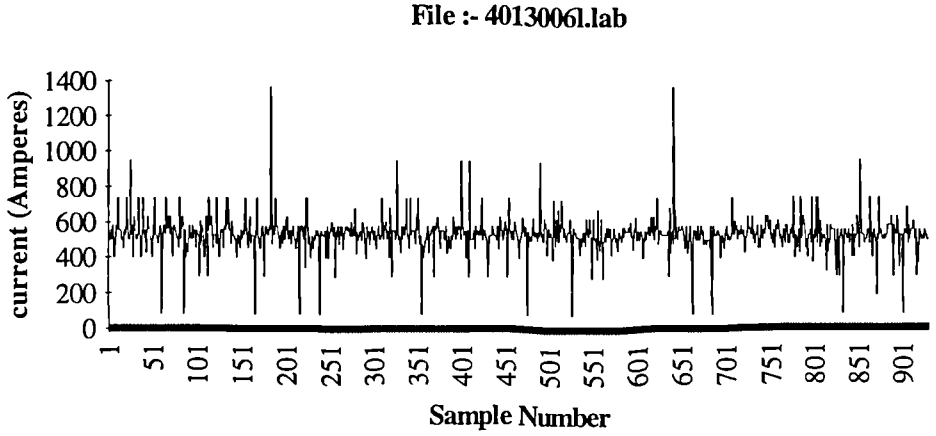


FIGURE 4.2
Typical Current Trace (600A - Nominal)

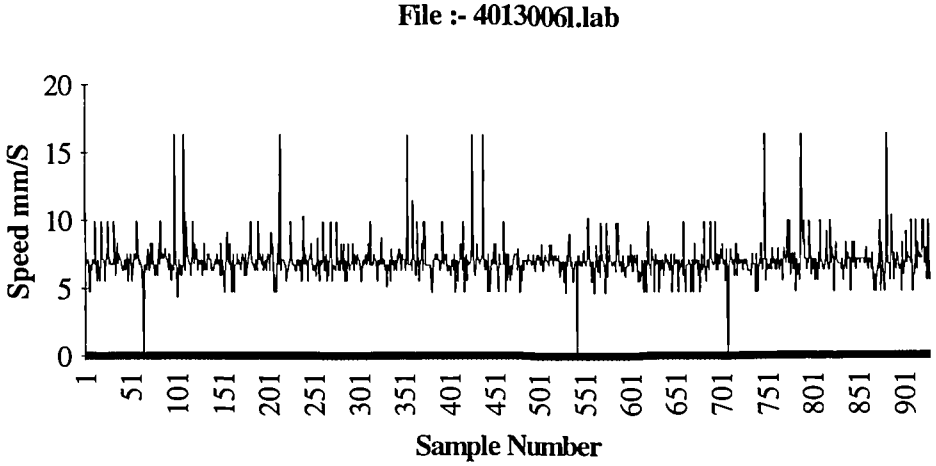


FIGURE 4.3
Typical Travel Speed Trace (6.5 mm/s - Nominal)

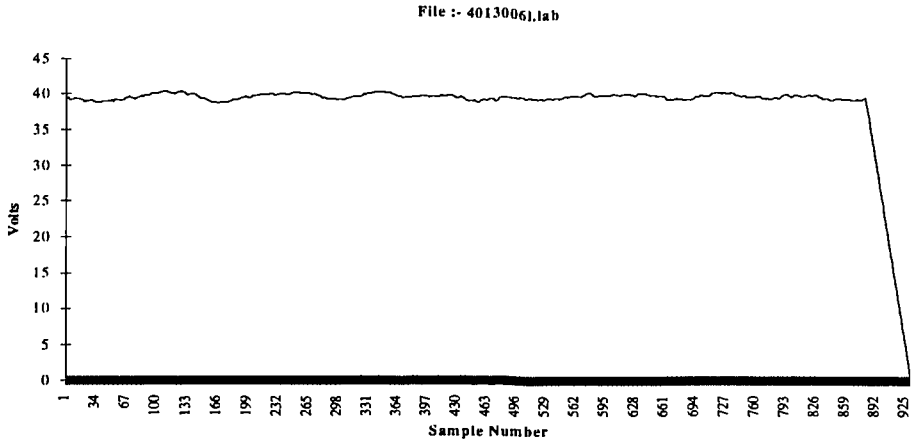


FIGURE 4.4
Typical Averaged Voltage Trace (Sample width = 50, 40V - Nominal)

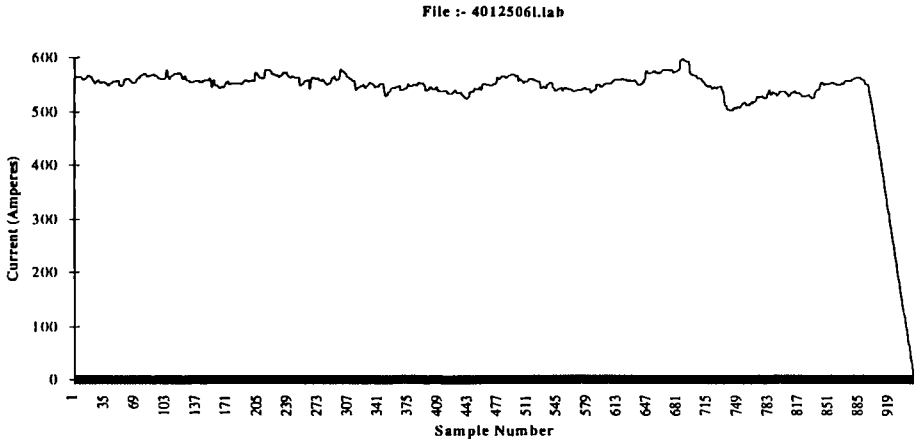


FIGURE 4.5
Typical Averaged Current Trace (Sample width = 50, 550A - Nominal)

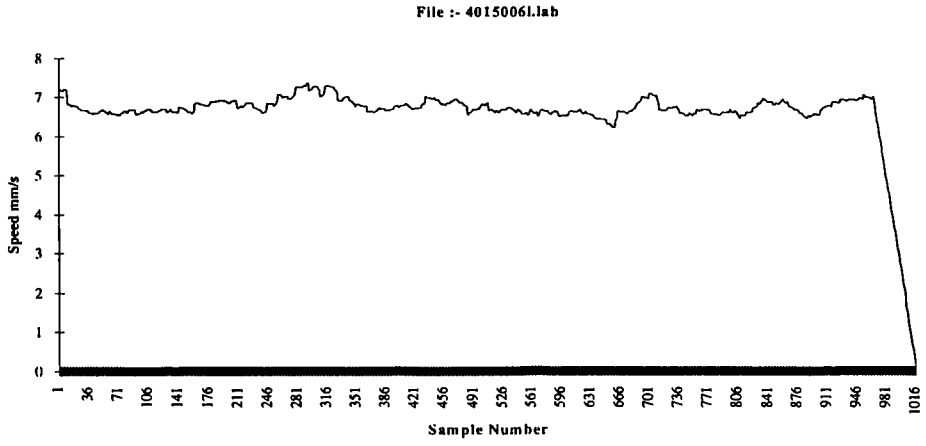
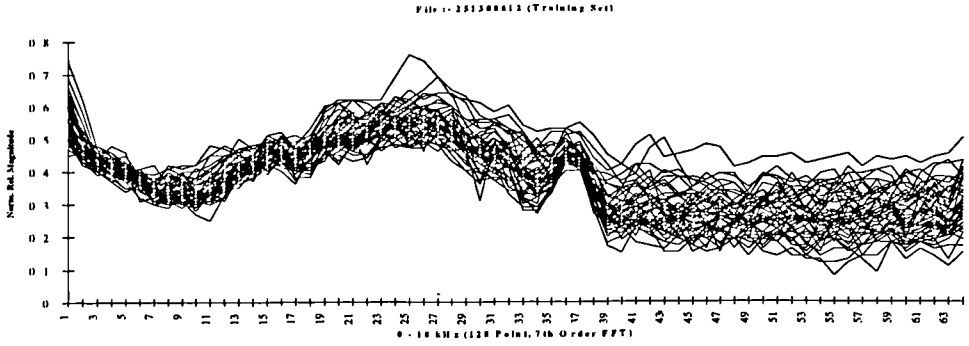
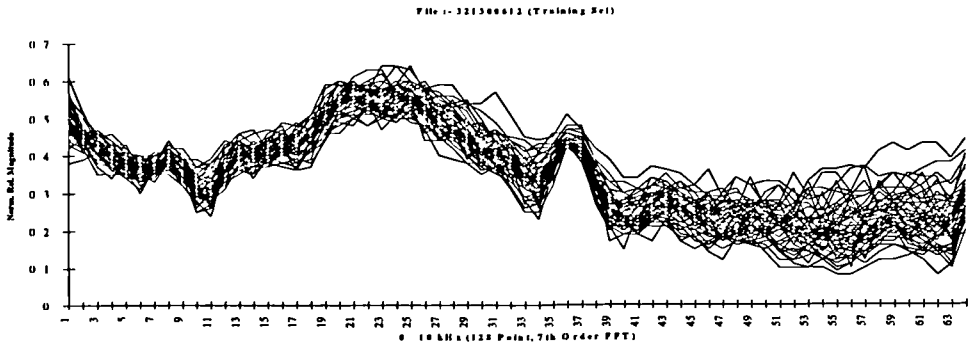


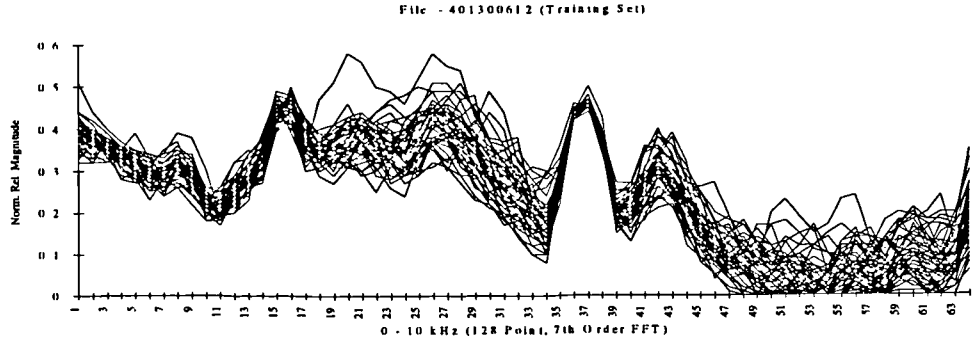
FIGURE 4.6
Typical Averaged Travel Speed Trace (Sample width = 50, 6.5mm/s - Nominal)



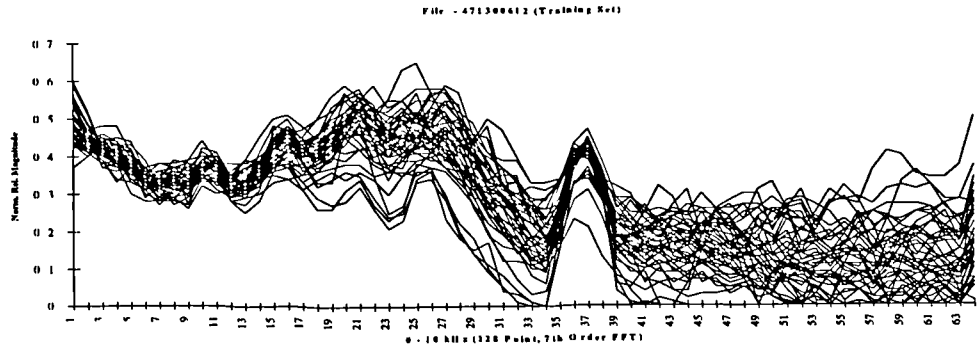
a) 25 Volts



b) 32 Volts

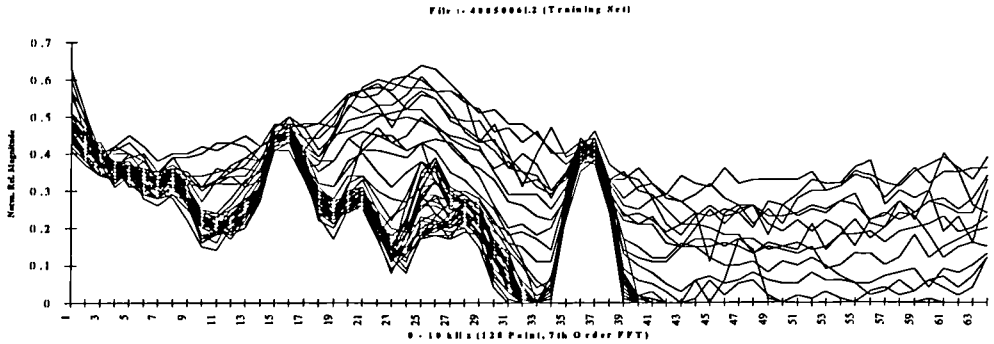


c) 40 Volts

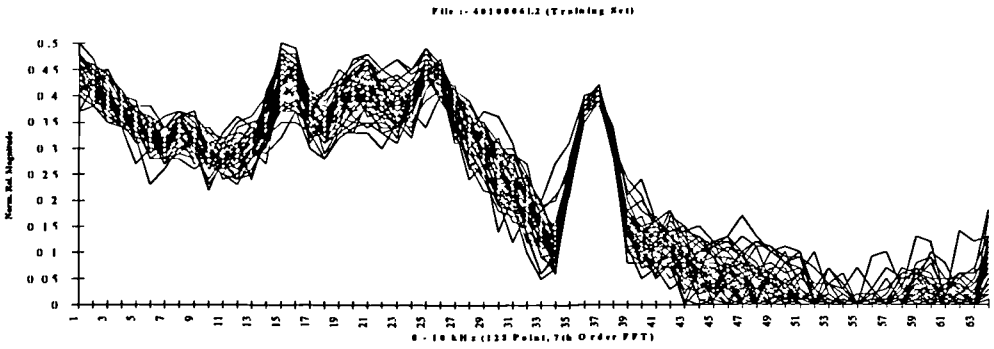


d) 47 Volts

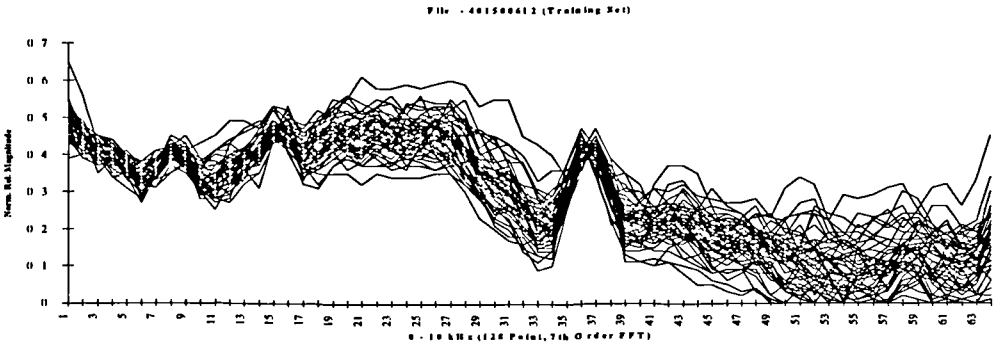
FIGURE 4.7a-d
Typical FFT Traces for Voltage Changes



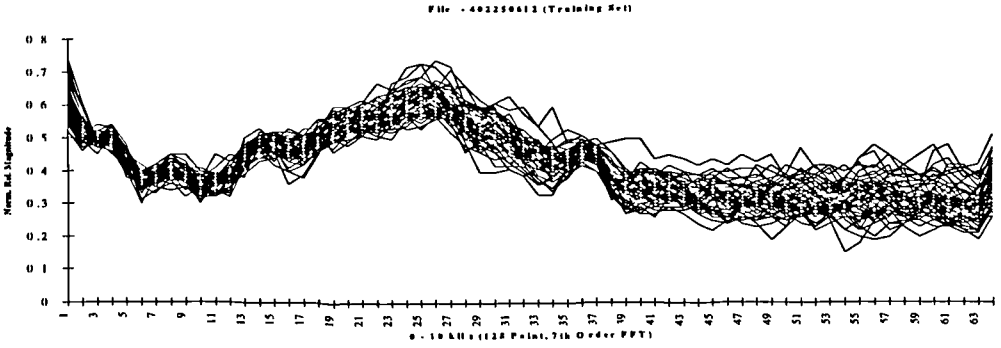
a) 175 Amps



b) 350 Amps

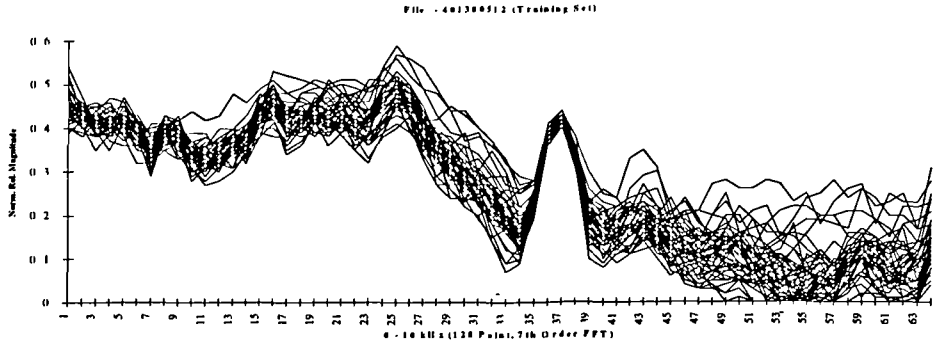


c) 625 Amps

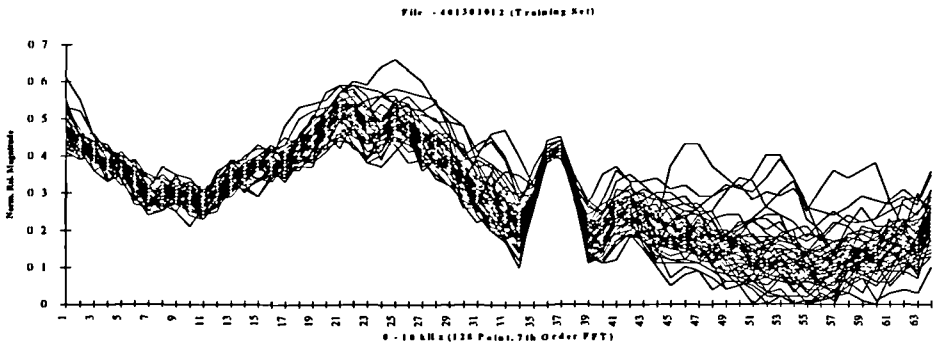


d) 930 Amps

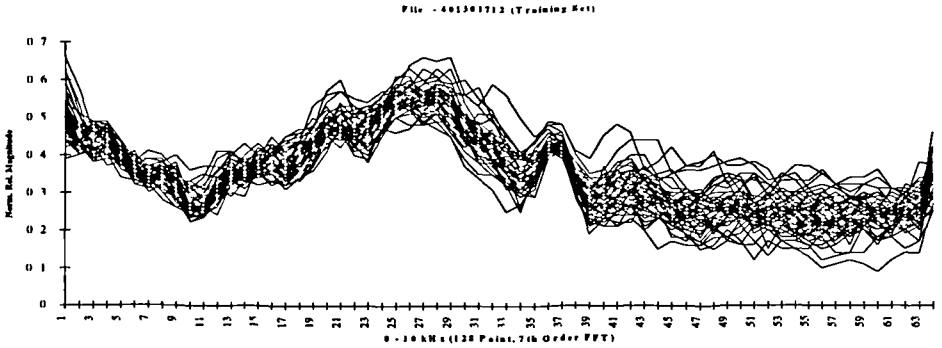
FIGURE 4.8a-d
Typical FFT Traces for Current Changes



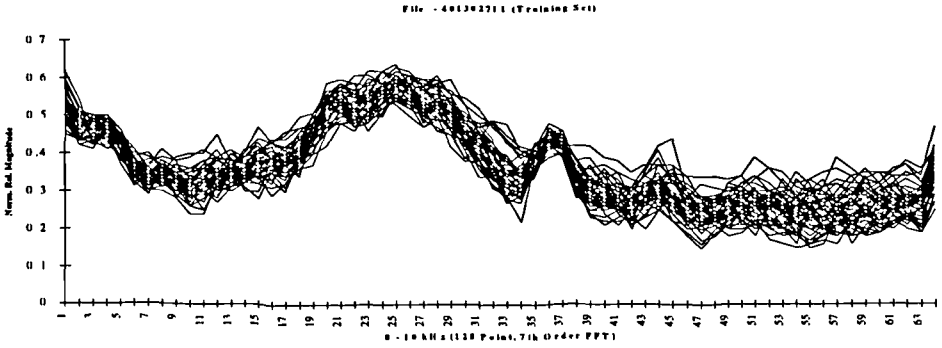
a) 5 mm/s



b) 10 mm/s

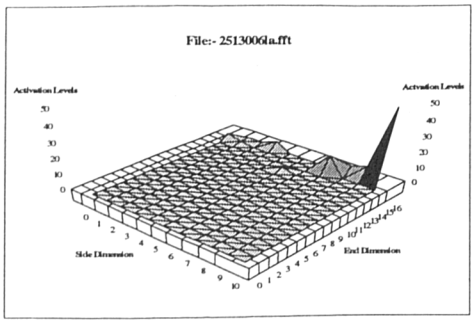


c) 17 mm/s

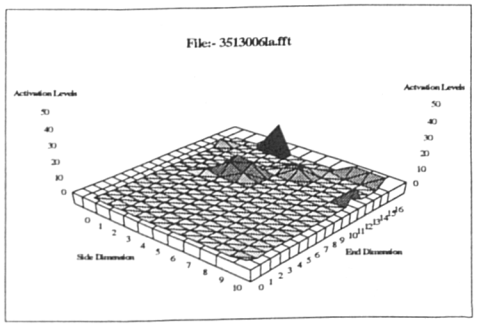


a) 27 mm/s

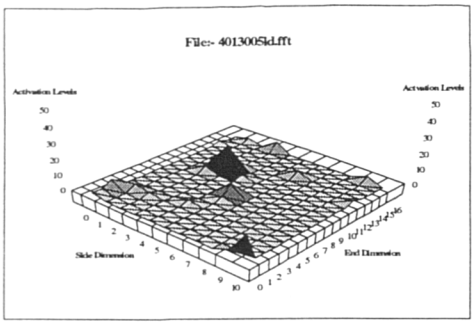
FIGURE 4.9a-d
Typical FFT Traces for Travel Speed Changes



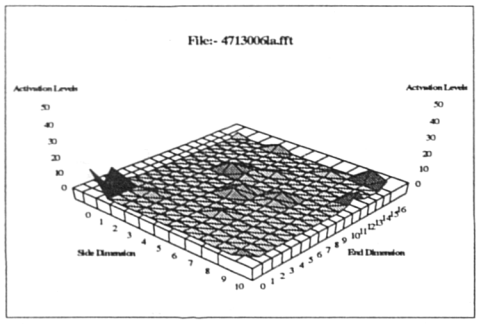
a) 25 Volts



b) 35 Volts

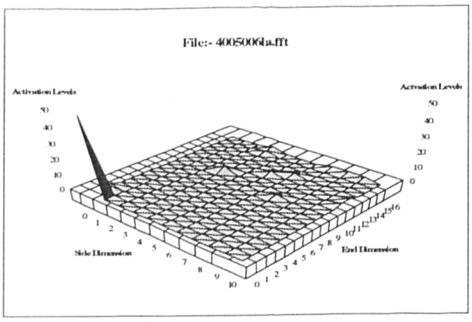


c) 40 Volts

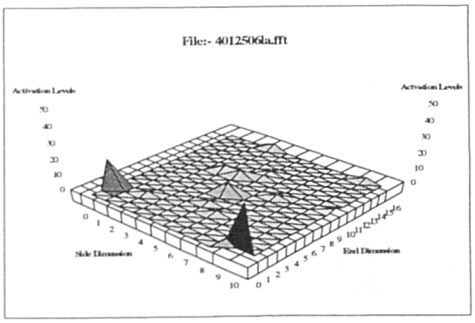


d) 47 Volts

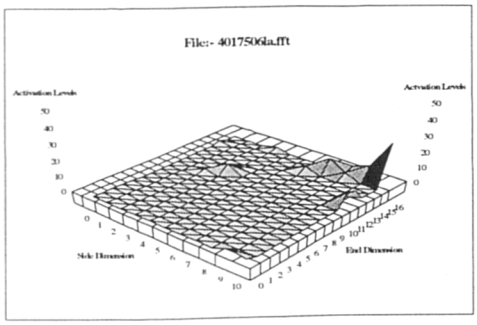
FIGURE 4.10a-d
Typical Activation Maps for Voltage Changes



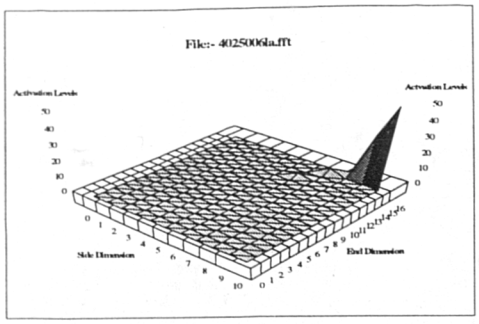
a) 175 Amps



b) 460 Amps

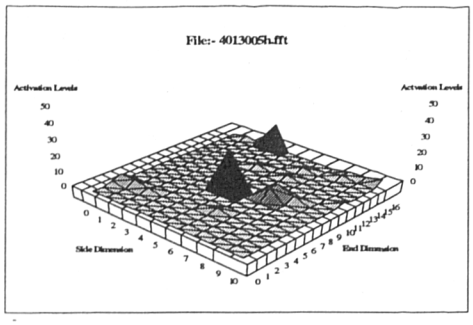


c) 700 Amps

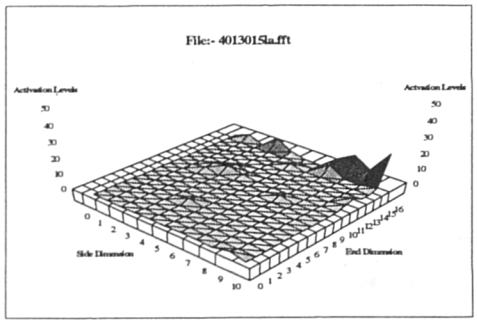


d) 930 Amps

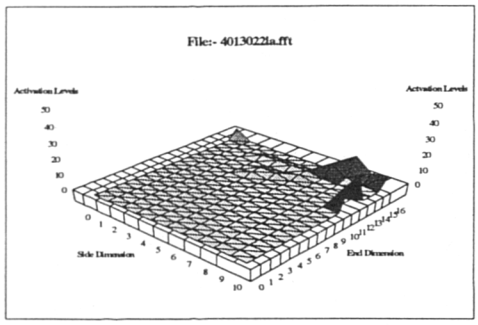
FIGURE 4.11a-d
Typical Activation Maps for Current Changes



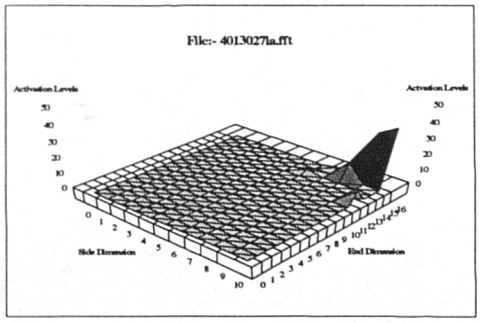
a) 5 mm/s



b) 15 mm/s



c)22 mm/s



d) 27 mm/s

FIGURE 4.12a-d
Typical Activation Maps for Travel Speed Changes

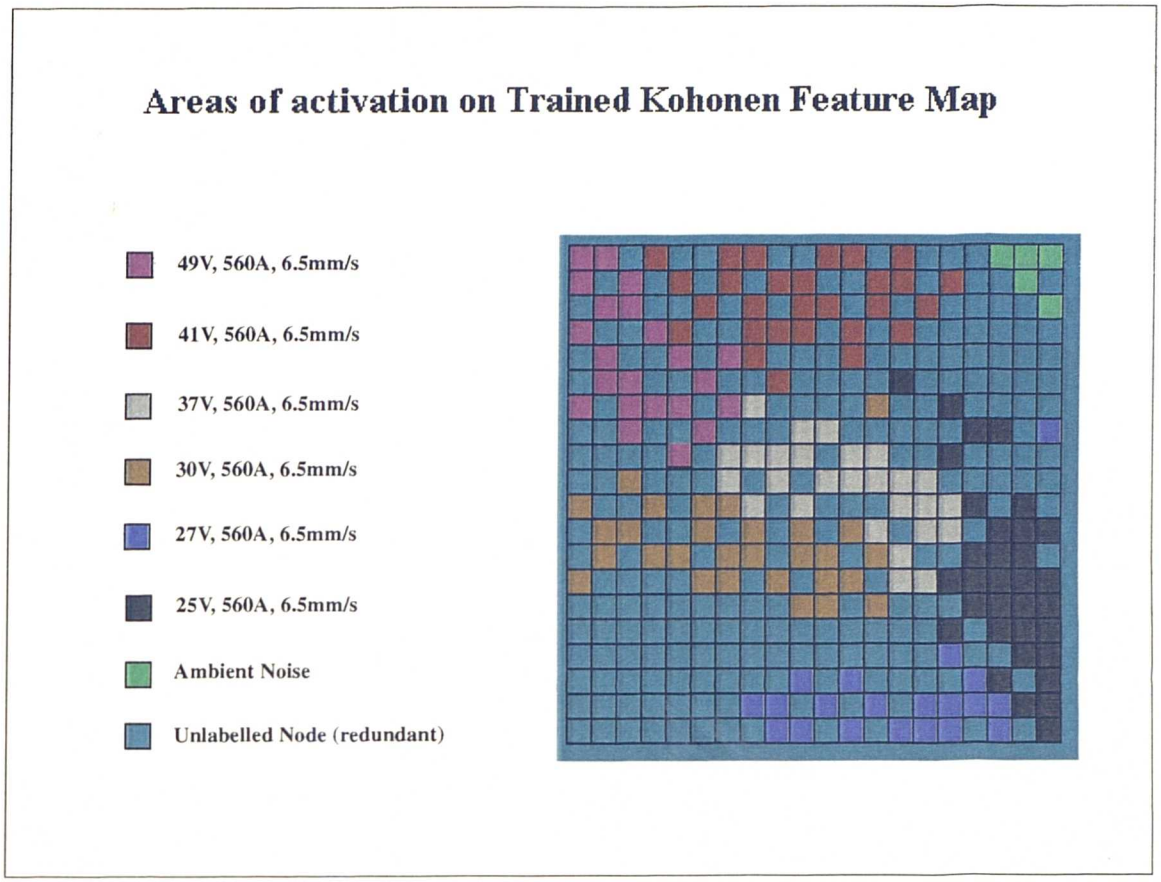


FIGURE 4.13a
Code Book Vector labelling

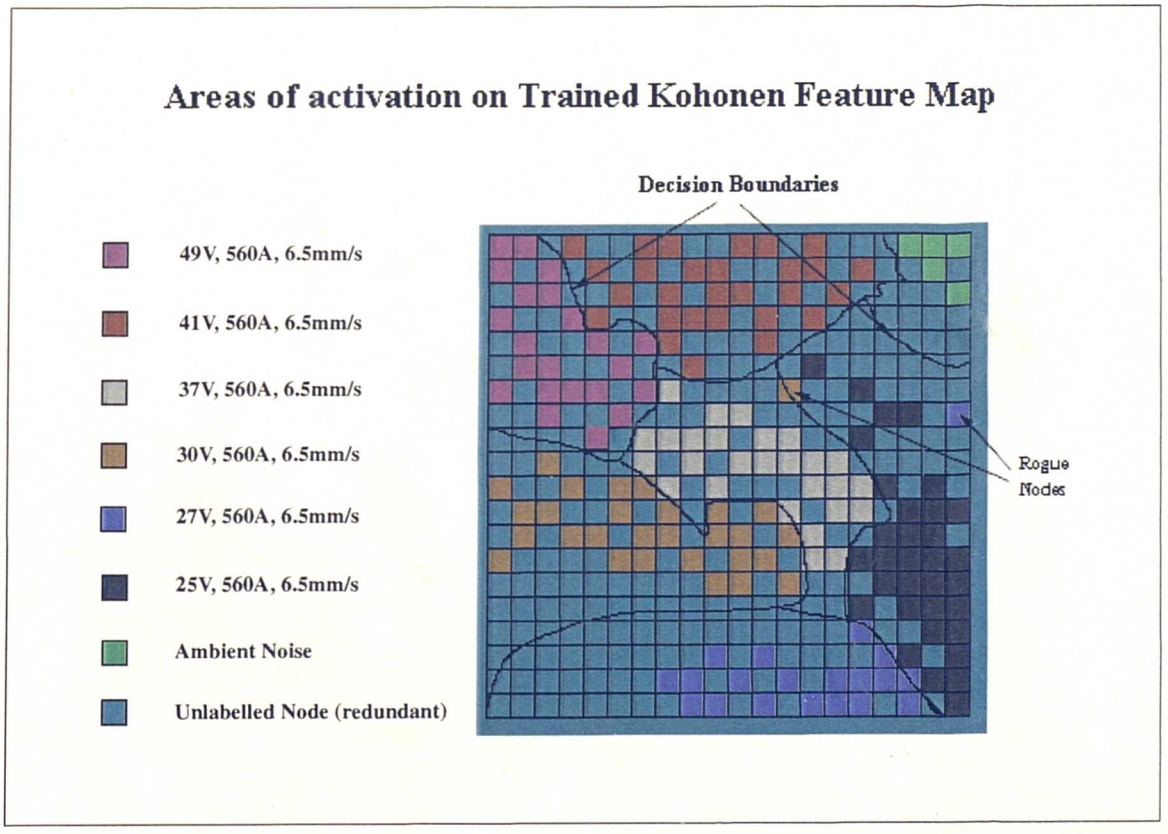


FIGURE 4.13b
Code Book Vector labelling (Boundary Decisions)

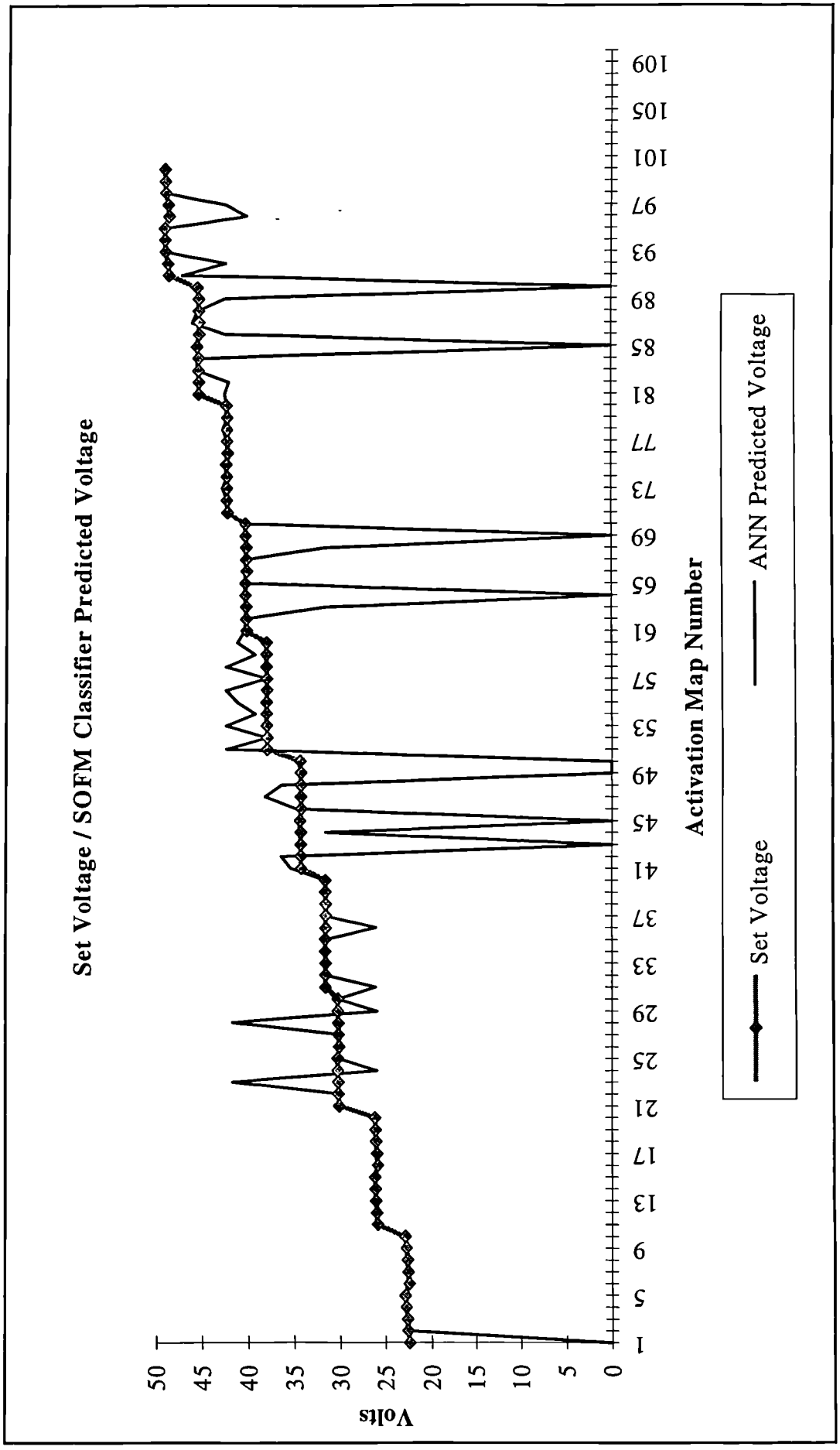


FIGURE 4.14a
Code Book Vector Labelling Results (Voltage Test Data)

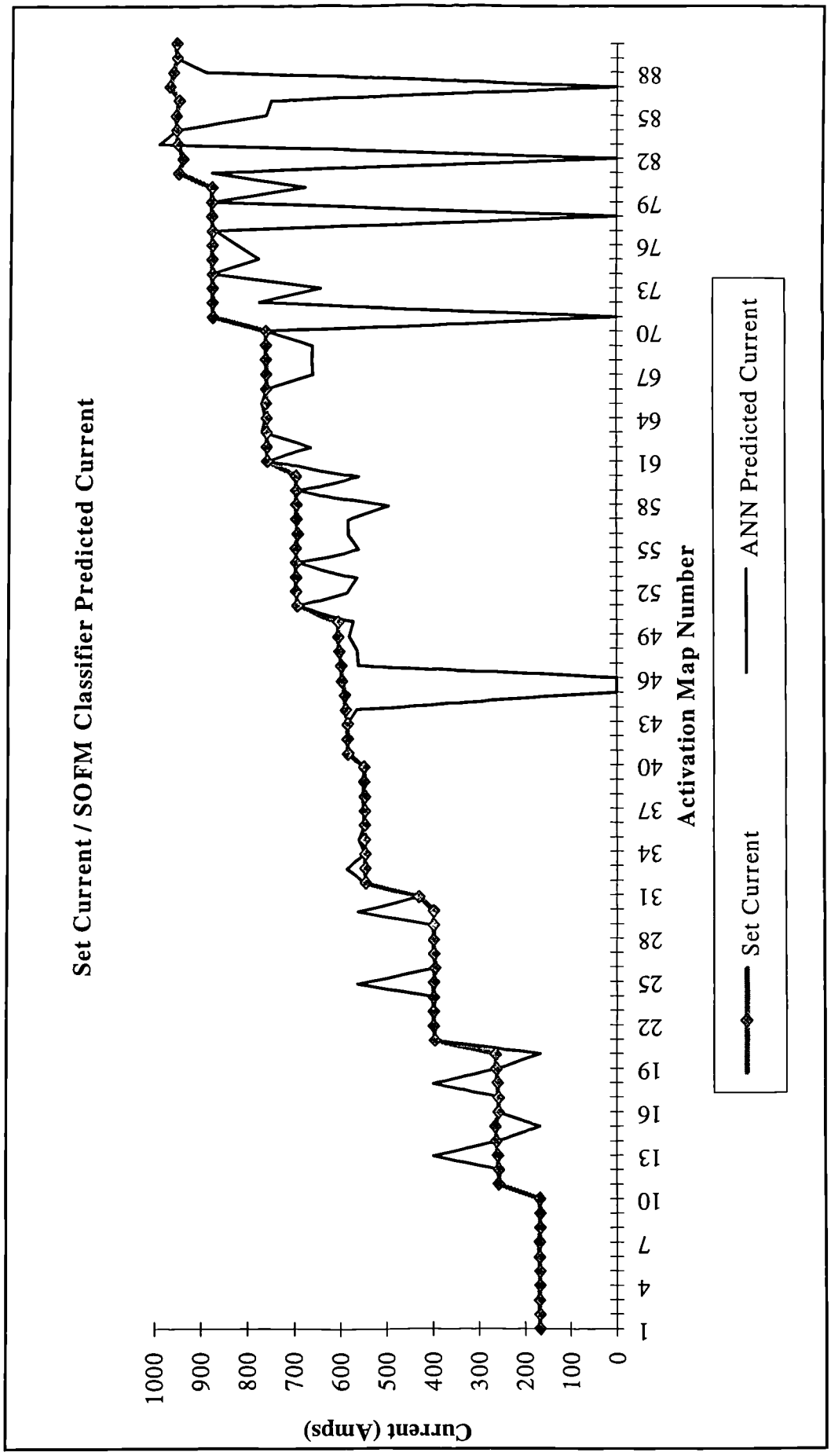


FIGURE 4.14b
Code Book Vector Labelling Results (Current Test Data)

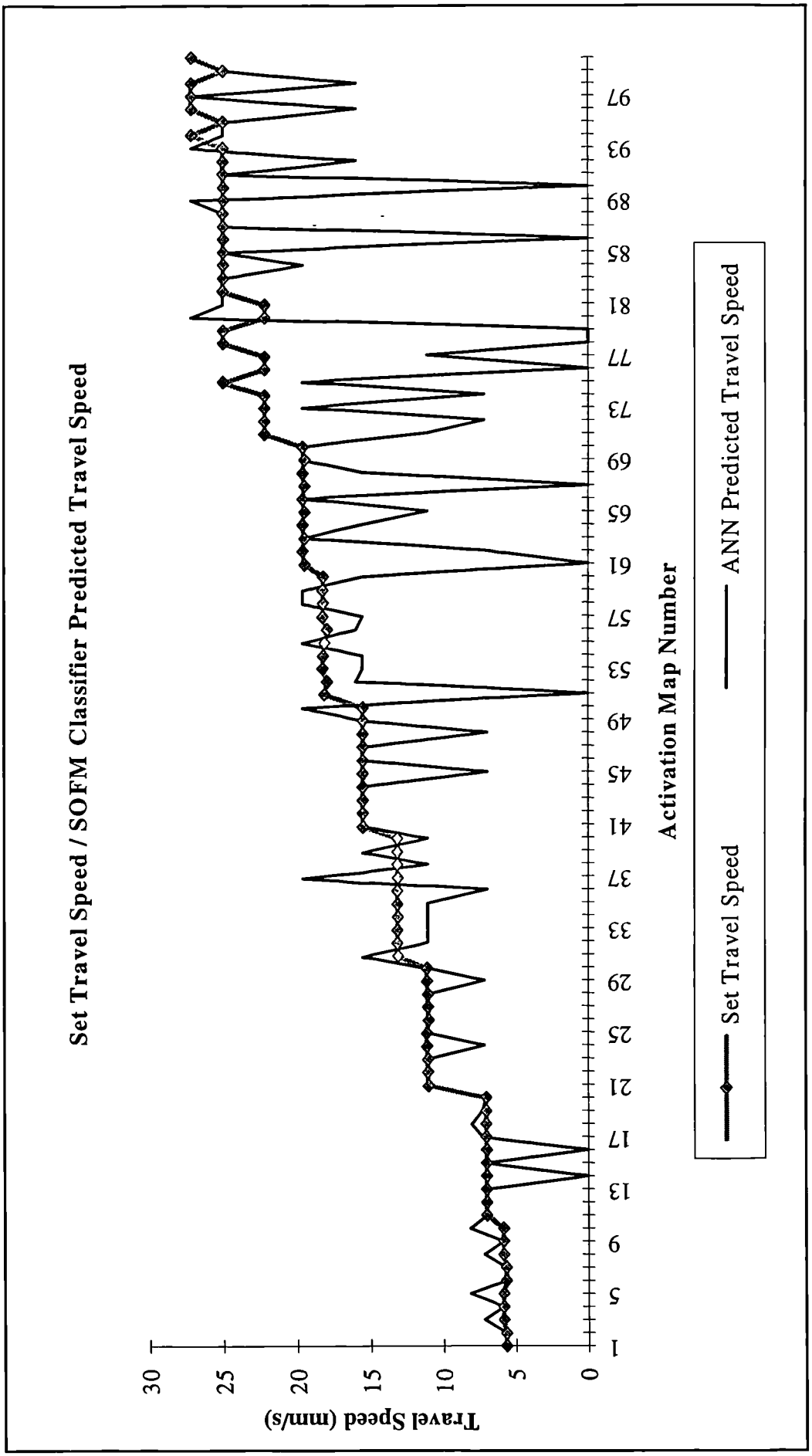


FIGURE 4.14c
Code Book Vector Labelling Results (Travel Speed Test Data)

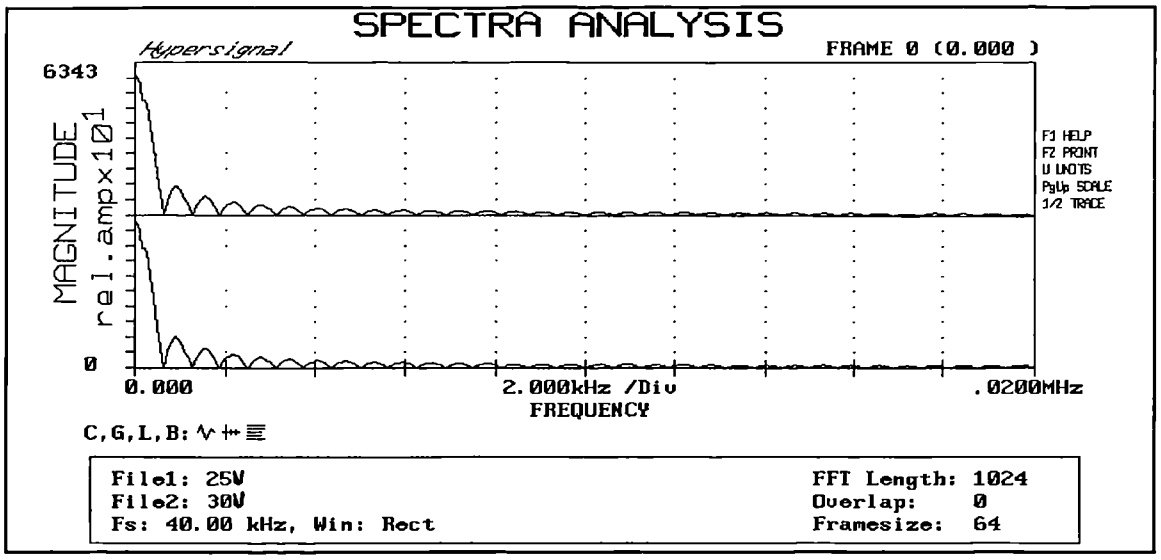


FIGURE 4.15a
Cepstral Analysis - 25volts - 30volts

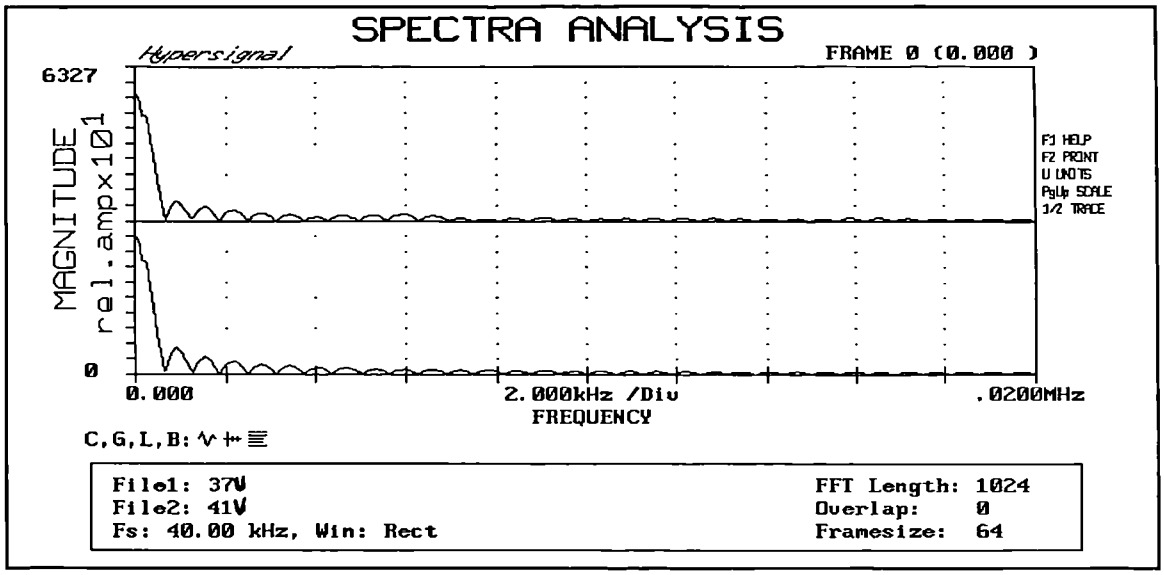


FIGURE 4.15b
Cepstral Analysis - 37volts - 41volts

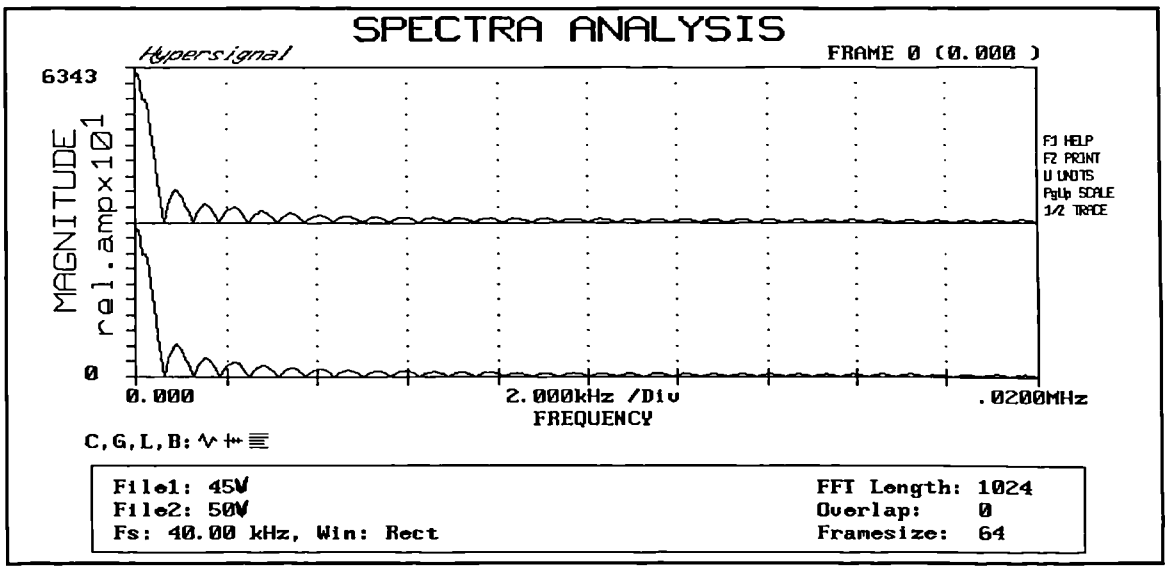


FIGURE 4.15c
Cepstral Analysis - 45volts - 50volts

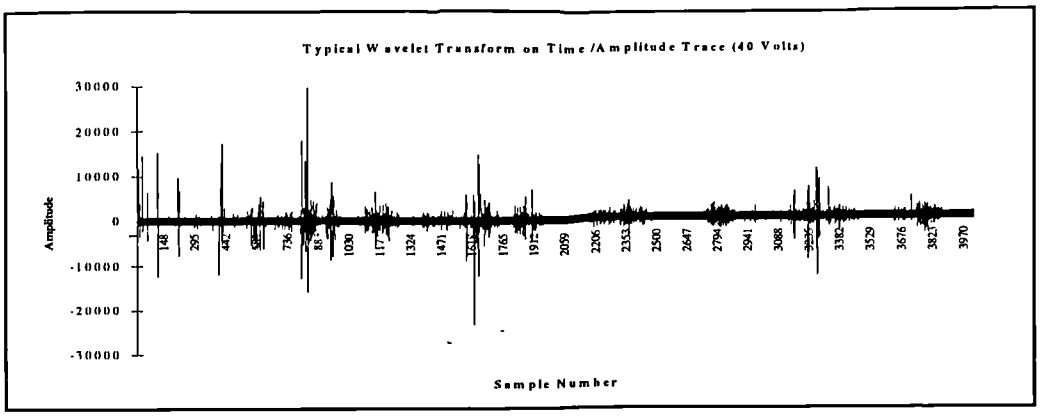


FIGURE 4.16
Typical Time / Amplitude Wavelet Transform

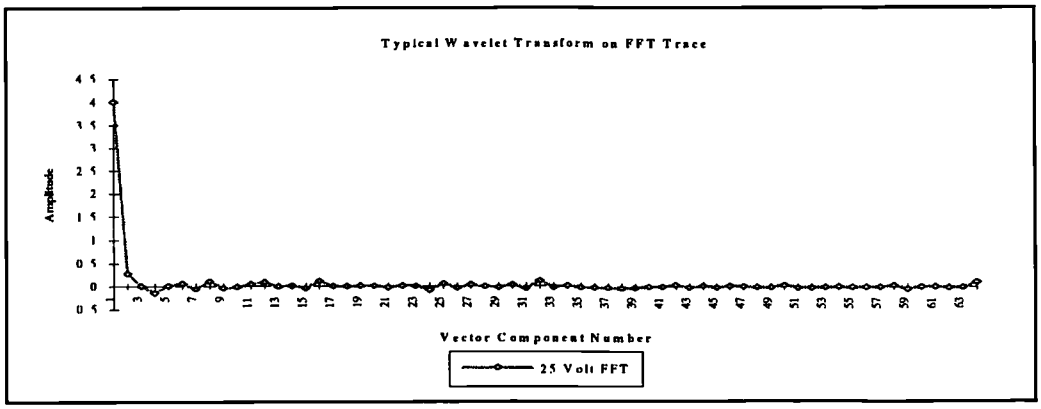


FIGURE 4.17a
Typical FFT Wavelet Transform (25 Volts)

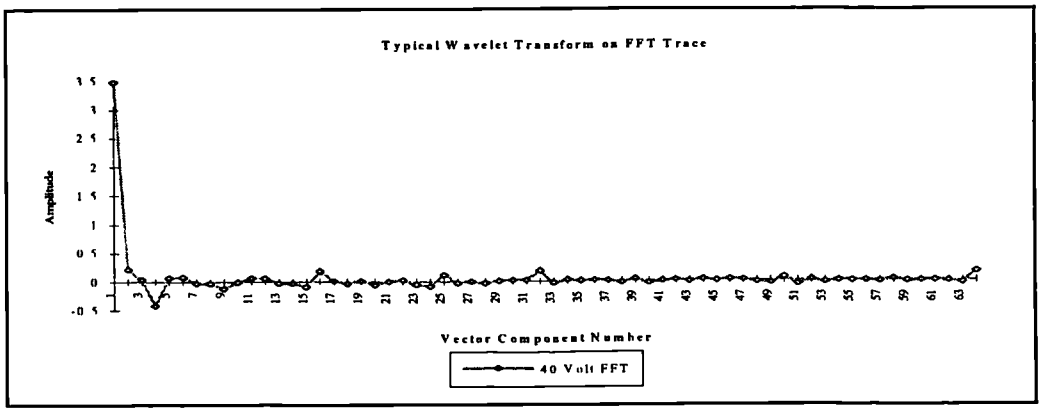


FIGURE 4.17b
Typical FFT Wavelet Transform (40 Volts)

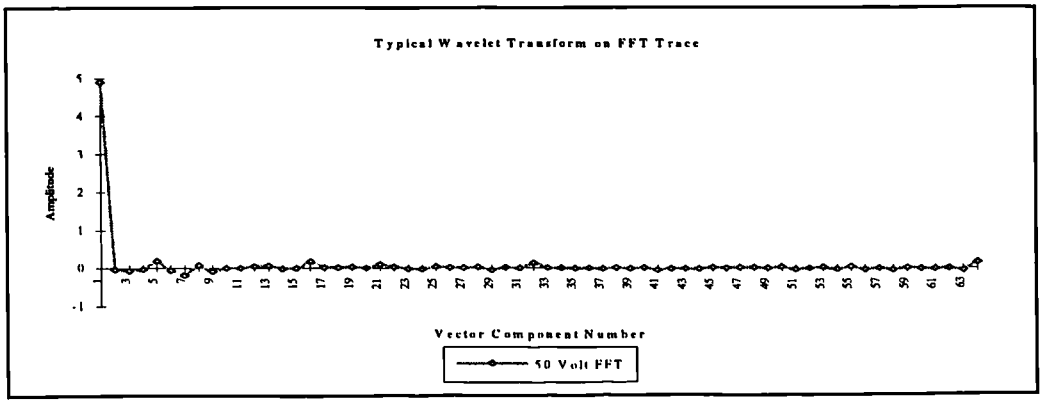


FIGURE 4.17c
Typical FFT Wavelet Transform (50 Volts)

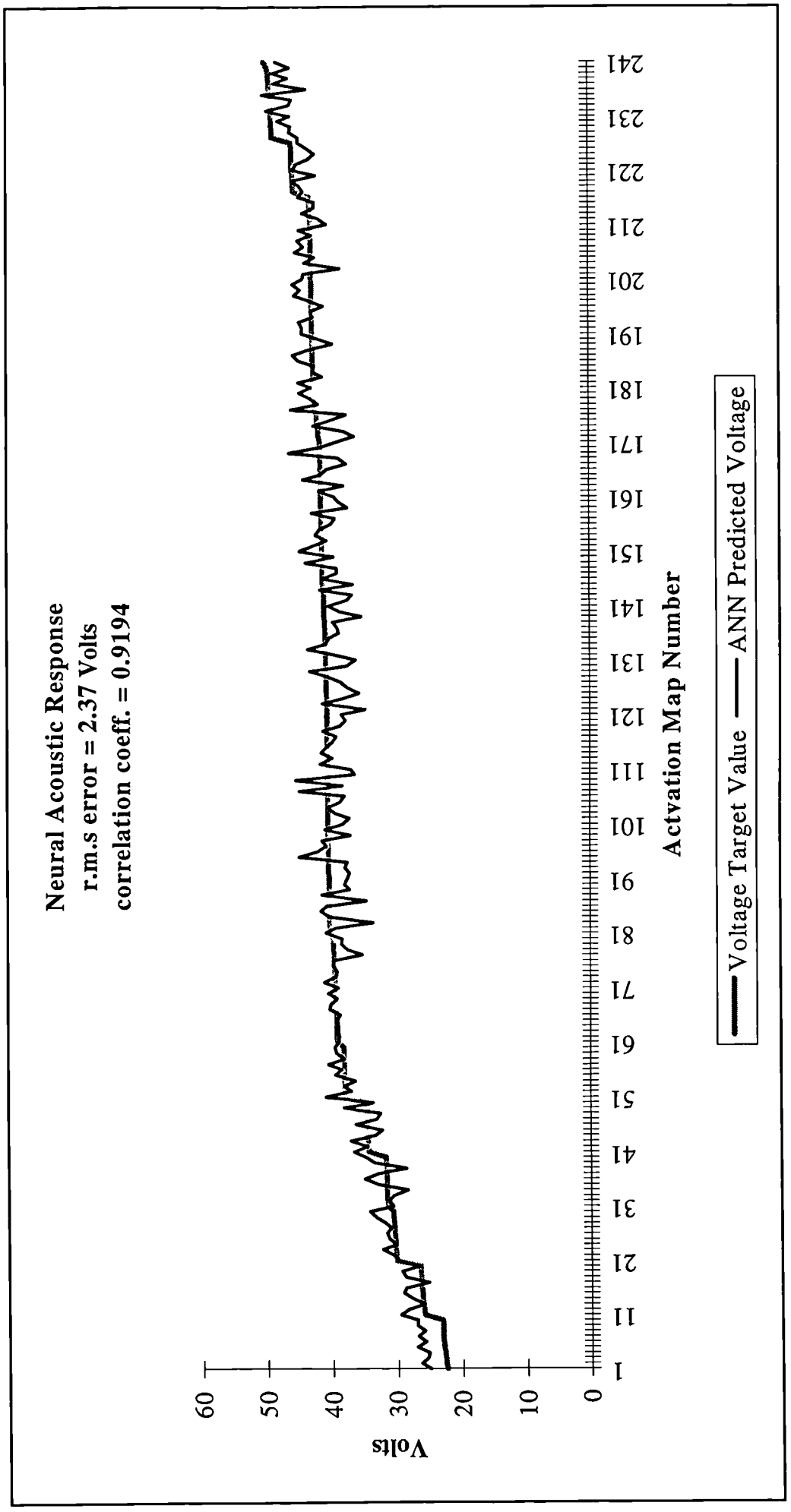


FIGURE 4.18
Off-Line Prediction Figures for Voltage Monitoring

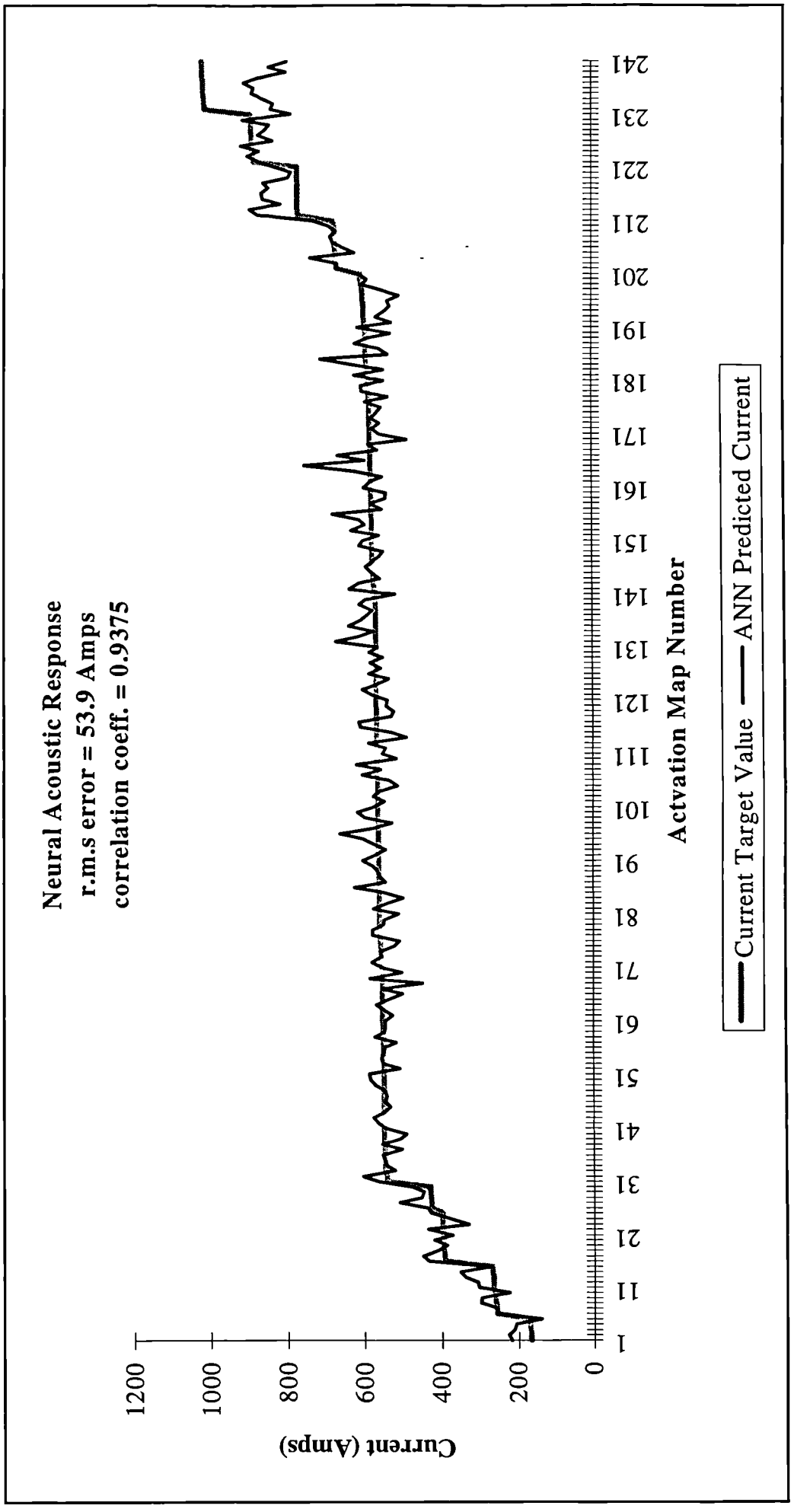


FIGURE 4.19
Off-Line Prediction Figures for Current Monitoring

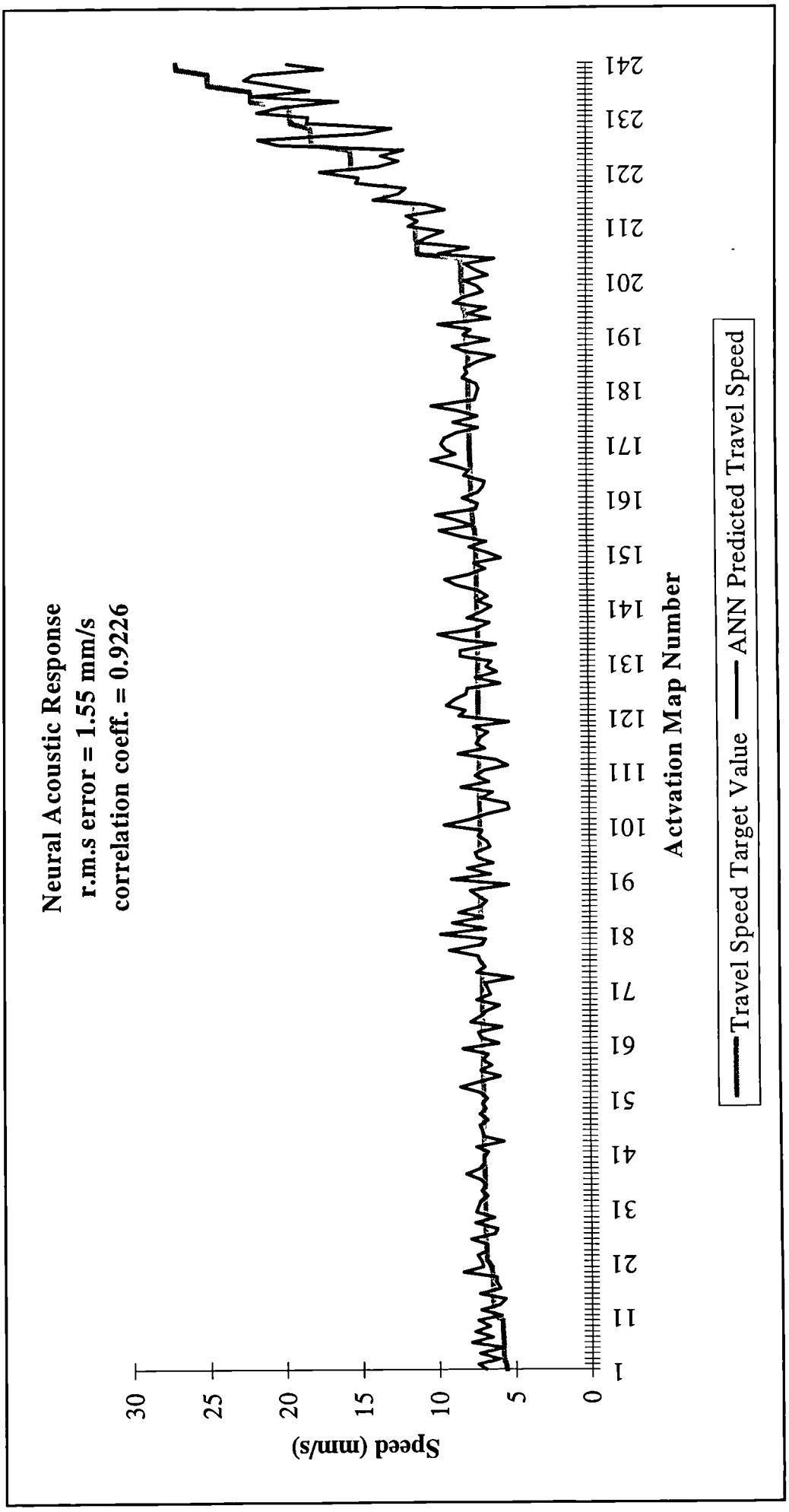


FIGURE 4.20
Off-Line Prediction Figures for Travel Speed Monitoring

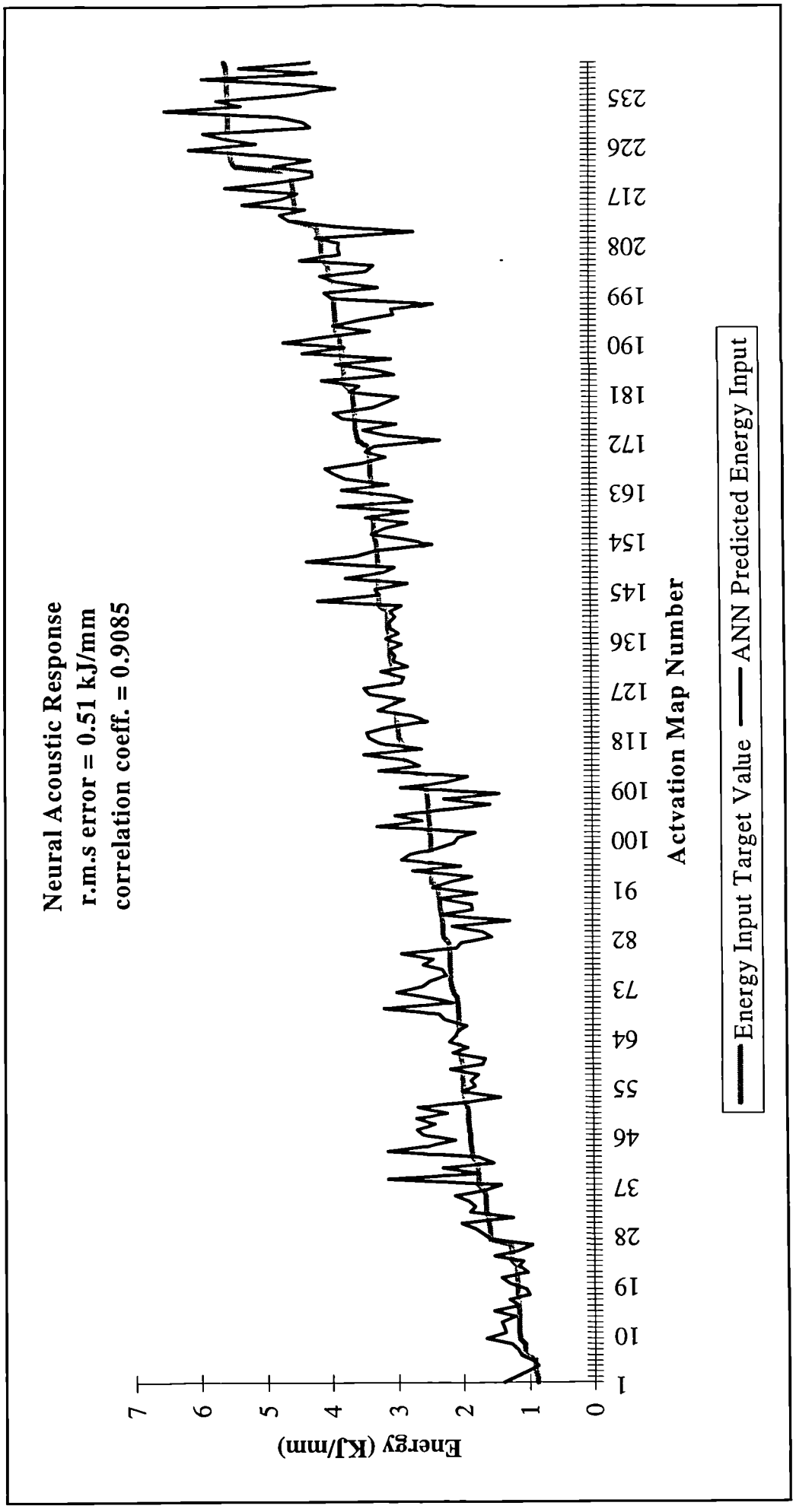


FIGURE 4.21
Off-Line Prediction Figures for Energy Monitoring

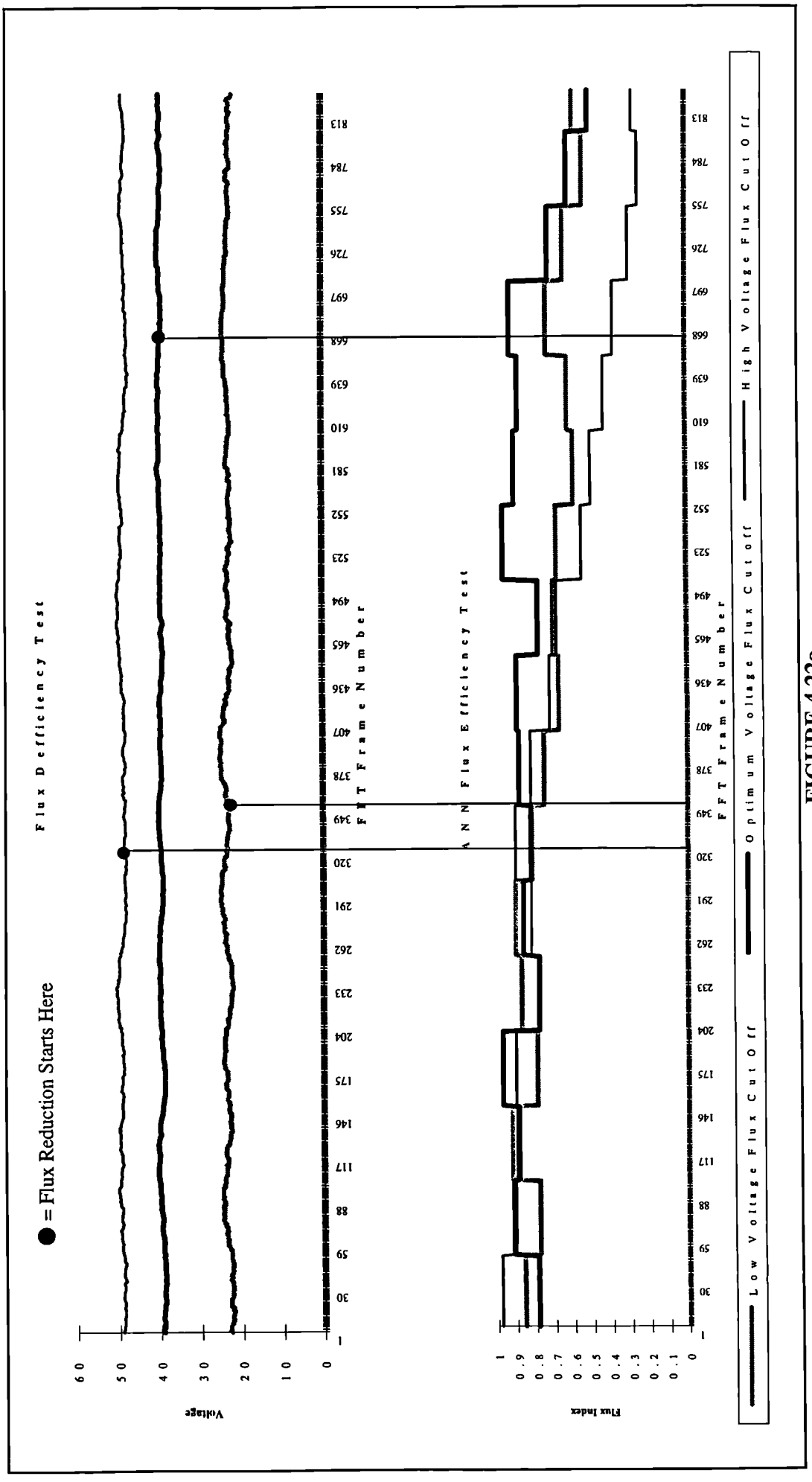


FIGURE 4.22a
Flux Coverage Depletion Test on Changing Voltage

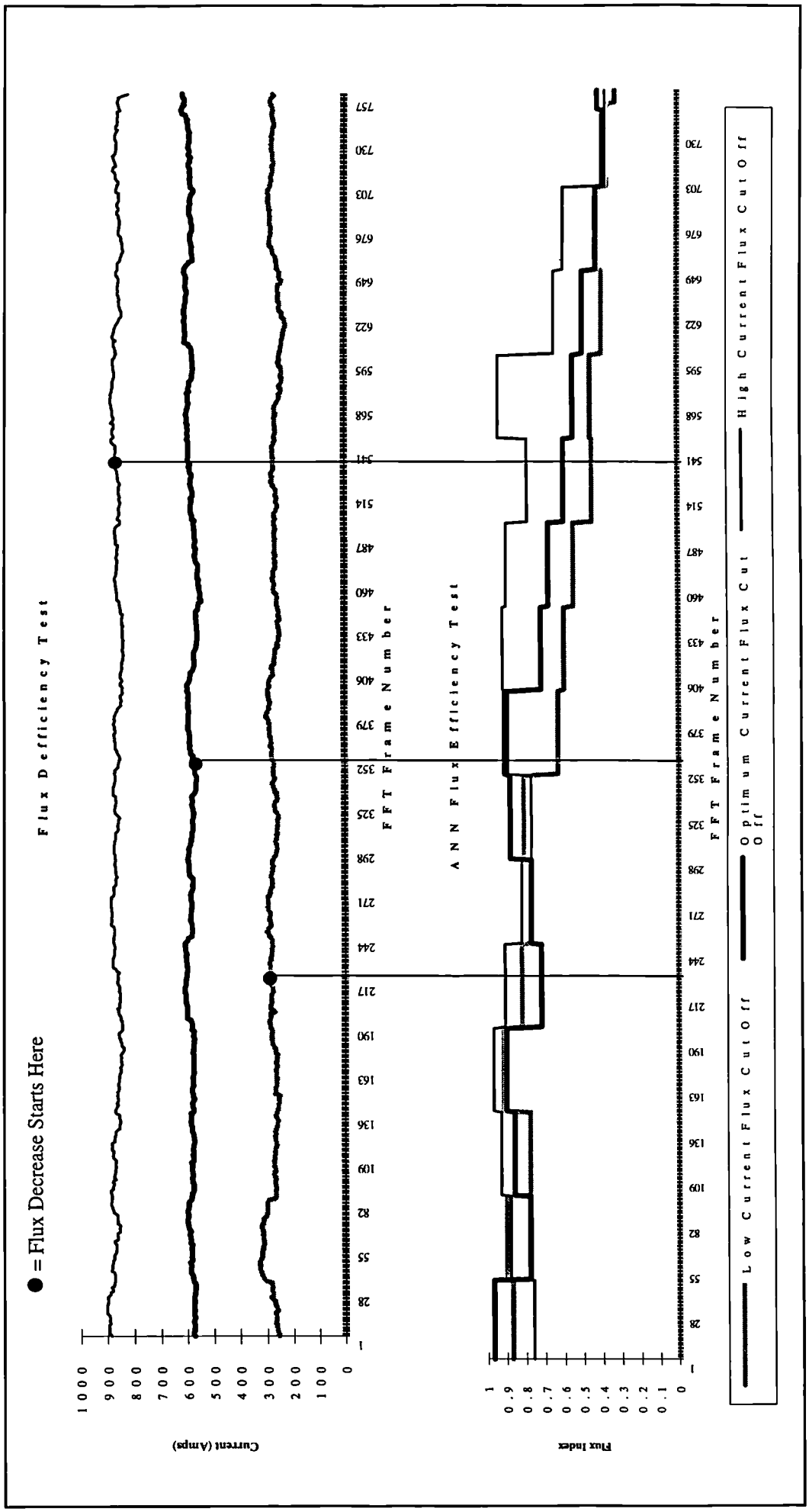


FIGURE 4.22b
Flux Coverage Depletion Test on Changing Current

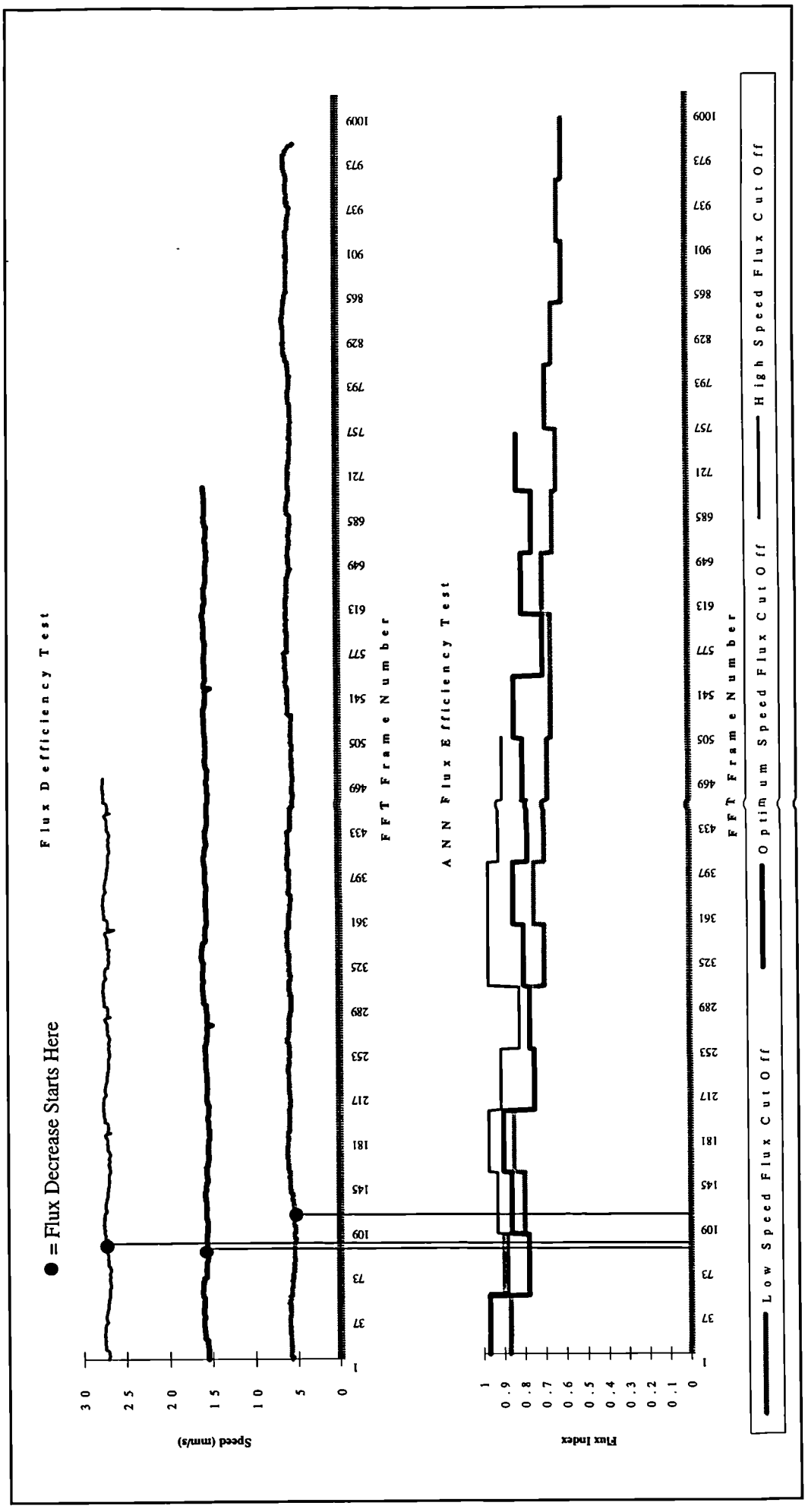
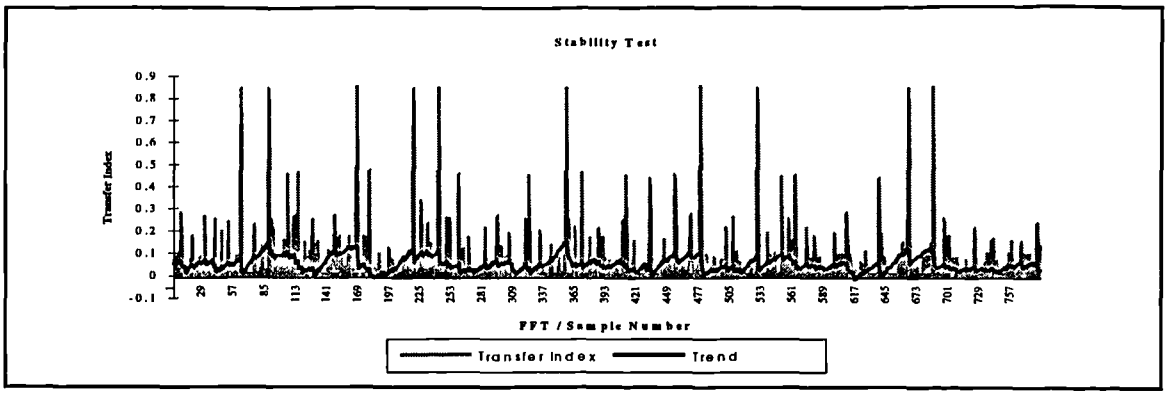
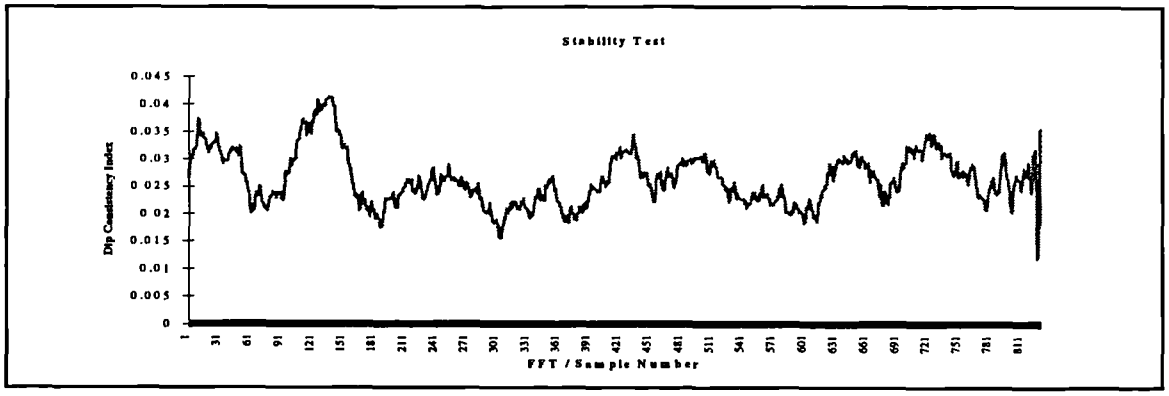


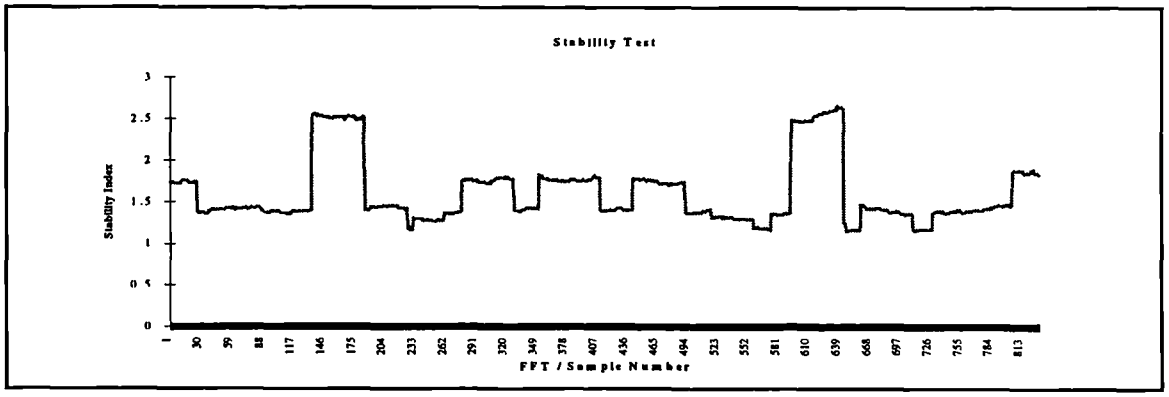
FIGURE 4.22c
Flux Coverage Depletion Test on Changing Travel Speed



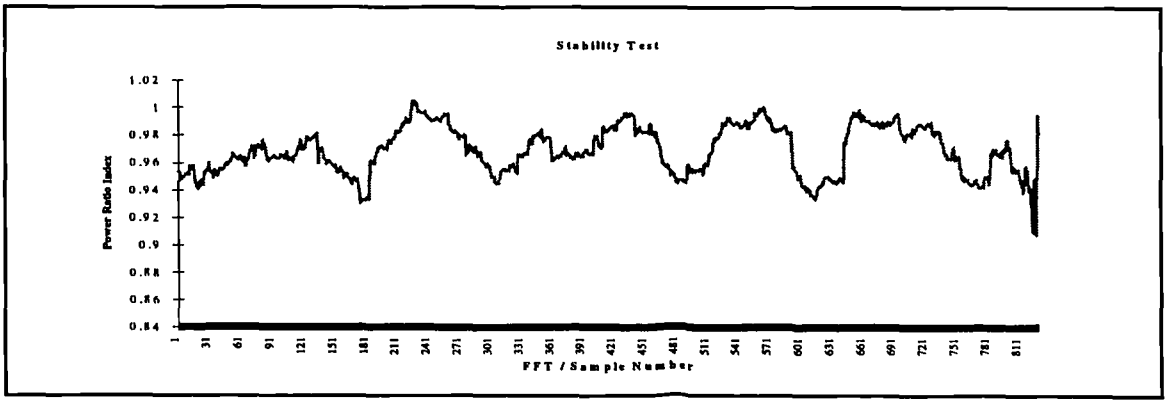
a) Transfer Index



b) Dip Consistency Index

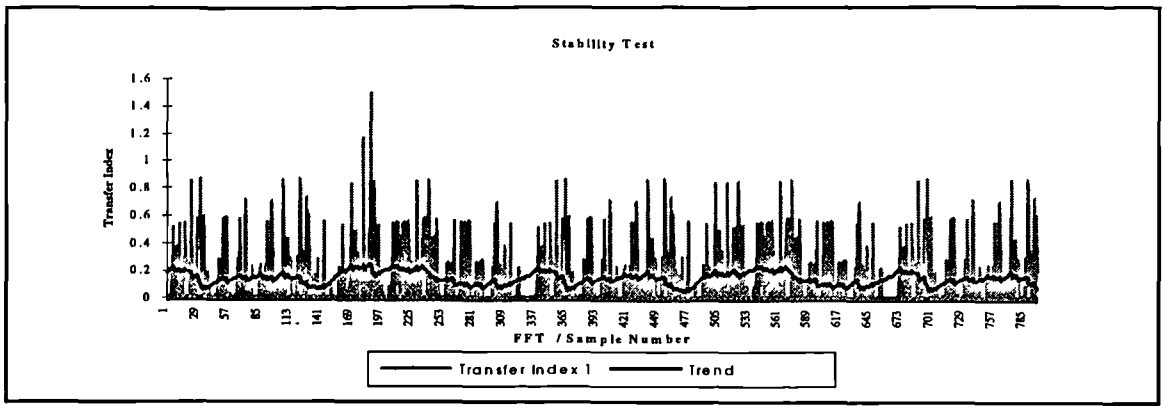


c) Transfer Stability Index

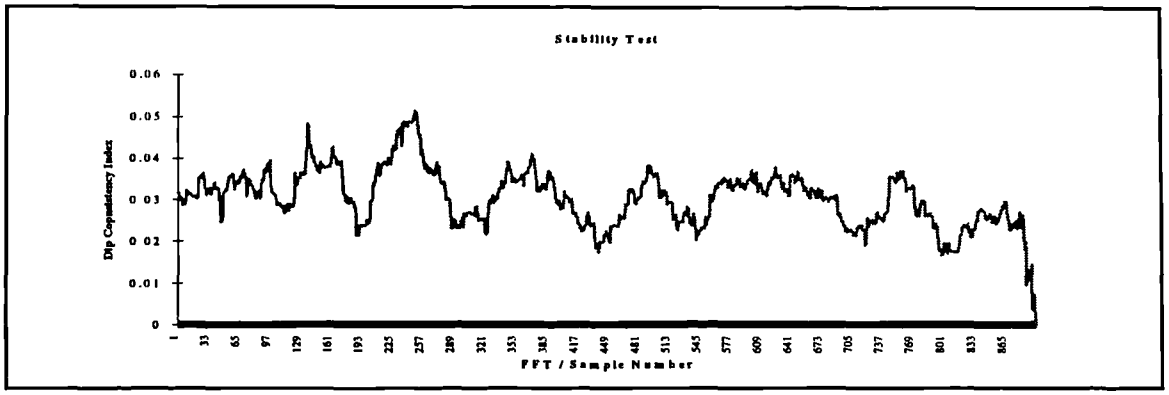


d) Power Ratio Index

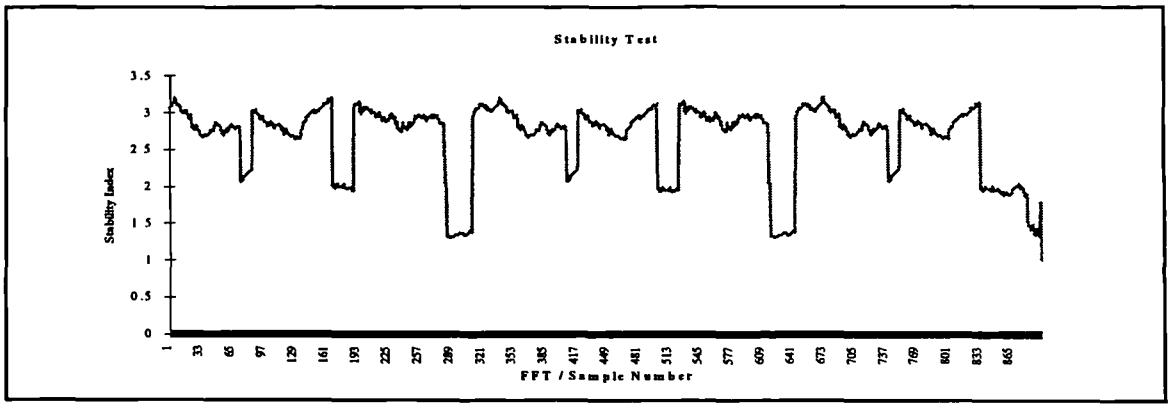
FIGURE 4.23a-d
Typical Calculated Stability Indices - Ideal Weld



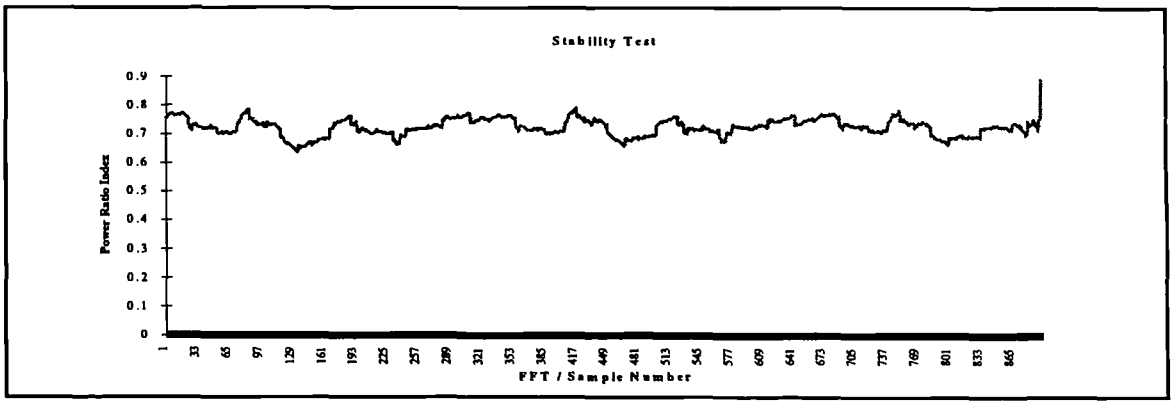
a) Transfer Index



b) Dip Consistency Index



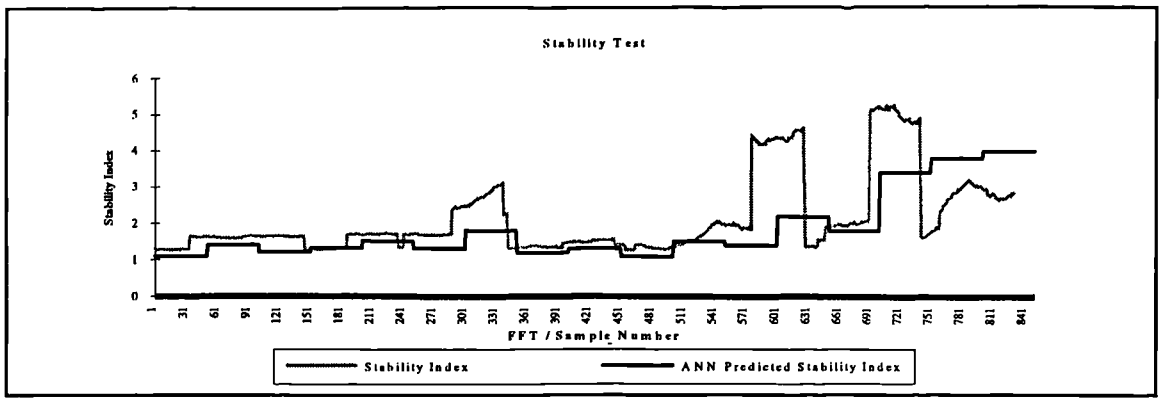
c) Transfer Stability Index



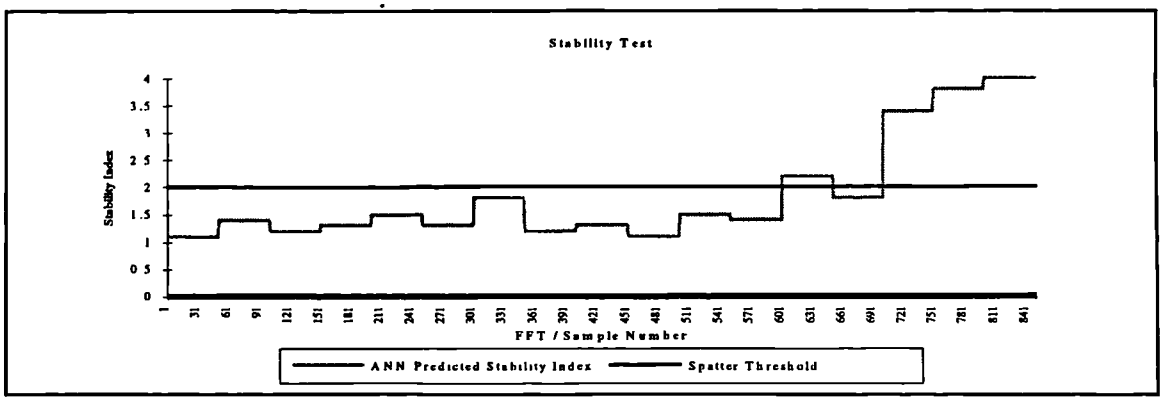
d) Power Ratio Index

FIGURE 4.24a-d

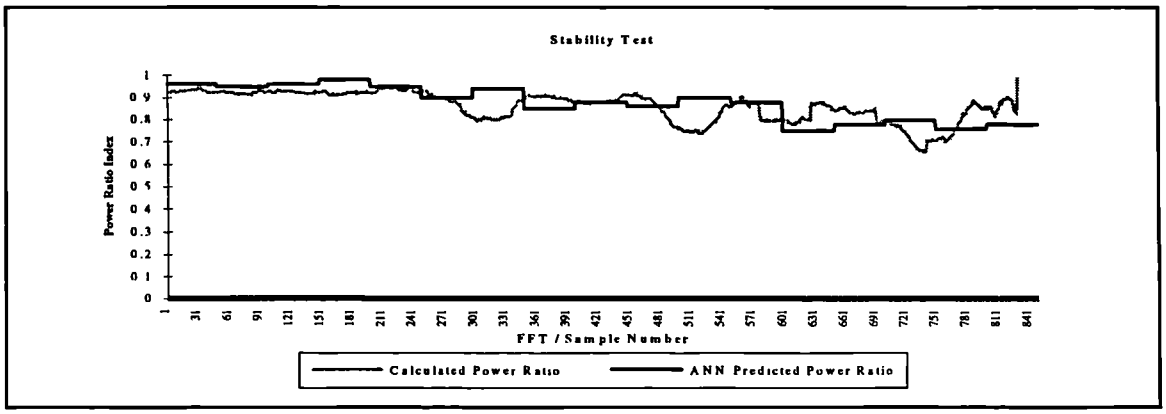
Typical Calculated Stability Indices - Forced Unstable (High Volts, Low Current)



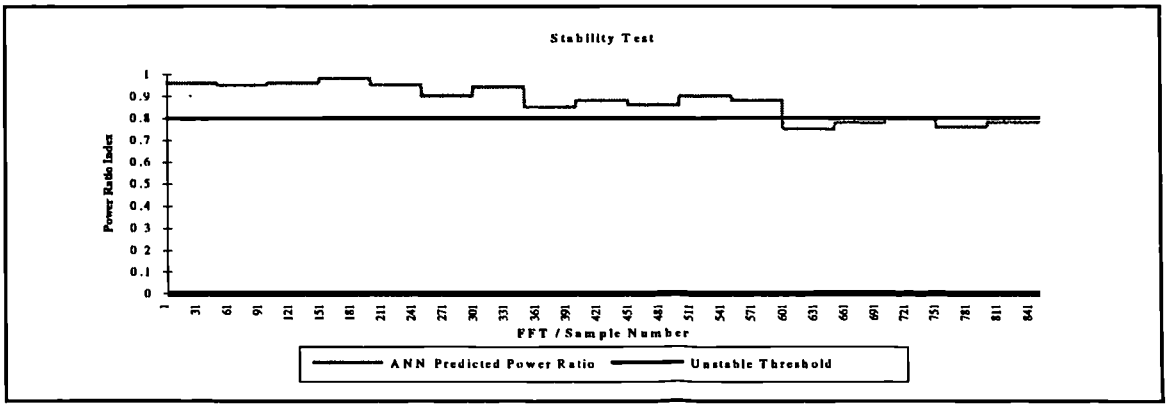
a) Stability Index



b) Threshold



c) Power Ratio Index



d) Threshold

FIGURE 4.25a-d
Stability Indices - Parameter Change / Neural Response

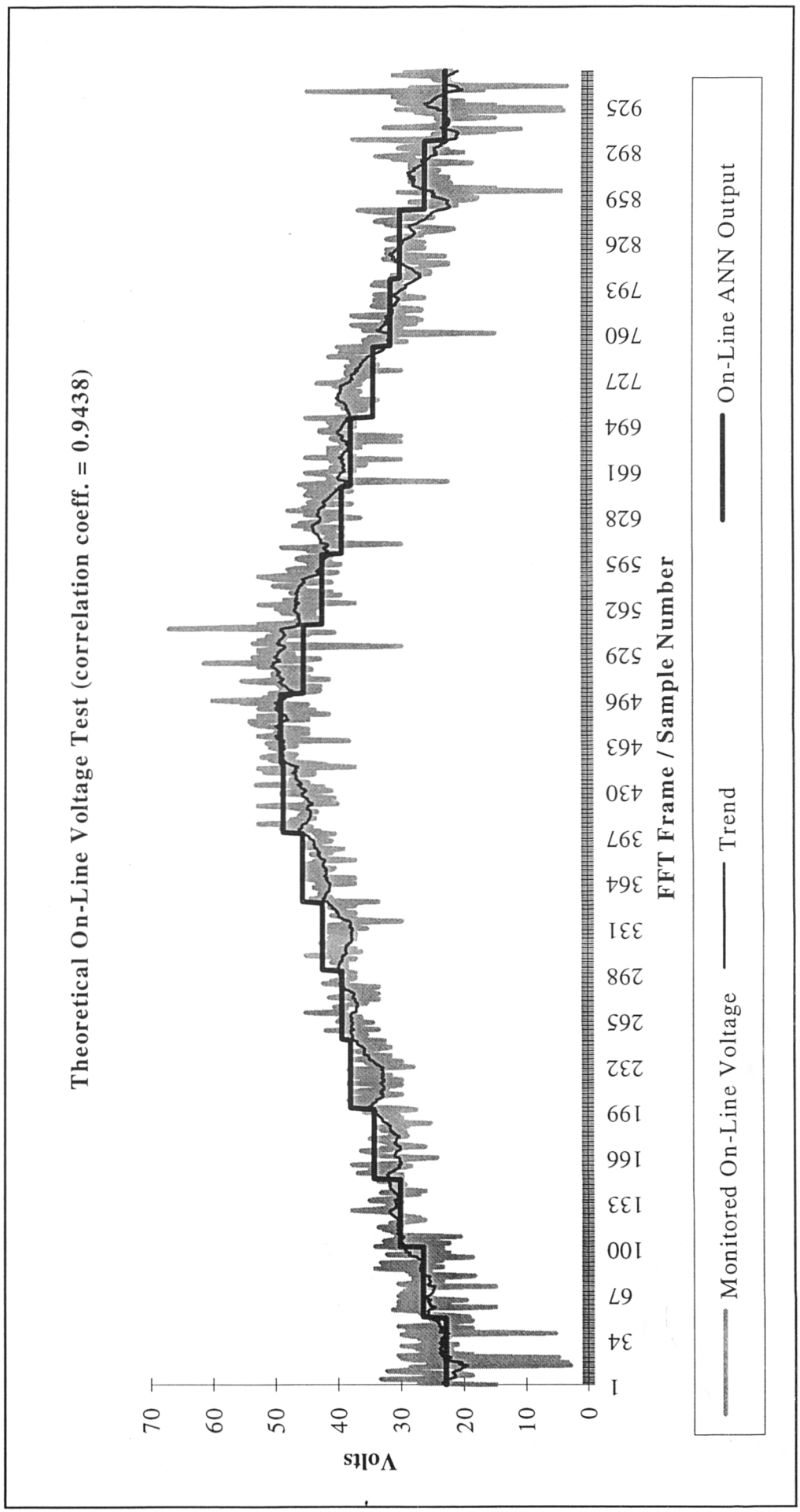


FIGURE 4.26a
Theoretical On -Line Result - Weld Voltage Monitoring

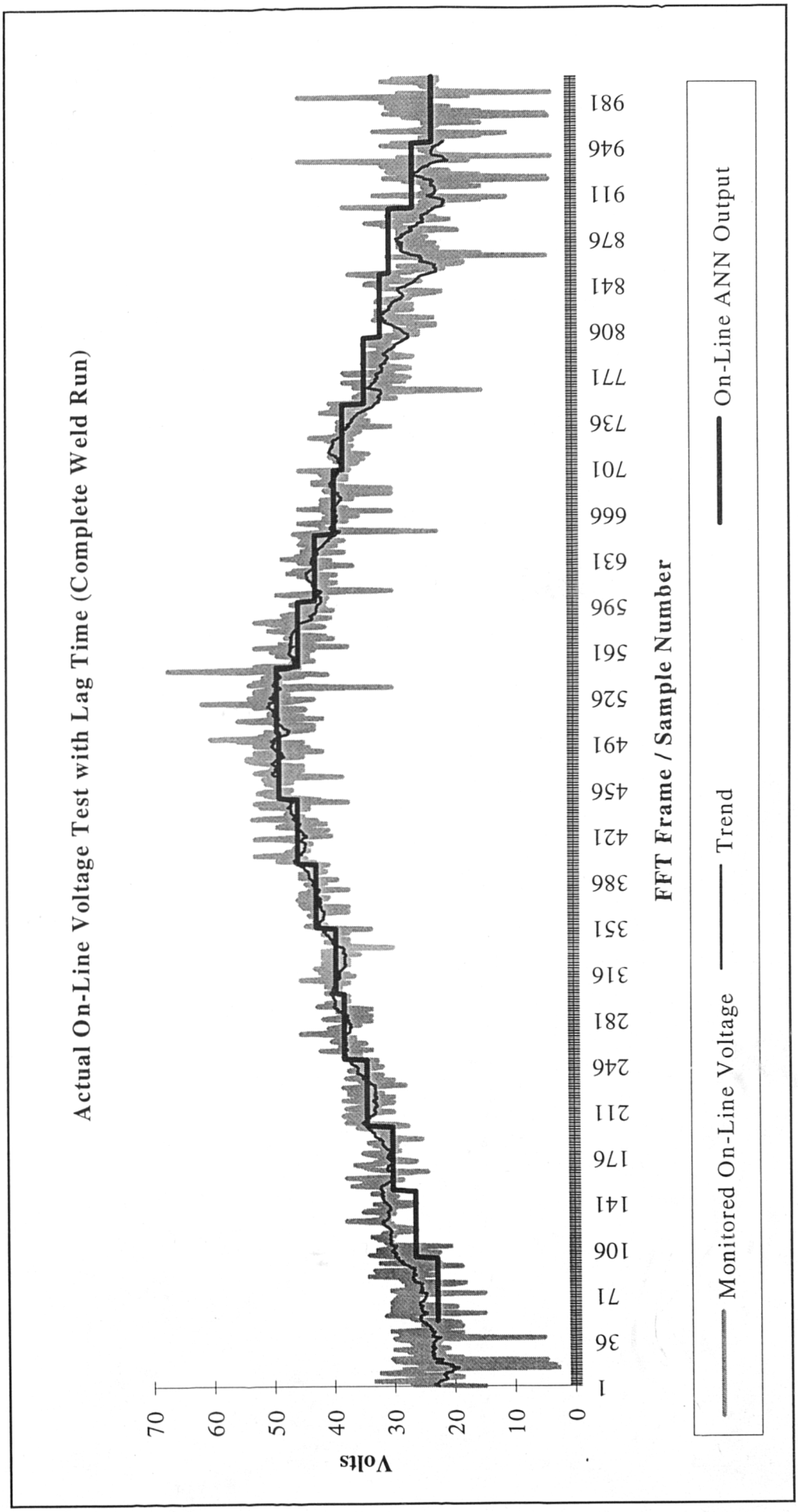


FIGURE 4.26b
Actaual On -Line Result - Weld Voltage Monitoring

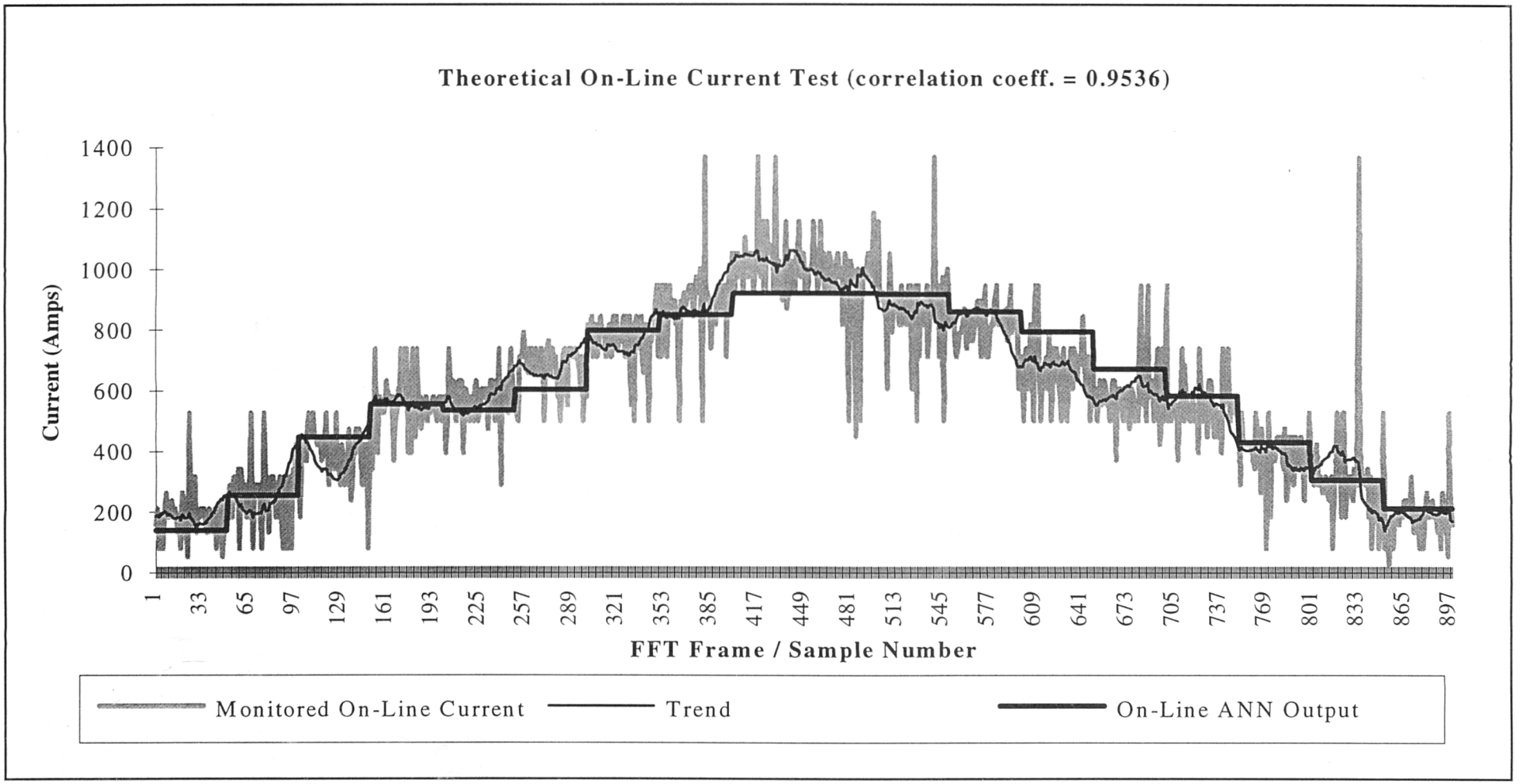


FIGURE 4.27a
Theretical On -Line Result - Weld Current Monitoring

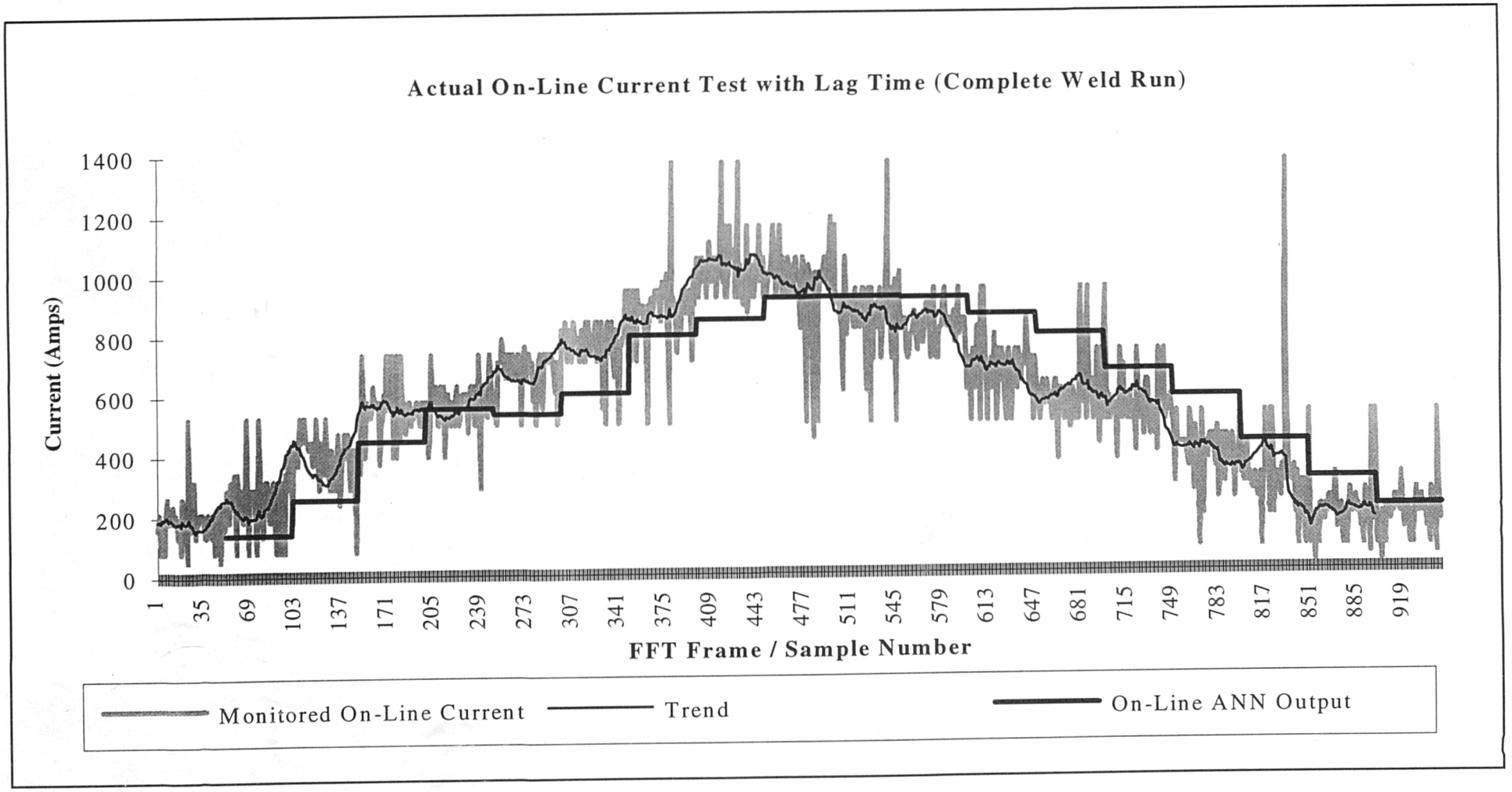


FIGURE 4.27b
Actual On -Line Result - Weld Current Monitoring

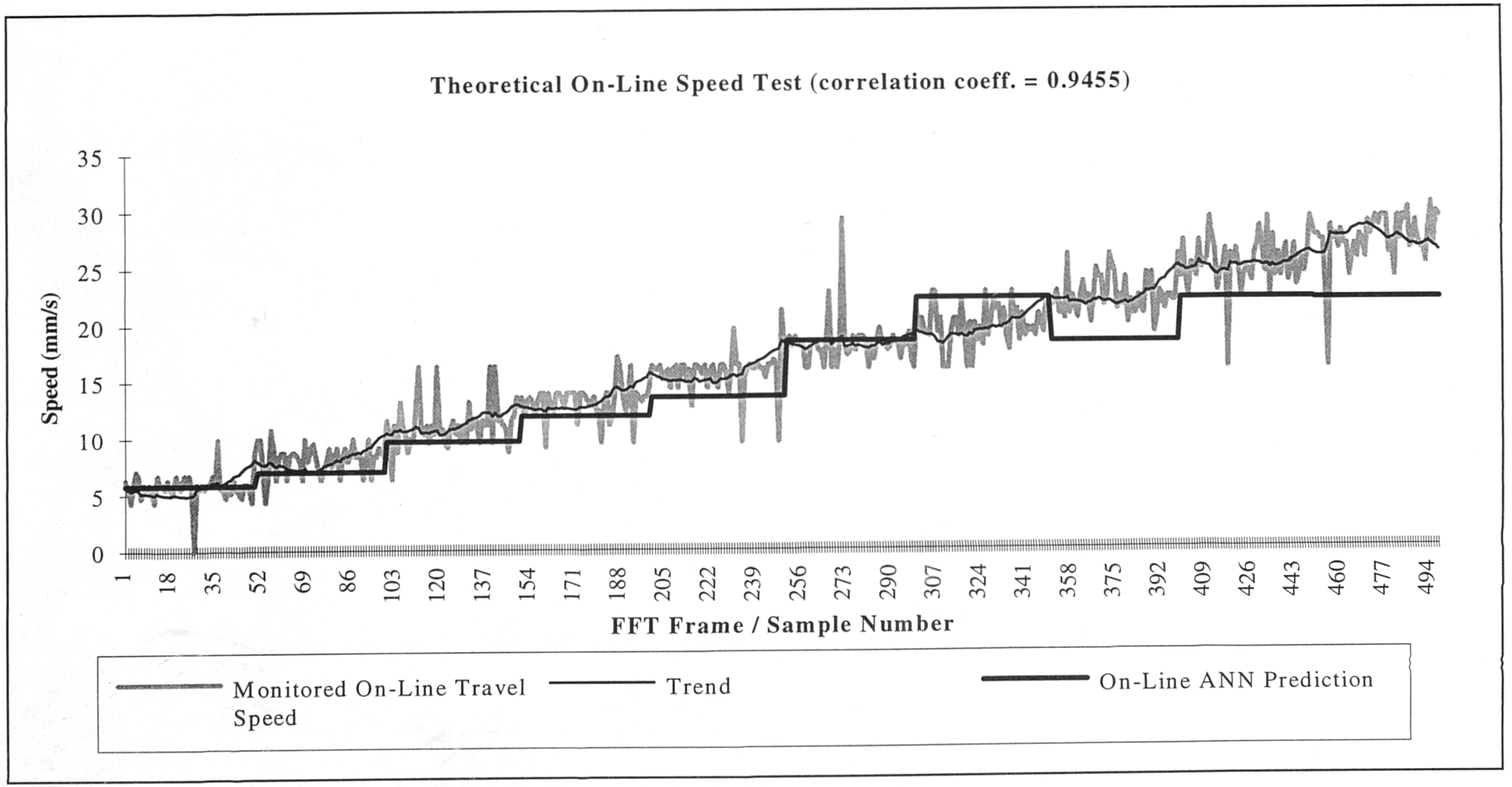


FIGURE 4.28a
Theoretical On -Line Result - Weld Travel Speed Monitoring

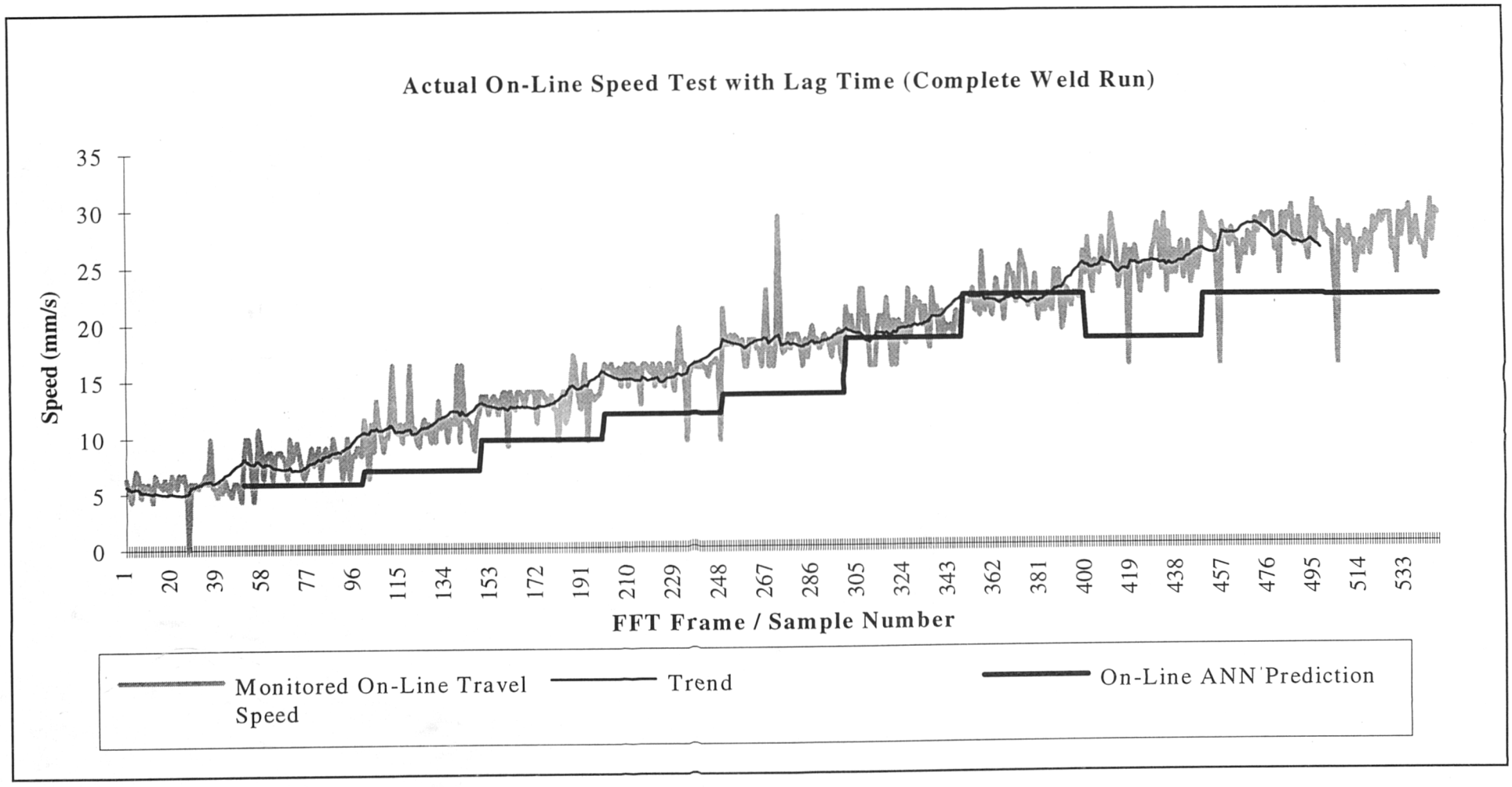


FIGURE 4.28b
Actual On -Line Result - Weld Travel Speed Monitoring

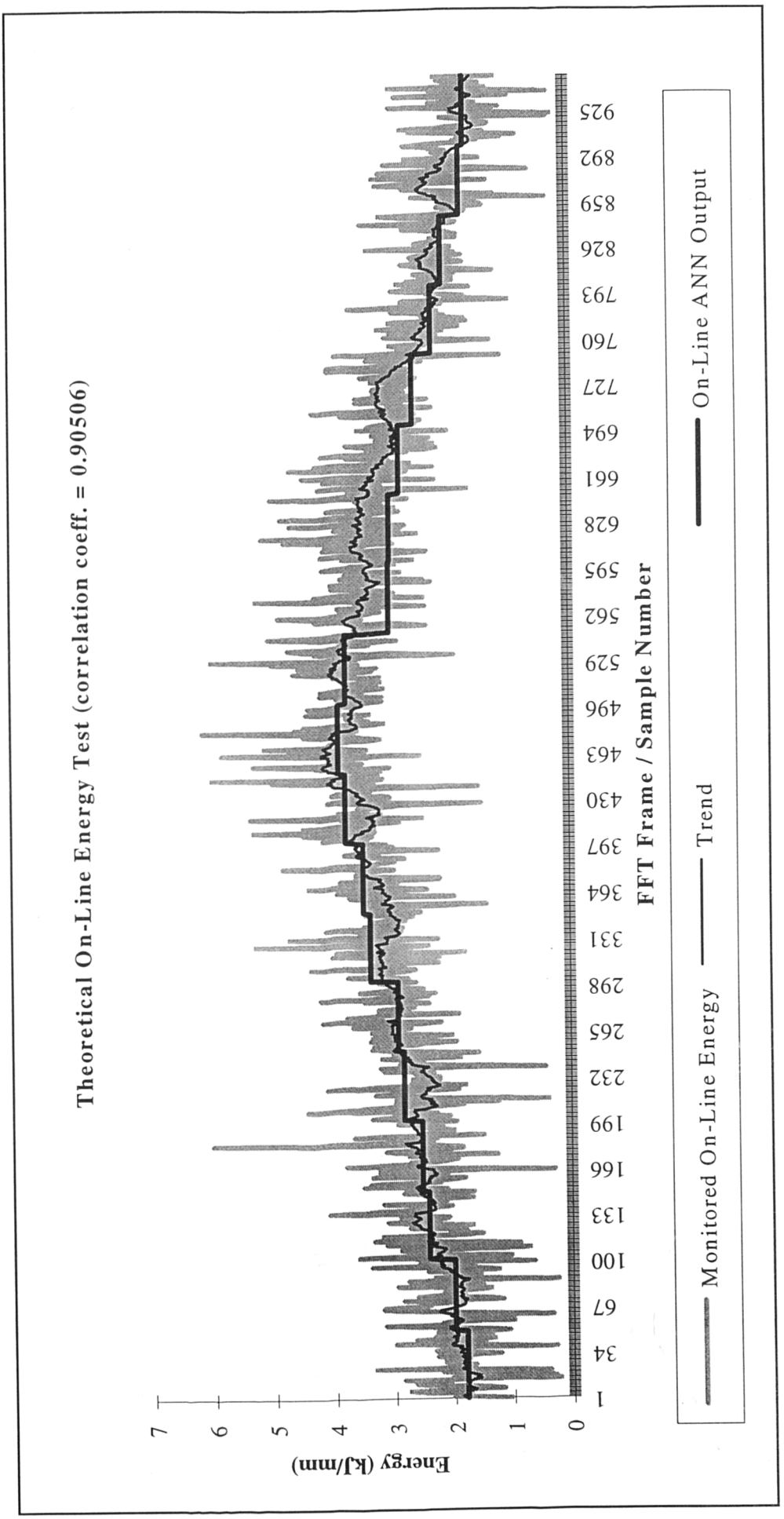


FIGURE 4.29a
Theoretical On -Line Result - Weld Energy Input Monitoring

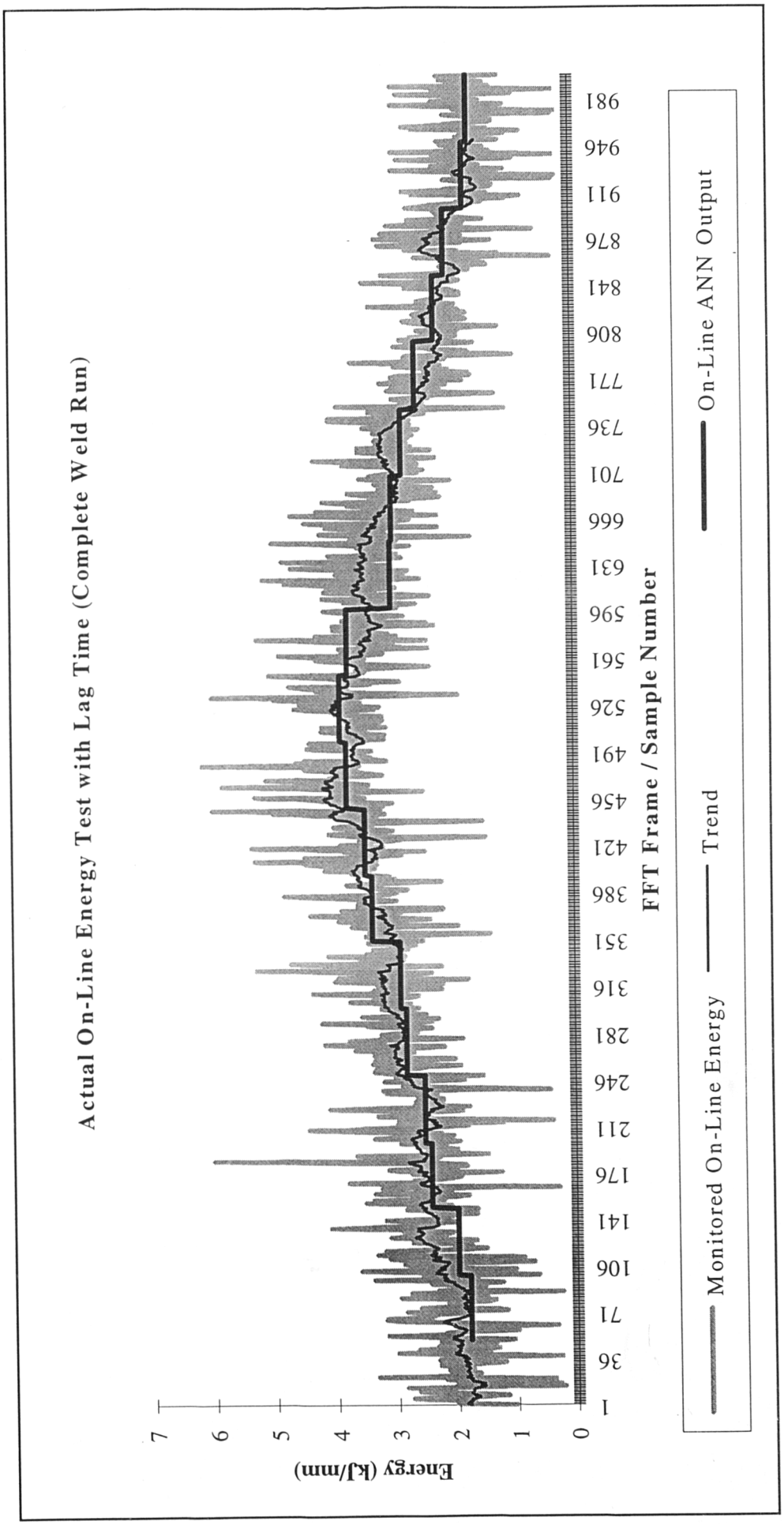


FIGURE 4.29b
Actual On -Line Result - Weld Energy Input Monitoring

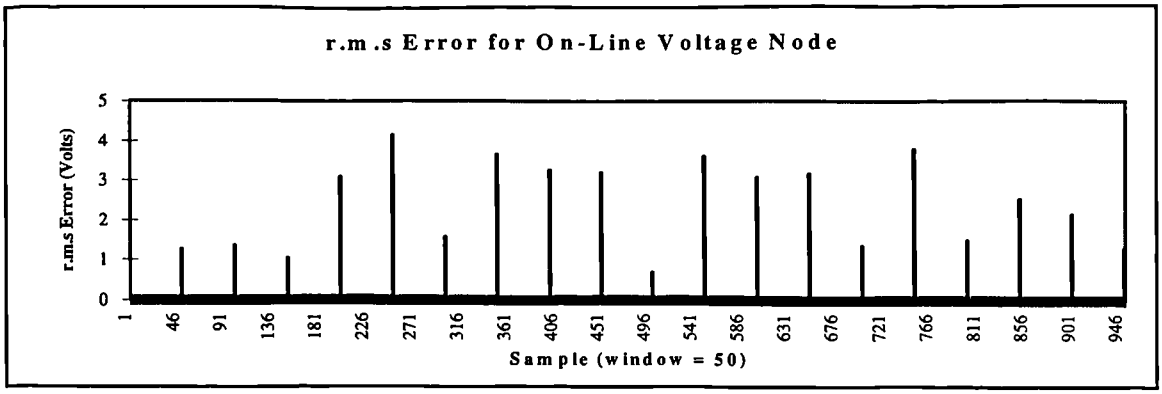


FIGURE 4.30
r.m.s Error On-Line Test (Volts)

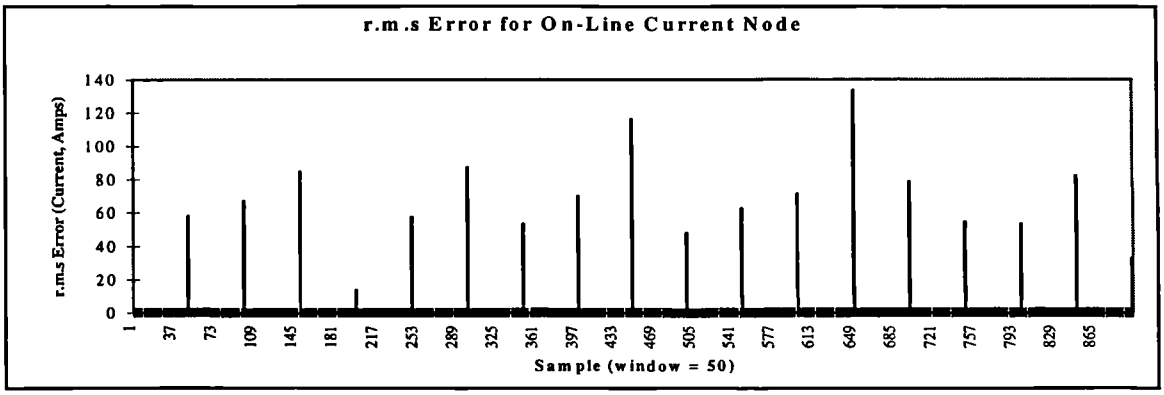


FIGURE 4.31
r.m.s Error On-Line Test (Current)

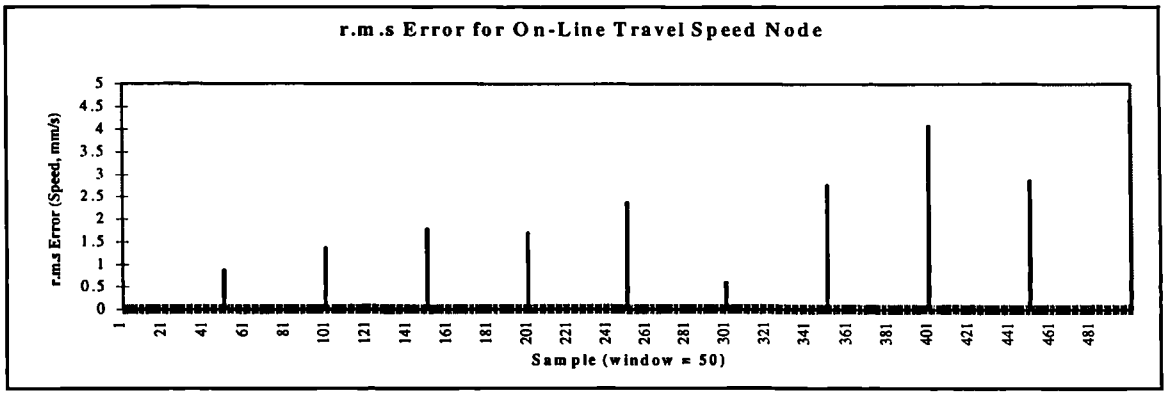


FIGURE 4.32
r.m.s Error On-Line Test (Speed)

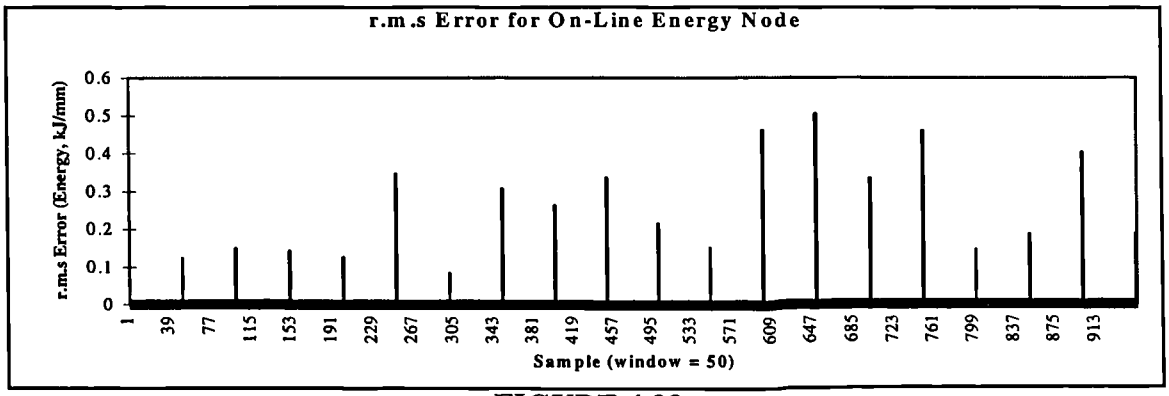


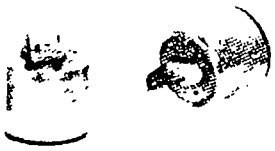
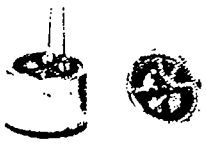
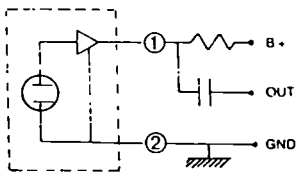
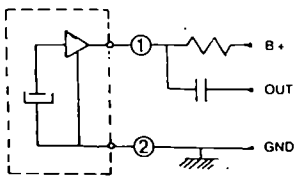
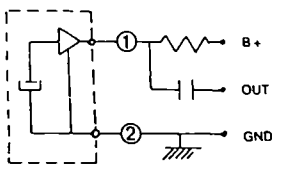
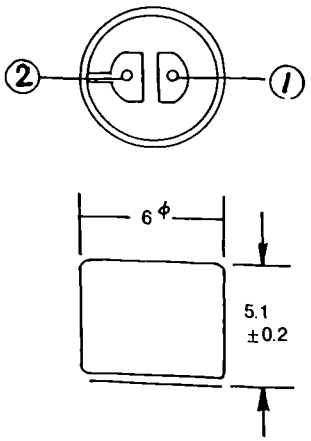
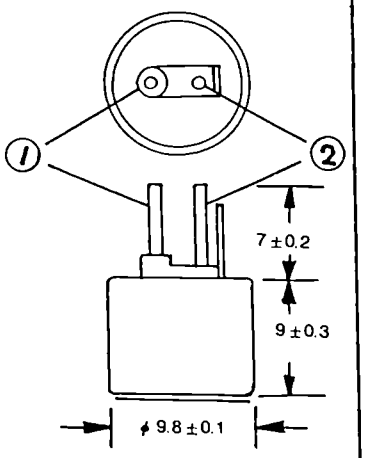
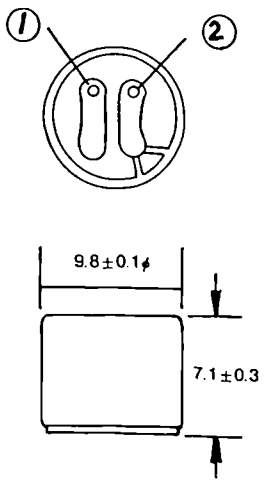
FIGURE 4.33
r.m.s Error On-Line Test (Energy)

9.0 APPENDICES

APPENDIX 1

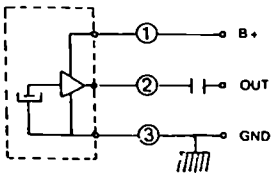
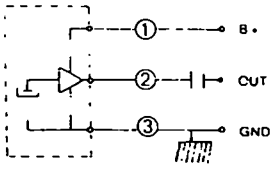
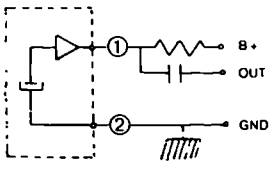
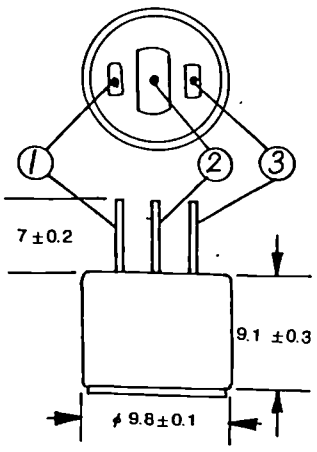
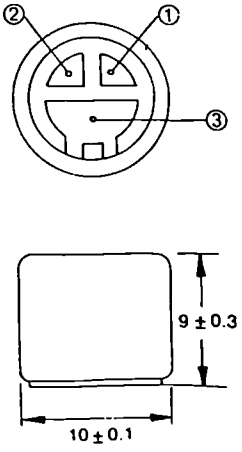
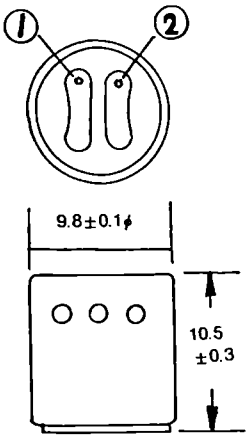
Transducer Specifications

ELECTRET CONDENSER MICROPHONE UNITS

ITEM NO.	E.C.M. 10	E.C.M. 30	E.C.M. 60
OUTLINES	 2-terminal	 2-pin	 2-pin or 2-terminal
DIRECTIONALITY	Omni Directional	Omni Directional	Omni Directional
SENSITIVITY (0db = 1V/1μbar. 1KHz) VCC=4.5V, RL=1KΩ	A: −56±3db B: −60±3db C: −64±3db D: −68±3db E: −72±3db	A: −55±3db B: −61±3db C: −67±3db	A: −56±3db B: −60±3db C: −64±3db D: −68±3db E: −72±3db
OUTPUT IMPEDANCE	Low	Low	Low
FREQUENCY	50Hz~13KHz	50Hz~13KHz	50Hz~13KHz
RANGE OF OPERATING VOLTAGE	1.5V to 10V Standard Voltage: 4.5V	1.5V to 10V Standard Voltage: 4.5V	1.5V to 10V Standard Voltage: 4.5V
CURRENT DRAIN	0.5mA Max.	0.6mA Max.	0.4mA Max.
S/N RATIO	40db or more	40db or more	40db or more
MAXIMUM INPUT SOUND PRESSURE	120db SPL	120db SPL	120db SPL
CIRCUIT DIAGRAM		 RL = 300Ω~5KΩ	 RL = 300Ω~5KΩ
DIMENSIONS (MM)			

MAIN USAGE OF ECM UNITS: Microphone, Cassette Recorder, Sonic Controlled Toy, Intercom System, Sonic Controlled Switch, Telephone and Disco Light, ETC.

ELECTRET CONDENSER MICROPHONE UNITS

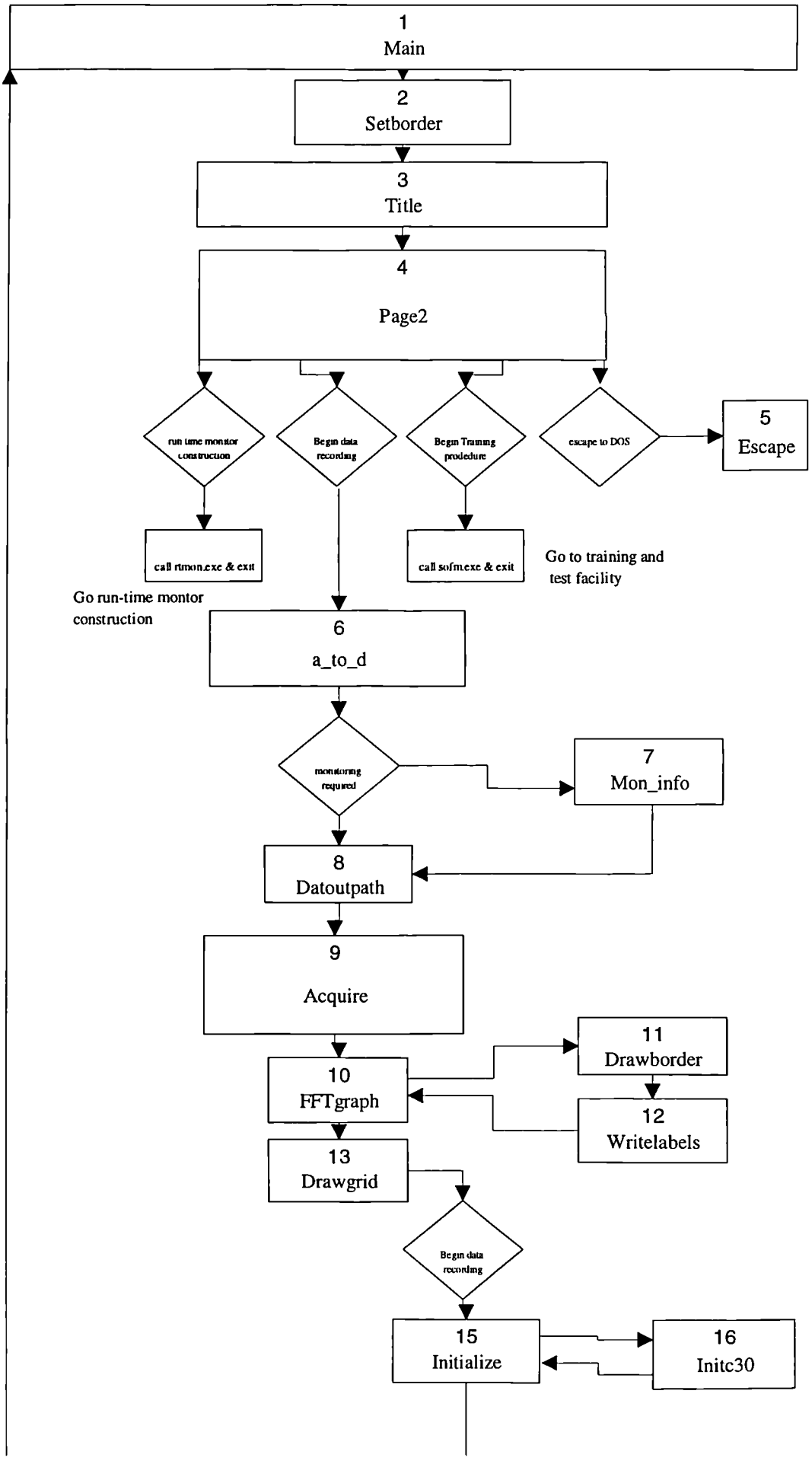
	ECM-65	ECM-66	UECM-3
OUTLINES	 3-pin	 3-terminal	 2-terminal
DIRECTIONALITY	Omni Directional	Omni Directional	UNI Directional
SENSITIVITY (0db = 1V/1μbar. 1KHz) VCC=4.5V, RL=1KΩ	A: −54±3db B: −60±3db C: −66±3db D: −72±3db	A: −54±3 db B: −60±3 db C: −66±3 db D: −72±3 db	A: −56±3db B: −60±3db C: −64±3db D: −68±3db E: −72±3db
OUTPUT IMPEDANCE	Low	Low	Low
FREQUENCY	50Hz~13KHz	50Hz~13KHz	50Hz~10KHz
RANGE OF OPERATING VOLTAGE	1.5V to 10V Standard Voltage: 4.5V	1.5V to 10V Standard Voltage: 4.5V	1.5V to 10V Standard Voltage: 4.5V
CURRENT DRAIN	0.6mA Max.	0.6mA Max.	0.6mA Max.
S/N RATIO	40db or more	40db or more	40db or more
MAXIMUM SOUND PRESSURE LEVEL	120db SPL	120db SPL	120db SPL
CIRCUIT DIAGRAM			
DIMENSIONS (MM)			

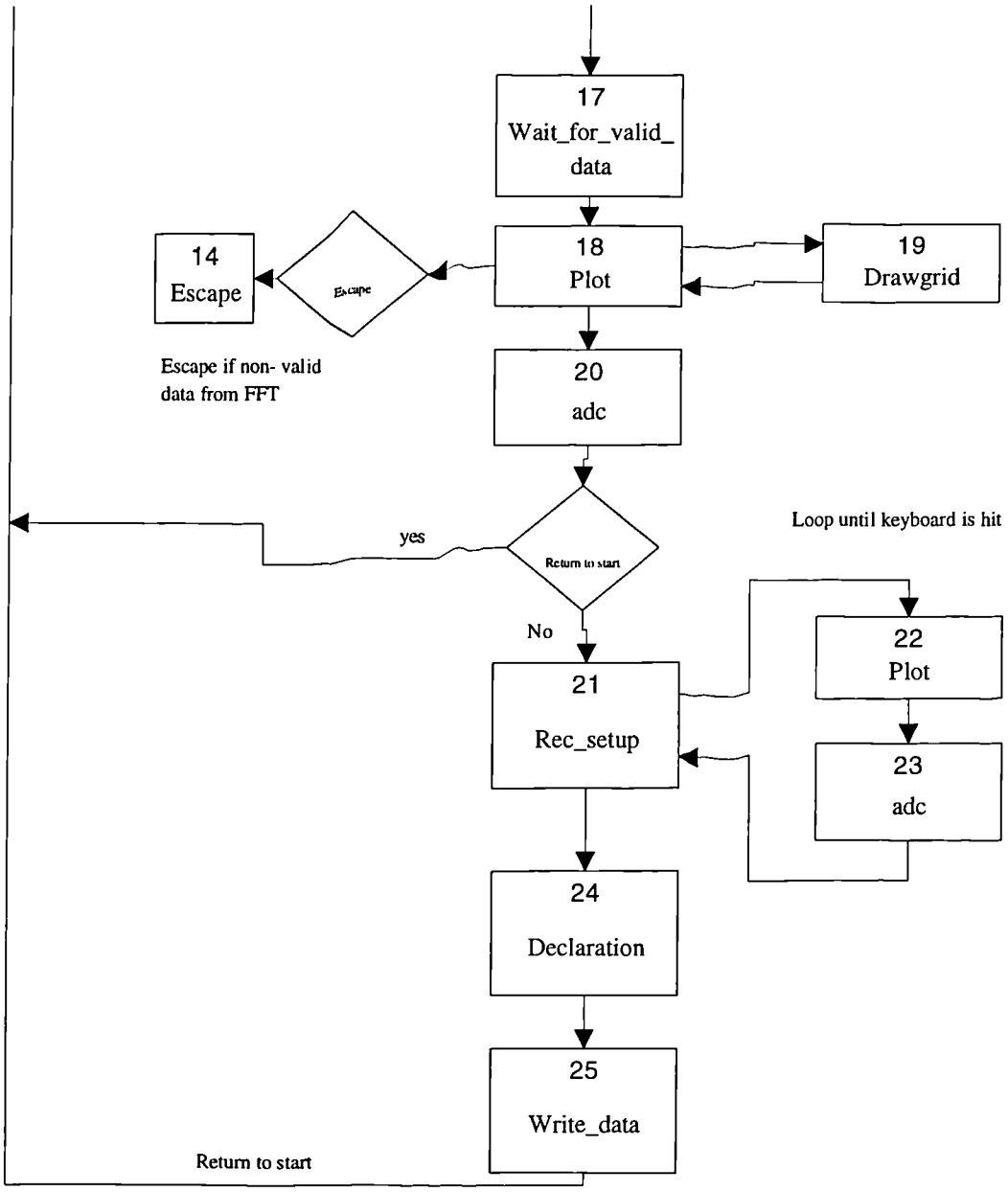
JIN IN ELECTRONICS CO., LTD.
TEL:(02)7716316•7520904 •7410450
FAX:886-2-7515631 P.O.BOX. 24-832 TAIPEI

APPENDIX 2

Source Code for TMS320C30 Data Acquisition

Flow Chart for Data Acquisition Software (NW_4.c)





FUNCTIONS KEY

2. void setborder(short color); Sets screen graphics boarder
3. void title(void); Draws title page
4. void page2(void); Gives information, record/ train /test decisions
5. void escape(void); Escapes program to DOS
6. void a_to_d(void); Takes information from user interface to initiate adc. process parameter logging
7. void mon_info(void); Takes information from user interface to initiate appropriate adc channels
8. void datoutpath(void); Sets up output path and file name for recorded data
9. void acquire(void); Sets up recording procedure screen and tests fft plots
10. void fftgraph(void); Sets up graphics for fft graph
11. void drawborder(void); Draws fft graph boarder and sets graphics window
12. void writelabels(void); Labels fft graph axes
13., 19. void drawgrid(void); Draws fft grid
15. void initialize(void); Initializes global variables , screen parameters and TMS320C30 board
16. void initc30(void); Initiates TMS320C30 board and downloads fft program
17. void wait_for_valid_data(void); Holds program untill valid data from TMS320C30 board is.. present
18., 22. void plot(void); Reads TMS320C30 board and plots fft graph
20., 23. void adc(void); Reads adc board scales data and loads in memory
21. void rec_setup(void); Initiates recording procedure
24. void declaration(void); Accepts or rejects acquired data
25. void write_data(void); Writes recorded data to output file

J. R. McCardle 1997

```
/*************************************************************************
*  NW_4.C     (NEURAL-WELD N.A.M) Neural Acoustic Monitor                  *
*                                                                         *
*  Dedicated real time FFT analyser and data acquisition system.          *
*                                                                         *
*  Compiled by J.R. McCardle Neural Applications Group, Brunel            *
*  University, Dept of Design.                                            *
*                                                                         *
*  FFT is of 7th order (128 point), 20 kHz samping rate on TMS320C30      *
*  DSP. Taking SAW acoustic emissions between 0Hz and 10kHz. 64 data      *
*  points are computed and saved as ascii data.                           *
*************************************************************************/


#include <graph.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <process.h>
#include <errno.h>
#include <share.h>
#include <math.h>
#include <tms30.h>          /* LSI C30  Interface Library. */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* TMS320C30 Board parameters:*/

#define BOARDADR        0x290       /* Default I/O Base address of board */

#define BOARD_BUFF_ADR  0x3e010     /* Data output buffer on C30 board */
#define VALID_DATA_FLAG 0x3e00a     /* Flag TRUE when valid data on C30 board */
#define TIMERVAL    0x3e00b         /* Loc of value put into SPOX timer */

#define ALPHA       0x3e00d         /* Location of filter (integration) */
                                    /* coefficient in C30's memory */
#define SCALE       0x3e00e   /* Location of screen scale factor    */
#define OFFSET      0x3e00f   /* Location of screen offset factor   */


#define BASEOFFSET  58.0      /* Starting offset, in dB, for plot */

/* Use intrinsic versions of outp and inp */
#pragma intrinsic( outp, inp )

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* BYTRONIC MPIBM6 I/O Board parameters:*/

#define BASEADDRESS 512             /* Base address for i/o board in trigem pc */
#define DATA    BASEADDRESS + 16    /* write address of adc */
#define BUSY    BASEADDRESS + 20    /* busy line address of adc */

#define VOLTS_CHAN  10              /* channels on single ended mode */
#define AMPS_CHAN   11
#define SPEED_CHAN  12

/* Use intrinsic versions of outp and inp */
#pragma intrinsic( outp, inp )

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Pretty colour bits:*/

#define BACKGNDCOLOR    0   /* black */
#define AXISCOLOR   3       /* cyan */
#define GRAPHCOLOR  4       /* red */
#define TITLECOLOR  14      /* yellow */
#define LABELCOLOR  15      /* white */
#define printat(x, y, z)    _settextposition(x, y);\
                _outtext(z);
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Standard defines:*/

#define TRUE        1
#define FALSE       0

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Screen graphics and labeling parameters:*/

#define DATALENGTH  64          /* Length of spectral data array.*/
#define YLENGTH     108         /* Axis (grid) lengths. */
#define YSPACING    YLENGTH/10
#define XLENGTH     512
#define XSPACING    512/8

#define YLABLIN 2   /* Locations (LINe and COLumn) of */
#define YLABCOL 3   /* X and Y labels */
#define XLABLIN 10
#define XLABCOL 8
#define YCENTER 240
#define XCENTER 320
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Function Prototypes:*/

void    title(void);            /* Draws title page */
void    page2(void);            /* Gives information, record/ train /test decisions */
```

```
void    acquire(void);              /* Sets up recording procedure screen and tests fft plots */
void    fftgraph(void);             /* Sets up graphics for fft graph */
void    initialize(void);           /* Initializes global variables , screen parameters and.. */
                                    /* TMS320C30 board */
void    initc30(void);              /* Initiates TMS320C30 board and downloads fft program */
void    drawgrid(void);             /* Draws fft grid */
void    plot(void);                 /* reads TMS320C30 board and plots fft graph */
void    a_to_d(void);               /* Takes information from user interface to initiate adc.. */
                                    /* process parameter logging */
void    mon_info(void);             /* Takes information from user interface to initiate.. */
                                    /* appropriate adc channels */
void    datoutpath(void);           /* Sets up output path and file name for recorded data */
void    rec_setup(void);            /* Initiates recording procedure */
void    write_data(void);           /* Writes recorded data to output file */
void    declaration(void);          /* Accepts or rejects acquired data */
void    adc(void);                  /* Reads adc board scales data and loads in memory */
void    drawborder(void);           /* Draws fft graph boarder and sets graphics window */
void    writelabels(void);          /* Labels fft graph axes */
void    escape(void);               /* Escapes program to DOS */
void    Enterkey(void);             /* Holds program until enter key hit */
void    wait_for_valid_data(void);/* Holds program untill valid data from TMS320C30 board is.. */
                                    /* present*/
void    setborder(short color);     /* Sets screen graphics boarder */
void    yesno(void);                /* Boolean switch */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  */

/*  Global Variables:*/

short   xorig,yorig;     /* Graph axis origins x & y */


long    timerval;        /* Holds hex val put into sample timer */
long    frameno;         /* Current FFT frame */
long    winsize;         /* No of FFT frames per batch */
long    w;               /* Winsize count flag */
float   alpha;           /* Sq. magnitude filter coeff */
float   scale;           /* Used by C30 to scale screen values  */
float   offset;          /* Used by C30 to offset screen values */
float   volts;           /* Actual float scale values of process parameters */
float   amps;
float   speed;


int volts_byte;     /* 8 bit descriptors for process parameters */
int amps_byte;
int speed_byte;
int frame;          /* FFT frame counter in block of winsize */

long    intdata[DATALENGTH];    /* Data arrays for FFT and recorded data */
long    data[DATALENGTH], prev[DATALENGTH];
float   out[200][DATALENGTH];   /* Max size of I/O data array */
float   voltages[200];          /* Max size for parameter readings */
float   currents[200];
float   speeds[200];

int recording;      /* Recording flag */
int recset;         /* Digital recording done flag */
int accept;         /* Recording accept flag */
int record_done;    /* Recording carried out flag */
int monitor;        /* Flag for on-line monitor display */
int label_file;     /* Flag to output labelled data file */
int cont;           /* Flag to provide continuous data file */
int p_file;         /* Flag to provide batch data file */
int ans;            /* Yes or no flag */
int tsp;            /* travel speed flag */
int cur;            /* current flag */
int vlt;            /* voltage flag */

float   av_vlt;          /* averaged values for process parameters */
float   av_cur;          /* using winsize as frequency */
float   av_tsp;

FILE    *outfile;        /* Pointer for recorded data */
FILE    *headfile;       /* Pointer to data header file */

/*********************************************************************/

void main(void)
{
_setvideomode(_MAXRESMODE);      /* Looks for maximum screen resolution */
_setbkcolor(0);                  /* Sets background colour to black */
setborder(11);                   /* Sets border colour */
title();



while(TRUE){
      page2();

      record_done = FALSE;
      accept = FALSE;
      recset = FALSE;              /*set flags*/
      recording = FALSE;

      while (accept == FALSE ){     /* Loop until accepted recording flag is set */
         _clearscreen(_GCLEARSCREEN);
         a_to_d();
         record_done = FALSE;       /* ensure flag is set for repeat loop */
         _clearscreen(_GCLEARSCREEN);
         acquire();
         if(record_done == TRUE && accept == FALSE){
        _clearscreen(_GCLEARSCREEN);
        declaration();
```

```
        }
    }

  } /* while true loop */

}   /* End of main(). */


/*************************************************************************/


void plot (void) {

long    * dataptr, * prevptr;
long    *intdataptr, temp;

int x;
int i;

_setcliprgn(xorig,yorig,xorig+XLENGTH,yorig+YLENGTH); /*  Set Graphics clip */

drawgrid();          /* Draw axis grid. */

RdBlkInt(BOARD_BUFF_ADR, DUAL, DATALENGTH, intdata);  /* Get data from board */

intdataptr = intdata;
dataptr = data;             /* Pointer to current data array. */

for (i = 0; i < DATALENGTH; i++) {
      temp = (*intdataptr++);
      if(recording){out[frameno][i] = (float)temp;}
      *dataptr++ = (long)temp;
      }


dataptr = data;         /* Pointer to current data array */
prevptr = prev;         /* Pointer to previous data array */


/* Plot spectral data: */

  {

  for(x = 0; x < DATALENGTH; x++)

  {  _setcolor(BACKGNDCOLOR);
     _rectangle(_GFILLINTERIOR,(x * 8) + 2, (short)YLENGTH - (short)*prevptr, (x * 8) + 6,
     (short)YLENGTH);


    *(prevptr++) = *dataptr;      /* Set prev data value = current value. */


    _setcolor(GRAPHCOLOR);
    _rectangle(_GFILLINTERIOR,(x * 8) + 2, (short)YLENGTH - (short)*(dataptr)++, (x * 8) + 6,
    (short)YLENGTH);

  }
  }

}     /* End of FFT graphics plot.*/

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Initialize global variables, screen parameters, and the C30 board.    */

void initialize(void)
{

initc30();          /* Initialize the C30 board by down-*/
                    /* loading and starting FFT.OUT .*/

timerval = 417;     /* Timer value (20KHz sampling).*/
PutInt(TIMERVAL,DUAL,timerval);     /* Enter timer value to C30 board. */
                                    /* (freq. range is 1/2 sample rate.)    */
alpha = (float)0.9;     /* Default sq mag filter coefficient.   */
scale = (float)23.0;    /* The screen displays 2.3db per pixel*/

offset = -((scale / 10.0) * (BASEOFFSET + 10 * log10( 1.0/(1.0-alpha) ) ))*2.30;

PutFloat(ALPHA, DUAL, alpha);    /* Update value on C30 board */

PutFloat(OFFSET, DUAL, offset);  /* Update value on C30 board */

} /* End of c30 initialisation */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


/* FFT graph set-up */

void fftgraph(void)
{
xorig = 60; yorig = 20; /* Origin of graph on screen.          */

drawborder();           /* Draws box around grid lines.     */
writelabels();          /* Write screen text. */

}  /* End of FFTgraph */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Draw Title Screen */
```

```
void title(void)
{

unsigned long a, b;

  /* Graphics page */

/* Draw concentric circles */
_setcolor(3);            /* cyan */
_ellipse(_GBORDER, XCENTER + 20, (YCENTER / 3) + 30, XCENTER - 20, (YCENTER / 3) + 70);
_ellipse(_GBORDER, XCENTER + 30, (YCENTER / 3) + 20, XCENTER - 30, (YCENTER / 3) + 80);
_ellipse(_GBORDER, XCENTER + 50, (YCENTER / 3), XCENTER - 50, (YCENTER / 3) + 100);
_ellipse(_GBORDER, XCENTER + 80, (YCENTER / 3) - 30, XCENTER - 80, (YCENTER / 3) + 130);

_registerfonts("TMSRB.FON");    /* Register Times Roman font */
_setfont("t'tmsrb' b");

_setcolor(GRAPHCOLOR);          /* text colour to red */
_moveto(200,60);
_outgtext("NEURAL-WELD");       /* Print title text */
_moveto(250,260);
_outgtext(" N . A . M ");

_settextcolor(GRAPHCOLOR);
printat(21,29, "Neural Acoustic Monitor");
printat(22,28, "For Submerged Arc Welding");

/* Draw graphics logo */

_settextcolor(TITLECOLOR);
printat(26,6, "Compiled by J.R. McCardle, Neural Application Group, Brunel University");

/* Draw welding rod! */
_setcolor(6);   /* Brown */
_rectangle(_GFILLINTERIOR, (XCENTER - 2), (YCENTER / 3) + 40, (XCENTER + 2), (YCENTER / 3) + 100);

/* Draw arc */
_setcolor(15);  /* White */
_ellipse(_GFILLINTERIOR, (XCENTER - 3), (YCENTER / 3) + 101,  (XCENTER + 3), (YCENTER / 3) + 150);

/* Draw base circle*/
_setcolor(7);   /* Grey */
_pie(_GFILLINTERIOR, XCENTER - 120, (YCENTER / 3) + 100, XCENTER + 120, (YCENTER / 3) + 201,
XCENTER + 118, (YCENTER / 3) + 201, XCENTER + 118, (YCENTER / 3) + 100);
_pie(_GFILLINTERIOR, XCENTER - 120, (YCENTER / 3) + 100, XCENTER + 120, (YCENTER / 3) + 201,
XCENTER - 118, (YCENTER / 3) + 100, XCENTER - 120, (YCENTER / 3) + 201);

/* delay to read title page */

for(a=0;a<10000000;a++){}

} /* End of title page and graphics logo */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  */

/* Construct information page */

void page2(void)
{


int input[3];
int ok = FALSE;


_clearscreen(_GCLEARSCREEN);        /* Resets screen */

{
_settextcolor(GRAPHCOLOR);
printat(2, 23, " INTRODUCTION TO NEURAL-WELD N.A.M");
printat(30, 23," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(5, 2, " NEURAL-WELD N.A.M is a dedicated acoustic analyser for Submerged Arc Welding.");
printat(6, 2, " The analysis is based upon the patterns created by a set 128 point FFT");
printat(7, 2, " performed by the TMS320C30 Digital Signal Processor.");
printat(9, 2, " The data generated is saved and used as training input vectors for a Kohonen");
printat(10, 2, " Self-Organising Feature Map, trained with SOFM.EXE.");
printat(11, 2, " Pre-trained networks can be loaded and executed to monitor on-line signals in");
printat(12, 2, " real-time using the run-time monitor (RTMON.EXE). Graphical representations");
printat(13, 2, " of the on-line FFT are given during the operation if requested.");
printat(15, 2, " Testing of pre-trained networks, off-line, is possible through SOFM.EXE ");
printat(16, 2, " using pre-recorded test data.");

_settextcolor(AXISCOLOR);
printat(19, 24, " (C)ONTINUE and begin recording.");
printat(21, 24, " (T)raining procedure.");
printat(23, 24, " (R)UN-TIME monitor construction.");
printat(25, 24, " (E)SCAPE to DOS.");


/* little red circles */
_setcolor(4);
_ellipse(_GFILLINTERIOR, 180, 292, 187, 299);
_ellipse(_GFILLINTERIOR, 180, 324, 187, 331);
_ellipse(_GFILLINTERIOR, 180, 356, 187, 363);
_ellipse(_GFILLINTERIOR, 180, 388, 187, 395);
}

/* Switch to Record mode -DOS -SOFM */

while(input[0] = getch())
```

```
{
    switch(input[0]) {
    case 'c':
    case 'C': _clearscreen(_GCLEARSCREEN);
          ok = TRUE;
          break;


    case 't':
    case 'T': _clearscreen(_GCLEARSCREEN);  /* Resets screen */
         execl("sofm_3.exe", NULL);         /* Call Kohonen training procedure */
         exit(1);

    case 'r':
    case 'R': _clearscreen(_GCLEARSCREEN);  /* Resets screen */
         execl("rtmon_4.exe", NULL);        /* Call run time monitor procedure */
         exit(1);

    case 'e':
    case 'E': _setvideomode(_DEFAULTMODE);
          _displaycursor(_GCURSORON);
          _settextposition(25,81);
          exit(0);
          break;

    }
    if (ok == TRUE) break;
}


}/* End of page2 () */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void acquire(void){

char mess[22];
int input[3];
int ok = FALSE;
char dat[12];

_settextcolor(4);
printat(18, 30, " RECORDING PROCEDURE ");

_settextcolor(14);
printat(20, 11, "1. Ensure you have a valid FFT response on the graph above.");
printat(21, 11, "2. Start the weld process then press the 'RECORD' key.");
printat(22, 11, "3. Wait for the 'END OF RECORD' audio-visual signal BEFORE");
printat(23, 11, "   stopping the weld.");

_settextcolor(AXISCOLOR);
printat(26,26, " ( )TART recording a file.");
printat(28,26, " ( )ETURN to start of program.");
_settextcolor(GRAPHCOLOR);
printat(26,28, "S");
printat(28,28, "R");

fftgraph();
drawgrid();


initialize();

wait_for_valid_data();

while(record_done == FALSE){    /* Loop until recording setup is null */

   plot();
   adc();

   if (kbhit())
       rec_setup();

   if(recording == TRUE){
      /* Recording message */
      _setvieworg(0, 0);
      _setcliprgn(0, 0, 640, 480); /* reset Graphics clip */
      _setcolor(0);
      _rectangle(_GFILLINTERIOR, 0, 260, 640, 480); /* clear lower screen */
      _setcolor(8);
      _rectangle(_GFILLINTERIOR, XCENTER - 150, YCENTER + 88, XCENTER + 170, YCENTER + 124);
      _setcolor(2);
      _rectangle(_GFILLINTERIOR, XCENTER - 160, YCENTER + 78, XCENTER + 160, YCENTER + 114);
      _setcolor(10);
      _rectangle(_GBORDER, XCENTER - 160, YCENTER + 78, XCENTER + 160, YCENTER + 114);
      _settextcolor(15);
      sprintf(mess, " RECORDING IN PROGRESS.");
      _settextposition(21,30);
      _outtext(mess);
      sprintf(mess, " PRESS ANY KEY TO HALT.");
      _settextposition(22,30);
      _outtext(mess);

      frame = 0; /* zero FFT block frame count */

     do{    /*while recording is true */
          for( frameno = 0; frameno < winsize; frameno++) { /* This is the data window loop */
         plot();
         adc();
         }  /* End of winsize loop */

          write_data();
```

```
          frame ++; /* increment block frame counter */

          _settextcolor(TITLECOLOR);
          _settextposition(25,30);
          sprintf(dat, "FFT frames = %d x %d", frame, winsize);
          _outtext(dat);

          if( kbhit() ){
         recording = FALSE;
         record_done = TRUE; /* recording has been done */
           }

          if(p_file == TRUE){
         recording = FALSE;
         record_done = TRUE; /* recording has been done */
           }

      }while (recording == TRUE);

      getch();

      frame *= winsize; /* calculates actual No of FFT frames gathered */

      /* Give end of record signal */
      _setvieworg(0, 0);
      _setcliprgn(0, 0, 640, 480); /* reset Graphics clip */
      _setcolor(8);
      _rectangle(_GFILLINTERIOR, XCENTER - 150, YCENTER + 88, XCENTER + 170, YCENTER + 124);
      _setcolor(4);
      _rectangle(_GFILLINTERIOR, XCENTER - 160, YCENTER + 78, XCENTER + 160, YCENTER + 114);
      _setcolor(12);
      _rectangle(_GBORDER, XCENTER - 160, YCENTER + 78, XCENTER + 160, YCENTER + 114);
      printf("\a");   /* Audio signal */
      _settextcolor(15);
      printat(21, 32, " RECORDING HALTED.");
      _settextcolor(14);
      _settextposition(25,30);
      sprintf(dat, "%d FFT frames stored", frame);
      _outtext(dat);
      printat(27, 32, "Enter to Continue");
      Enterkey();


  } /* end of recording loop ie) if recording == TRUE */


} /* this is record_done loop */

}    /* End of acquire  */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


void a_to_d (void){

int ok = FALSE;
char mess[12];
int input[1];

_settextcolor(GRAPHCOLOR);
printat(2, 31, "ADC & FILE FORMAT");
printat(30, 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(4, 10, " Please enter file format, recording & ADC information");

_settextcolor(3);
printat(7, 5, " TIME WINDOW SIZE FOR DATA FILE (FFT FRAMES, 200Max)...");
printat(9, 5, " FFT WINDOW PER FILE OR CONTINUOUS DATA (F/C).....");
printat(11, 5, " DATA FILE LABELLING REQUIRED? (Y/N).....");


/* Little red circles */
_setcolor(4);
_ellipse(_GFILLINTERIOR, 20, 100, 27, 107);
_ellipse(_GFILLINTERIOR, 20, 132, 27, 139);
_ellipse(_GFILLINTERIOR, 20, 164, 27, 171);


/* Ask for window size */
do{

_settextcolor(GRAPHCOLOR);
printat(7, 62, "                ");
_settextposition(7, 60);
_outtext("\x10");           /* ASCII delta to right character */
_settextposition(7, 62);
 scanf("%ld", &winsize);
    if(winsize > 200){
    printat(7, 62, "Sorry! Max 200. ");
    printf("\a");
    Enterkey();
    }
}while (winsize > 200);

sprintf(mess,"%ld", winsize);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(7, 60, " ");        /* Erase delta cursor */


/* Ask for per file or continuous data  */
```

```
_settextcolor(GRAPHCOLOR);
_settextposition(9, 60);
_outtext("\x10");           /* ASCII delta to right character */


    while(input[0] = getch())
       {

        switch(input[0]) {
           case 'f':
           case 'F':
                _settextcolor(TITLECOLOR);
                _outtext("Per File");              /* ouput value to screen */
                p_file = TRUE;
                ok = TRUE;
                break;

           case 'c':
           case 'C':
                _settextcolor(TITLECOLOR);
                _outtext("Continuous");            /* ouput value to screen */
                cont = TRUE;
                ok = TRUE;
                break;
        }
     if( ok == TRUE) break;
    }
      printat(9, 60, " ");  /* Erase delta cursor */

           /*--------------------------------------*/

/* Ask if data file labelling required (normally for recall testing only) */

_settextcolor(GRAPHCOLOR);
_settextposition(11, 60);
_outtext("\x10");
_settextposition(11, 62);
yesno();

label_file = ans;

_settextcolor(TITLECOLOR);
_settextposition(11, 62);
   if(ans) {
    _outtext("YES");
    mon_info();
    }
   else {
    _outtext("NO");
    }
printat(11, 60, " ");     /* Erase delta cursor */

datoutpath();

_settextcolor(GRAPHCOLOR);
printat(28, 17, " Options accepted. Press 'Enter' to continue...");

Enterkey();

} /* end of a_to_d() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void mon_info (void){

_setcolor(8);
_rectangle(_GFILLINTERIOR, XCENTER - 150, YCENTER - 40, XCENTER + 170, YCENTER + 60);
_setcolor(0);
_rectangle(_GFILLINTERIOR, XCENTER - 160, YCENTER - 50, XCENTER + 160, YCENTER + 50);
_setcolor(3);
_rectangle(_GBORDER, XCENTER - 160, YCENTER - 50, XCENTER + 160, YCENTER + 50);

_settextcolor(4);
printat(14, 24, "WHICH PROCESS VARIABLES FOR LABELS?");
_settextcolor(3);
printat(16, 24, "VOLTAGE (Y/N).......");
printat(17, 24, "CURRENT (Y/N).......");
printat(18, 24, "TRAVEL SPEED (Y/N)..");

/* Ask if voltage*/

_settextcolor(GRAPHCOLOR);
_settextposition(16, 44);
_outtext("\x10");
_settextposition(16, 46);
yesno();

vlt = ans;

_settextcolor(TITLECOLOR);
_settextposition(16, 46);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }
printat(16, 44, " ");     /* Erase delta cursor */

/* Ask if current */

_settextcolor(GRAPHCOLOR);
_settextposition(17, 44);
_outtext("\x10");
```

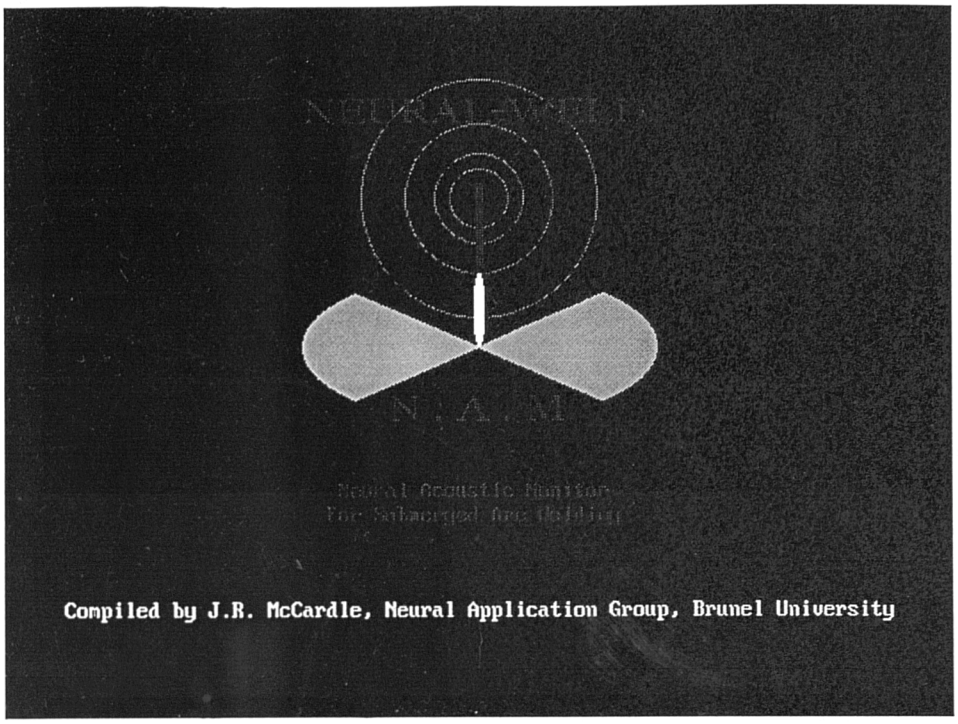
```
_settextposition(17, 46);
yesno();

cur = ans;

_settextcolor(TITLECOLOR);
_settextposition(17, 46);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }
printat(17, 44, " ");     /* Erase delta cursor */

/* Ask if travel speed*/

_settextcolor(GRAPHCOLOR);
_settextposition(18, 44);
_outtext("\x10");
_settextposition(18, 46);
yesno();

tsp = ans;

_settextcolor(TITLECOLOR);
_settextposition(18, 46);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }
printat(18, 44, " ");     /* Erase delta cursor */

Enterkey();

/* Erase info box */
_setcolor(0);
_rectangle(_GFILLINTERIOR, XCENTER - 160, YCENTER - 50,XCENTER + 170, YCENTER + 60);

} /* end of mon_info */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void datoutpath (void) {

char  outpath[_MAX_PATH];
char drive[_MAX_DRIVE], dir[_MAX_DIR];
char fname[_MAX_FNAME], ext[_MAX_EXT];
char mess[45];

do {

/* Ask for drive and file declaration */
_settextcolor(4);
printat(18, 26, " DRIVE AND FILE DECLARATION ");
_settextcolor(3);
printat(20, 28, " Give path to store data.");
printat(21, 23, " An extention of .NWD is automatic.");

printat(22, 18, " Enter Output drive, directory and file name. ");
printat(23, 17, " Default is the given name in home directory :-");

_settextposition(25, 32);  /* Display cursor to prompt file output path */
_settextcolor(GRAPHCOLOR);
_outtext("\x10");

flushall();
_settextposition(25, 34);
gets( outpath );

_splitpath( outpath, drive, dir, fname, ext);   /* create output path */
strcpy ( ext, "nwd" );
_makepath( outpath, drive, dir, fname, ext );

if( (outfile = _fsopen( outpath, "wb", SH_DENYWR )) == NULL ){  /* error message */
    printf("\a");
    sprintf( mess, " Unable to open,(err: %d)", errno);
    _outtext(mess);
    Enterkey();
    printat(25,34, "                          "); /*erase error message */
}

}while( outfile == NULL );  /* Loop declaration until valid */

_settextcolor(TITLECOLOR);
_outtext(outpath);
printat(25,32, " "); /* erase delta cursor */

}  /* End of Datoutpath */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void rec_setup(void){

int ok = FALSE;
int rec[1];


while(rec[0] = getch()){
    switch(rec[0]){
```

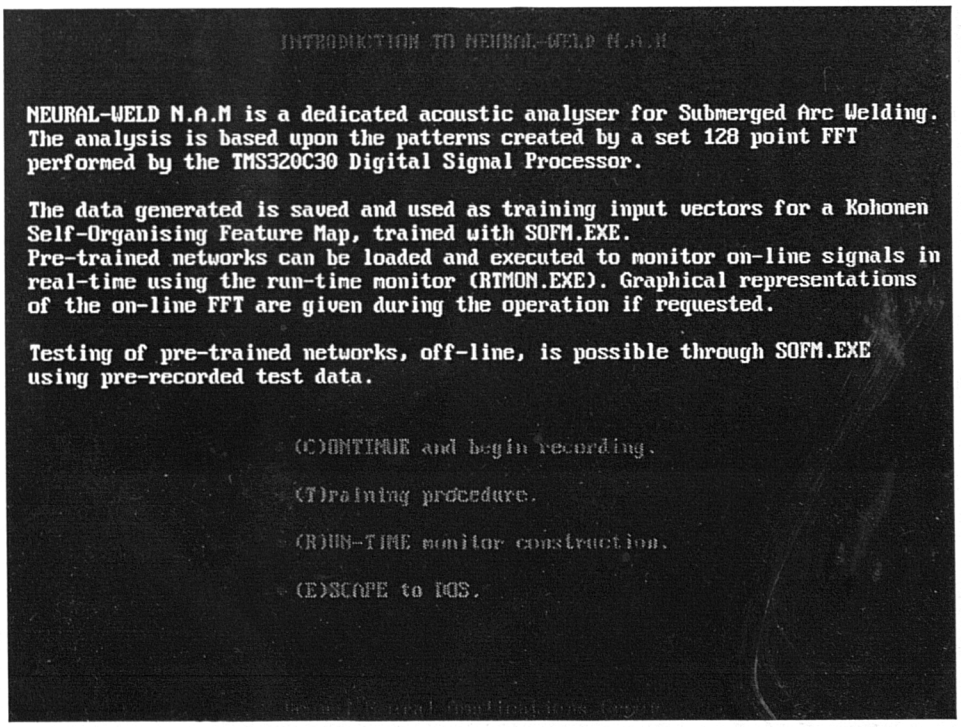
```
    case 'r':   /* Return to start of program */
    case 'R':
        recording = FALSE;
        recset = TRUE;
        accept = TRUE;
        record_done = TRUE;
        ok = TRUE;
        _setvieworg(0, 0);
        _setcliprgn(0, 0, 640, 480);/* reset Graphics clip */
        fclose(outfile);
        break;


    case 's':   /* Start record Key*/
    case 'S':
         recording = TRUE;
         ok = TRUE;
         break;
    } /*end of switch */

if(ok == TRUE) break;

}

} /* end of rec_setup() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void declaration(void){

int ok = FALSE;
char    dat[12];
int input[3];


_settextcolor(4);
printat(18, 30, " ACCEPTANCE DECLARATION ");
_settextcolor(3);
printat(20, 11, "  If you are NOT happy with the recording you can repeat it.");
printat(21, 11, "  Re-typing the same file name WILL overwrite the existing ");
printat(22, 11, "                                file.");

_settextcolor(AXISCOLOR);
printat(26,26, " ( )CCEPT data file/s and return to start.");
printat(28,26, " ( )EPEAT or record another file.");
_settextcolor(GRAPHCOLOR);
printat(26,28, "A");
printat(28,28, "R");


while(input[0] = getch())
{

    switch(input[0]) {
    case 'a':
    case 'A':
          fclose(outfile);
          ok = TRUE;
          accept = TRUE;    /* Set accept recording flag */
          break;

    case 'r':
    case 'R':
          ok = TRUE;
          accept = FALSE;
          fclose(outfile);

         }
    if (ok == TRUE) break;
}


} /* end of declaration() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void write_data(void){

float v;
int i;

for(w = 0; w < winsize; w++) {
    for(i = 0; i < 64; i++) {
    if( out[w][i] < 0) {      /* clip output data 0 -100 */
        out[w][i] = 0;
        }
    if( out[w][i] > 100) {
        out[w][i] = 100;
        }
    out[w][i] /= 100;
     }
    }

/* transfer data to named file */
for(w = 0; w < winsize; w++) {
    for(i = 0; i < 64; i++) {
    fprintf(outfile, "%f,", out[w][i]);
    }
    fprintf(outfile, "\n");
}

/* calculate average process parameters for data labels */
```

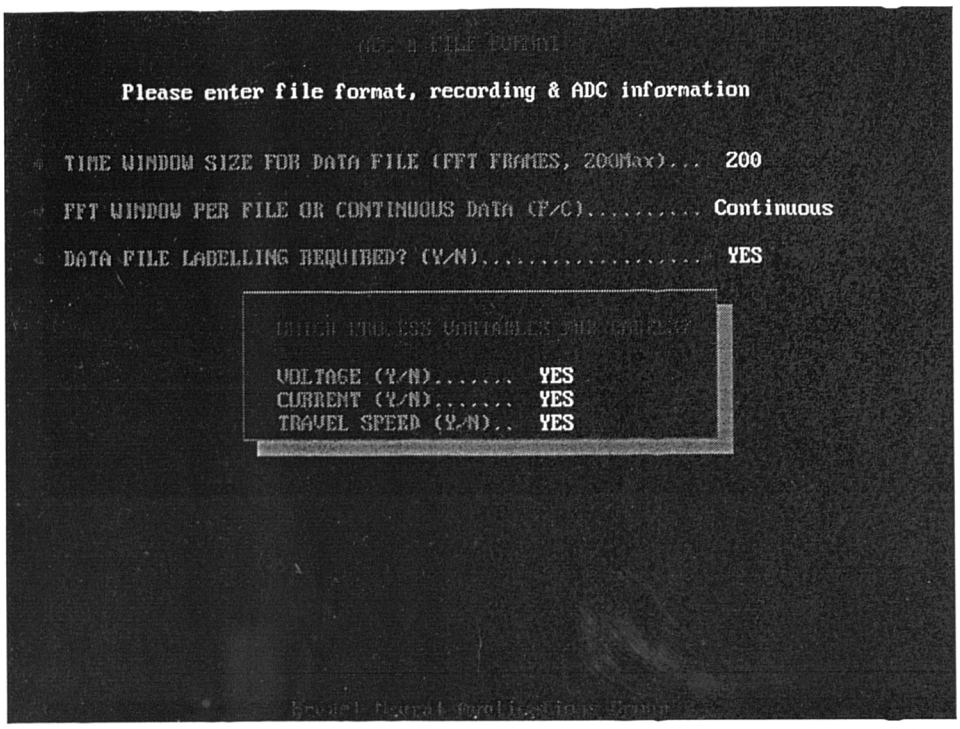
```
if(vlt){
v = 0.0F;
for(w = 0; w < winsize; w++) {v += voltages[w];}
av_vlt = v / winsize;
fprintf(outfile, "%.2f,", av_vlt);
}

if(cur){
v = 0.0F;
for(w = 0; w < winsize; w++) {v += currents[w];}
av_cur = v / winsize;
fprintf(outfile, "%.2f,", av_cur);
}

if(tsp){
v = 0.0F;
for(w = 0; w < winsize; w++) {v += speeds[w];}
av_tsp = v / winsize;
fprintf(outfile, "%.2f,\n", av_tsp);
}

} /* end of write_data */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void adc (void){

float scale_volts = 0.465F;  /* 0-50v scale range */
float scale_amps = 12.0F;    /* scale range offset */
float scale_speed = 0.4F;    /* 0-28mm\s scale range over 2v max range*/


_setvieworg(0, 0);

_settextcolor(GRAPHCOLOR);
printat(13,13, "VOLTAGE");
printat(13,37, "CURRENT");
printat(13,60, "TRAVEL SPEED");


outp(DATA, VOLTS_CHAN);

while((inp(BUSY) & 1) == 1);

volts_byte = inp(DATA);
volts =  ((float)volts_byte * scale_volts) + 0.6;
if(volts < 2){volts = 0.0;}
_settextposition(14, 13);
printf("        ");
_settextposition(14, 13);
printf("%.2f V", volts);

if(recording && vlt){voltages[frameno] = volts;}

          /* - - - - - - - - - - - - - - - - - - - */

outp(DATA, AMPS_CHAN);

while((inp(BUSY) & 1) == 1);

amps_byte = inp(DATA);
amps =  ((float)amps_byte - scale_amps);
amps /= 0.038F;                          /* calibrated linear scale factor y=mx+c (m=0.038) */
_settextposition(14, 37);
printf("          ");
_settextposition(14, 37);
printf("%.2f A", amps);

if(recording && cur){currents[frameno] = amps;}

        /* - - - - - - - - - - - - - - - - - - - */

outp(DATA, SPEED_CHAN);

while((inp(BUSY) & 1) == 1);

speed_byte = inp(DATA);
speed = ((float)speed_byte * scale_speed) + 3.5;
if(speed < 4.2){speed = 0.0;}
_settextposition(14, 62);
printf("            ");
_settextposition(14, 62);
printf("%.2f mm\\s", speed);

if(recording && tsp){speeds[frameno] = speed;}

} /* end of adc */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void initc30(void)
{
    SelectBoard(BOARDADR);          /* Set I/O base address: */

    Hold();                         /* Hold processor */
    LoadObjectFile("fft.out");      /* Download C30 program */
    Reset();                        /* Reset & run C30 code */

    *(((int *)VALID_DATA_FLAG)) = 0;

}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

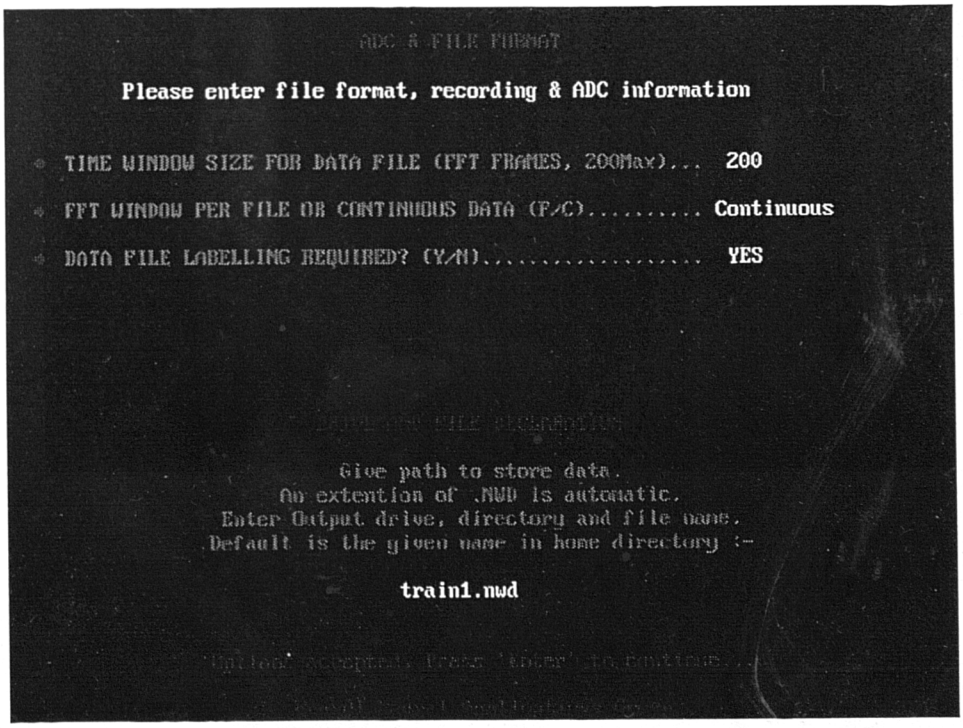
```
void wait_for_valid_data(void)
{
    while (GetInt(VALID_DATA_FLAG, DUAL) == 0);
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void drawborder(void)
{

/* Draws border around axis grid -- one pixel outside "viewport" that */
/* will be set up to clip graphics area. */

    _setcolor(AXISCOLOR);
    _moveto(xorig - 1, yorig - 1);                  /* Top line */
    _lineto(xorig + XLENGTH + 1, yorig - 1);

    _moveto(xorig - 1, yorig + YLENGTH + 1 );
    _lineto(xorig + XLENGTH + 1, yorig + YLENGTH + 1);  /* Bottom line */

    _moveto(xorig - 1, yorig - 1);
    _lineto(xorig - 1, yorig + YLENGTH + 1);         /* Left side */

    _moveto(xorig + XLENGTH + 1, yorig - 1);
    _lineto(xorig + XLENGTH + 1, yorig + YLENGTH + 1);  /* Right side */

} /* End of drawborder(). */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void drawgrid(void) /* Draws vertical and horizantal graph lines */
{
    int x,y;
    _setcolor(AXISCOLOR);   /* Axis colour */

_setvieworg(xorig, yorig);  /* set origin to suit clip region */

/* All graphics coordinates within graphics clipping region */

/* Draw horizontal graph lines.*/

    for(x=0,y=YLENGTH - YSPACING; y > 0 ;y -=YSPACING) {
        _moveto(x,y);_lineto(XLENGTH,y);
        }


/* Draw vertical graph lines.*/

    for(x=XLENGTH-XSPACING ,y=0; x > 0; x -=XSPACING) {
        _moveto(x,y);_lineto(x,YLENGTH);
        }
}       /* End of drawgrid() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void writelabels(void)
{

/* Write axis labels etc */

_settextcolor(LABELCOLOR);

/* X Axis labeling:*/

printat(XLABLIN, XLABCOL, "0               2.5              5               7.5              10Khz");
printat(XLABLIN+1, XLABCOL, "                        F R E Q U E N C Y                             ");


/* Y Axis labeling: */


printat (YLABLIN+3, YLABCOL, " M ");
printat (YLABLIN+4, YLABCOL, " A ");
printat (YLABLIN+5, YLABCOL, " G ");


/* Graphics titles */

_settextcolor(TITLECOLOR);
printat(1, 20, "ON-LINE FAST FOURIER TRANSFORM (7th ORDER)");

} /* End of writelabels() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Wait for an enter key press */

void Enterkey(void)
 {

int  input[3];
int  ok;

ok = 0;

   while(input[0] = getch()) {     /*  Do nothing until user hits enter     */
   switch(input[0]) {
    case '\x0d' : ok = 1;
          break;
            }
  if (ok == 1) break;

 }
```

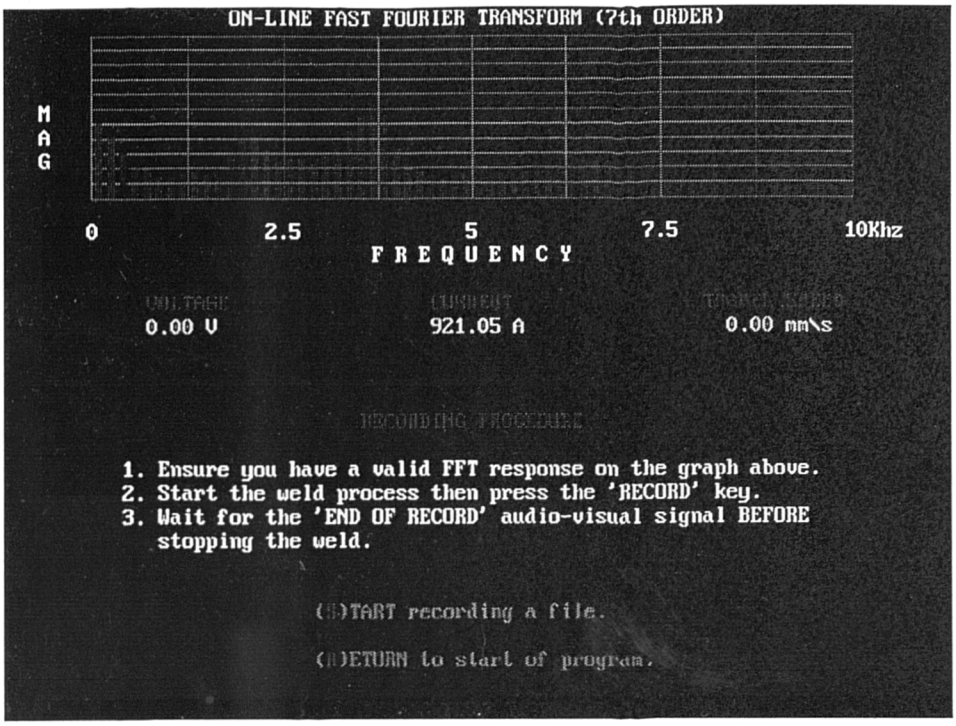
```
} /*   End Enterkey() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

  /* Yes or No switch */

void yesno(void) {

int ok;
int input[1];

ok = FALSE;
    while(input[0] = getch())
       {

        switch(input[0]) {
           case 'y':
           case 'Y':ans = TRUE;
                ok = TRUE;
                break;

           case 'n':
           case 'N':ans = FALSE;
                ok = TRUE;
                break;
         }
     if( ok == TRUE) break;
    }

} /* end of yesno */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Escape program to DOS */

void escape(void)
{
int esc[1];
esc[0] = getch();

switch(esc[0])
    {
            /* Escape Key */
    case '\x1b':     _setvideomode(_DEFAULTMODE);
            _displaycursor(_GCURSORON);
            _settextposition(25,81);
            exit(0);
            break;


    default:    break;

    }  /* End of switch */

}       /* End of escape(). */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Set PC overscan register */

void setborder(short colour)
{
    union REGS regs;

    regs.x.ax = 0x01001;     /* Overscan register */
    regs.h.bh = colour;
    int86(0x10, &regs, &regs);
}

/********************************* THE END ****************************/
```

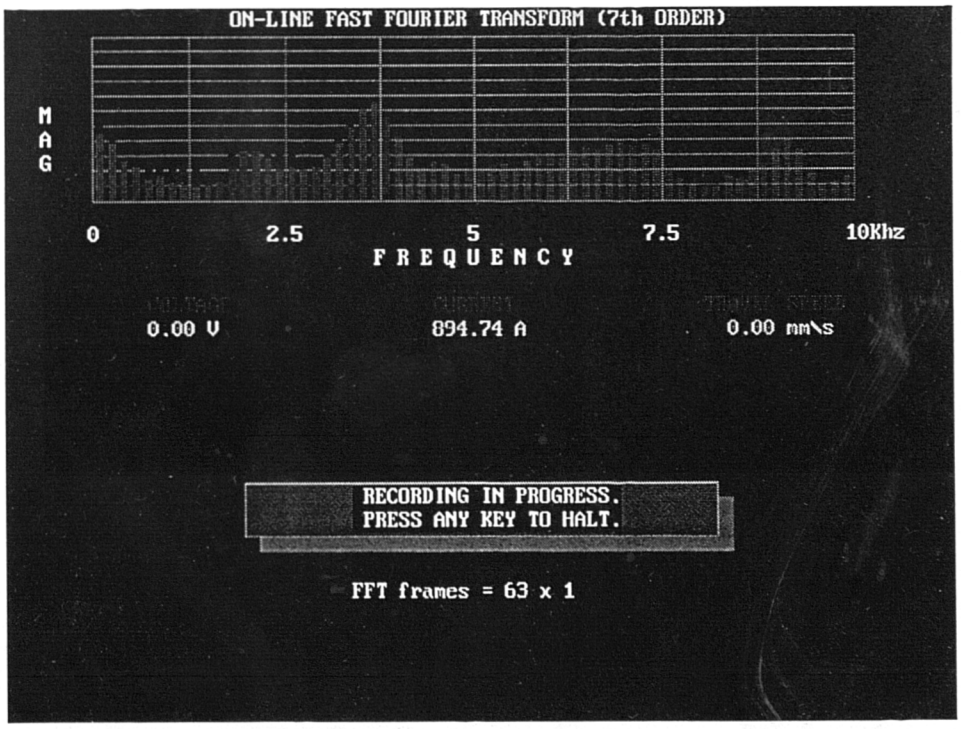

N.A.M
Compiled by J.R. McCardle, Neural Application Group, Brunel University

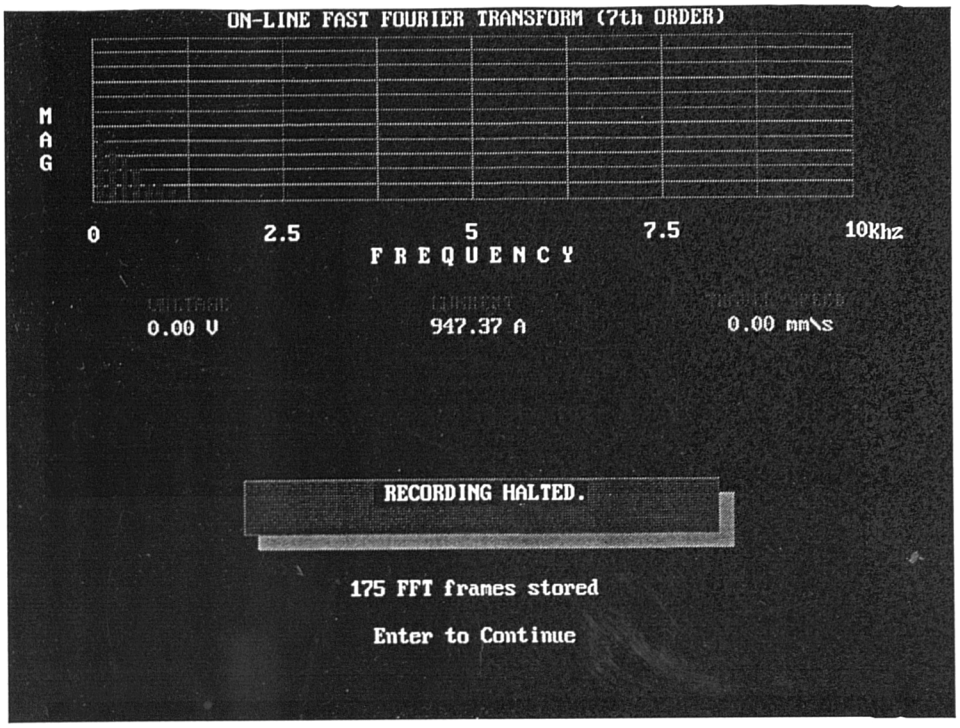

INTRODUCTION TO NEURAL-WELD N.A.M
NEURAL-WELD N.A.M is a dedicated acoustic analyser for Submerged Arc Welding.
The analysis is based upon the patterns created by a set 128 point FFT
performed by the TMS320C30 Digital Signal Processor.
The data generated is saved and used as training input vectors for a Kohonen
Self-Organising Feature Map, trained with SOFM.EXE.
Pre-trained networks can be loaded and executed to monitor on-line signals in
real-time using the run-time monitor (RTMON.EXE). Graphical representations
of the on-line FFT are given during the operation if requested.
Testing of pre-trained networks, off-line, is possible through SOFM.EXE
using pre-recorded test data.
(C)ONTINUE and begin recording.
(T)raining procedure.
(R)UN-TIME monitor construction.
(E)SCAPE to DOS.

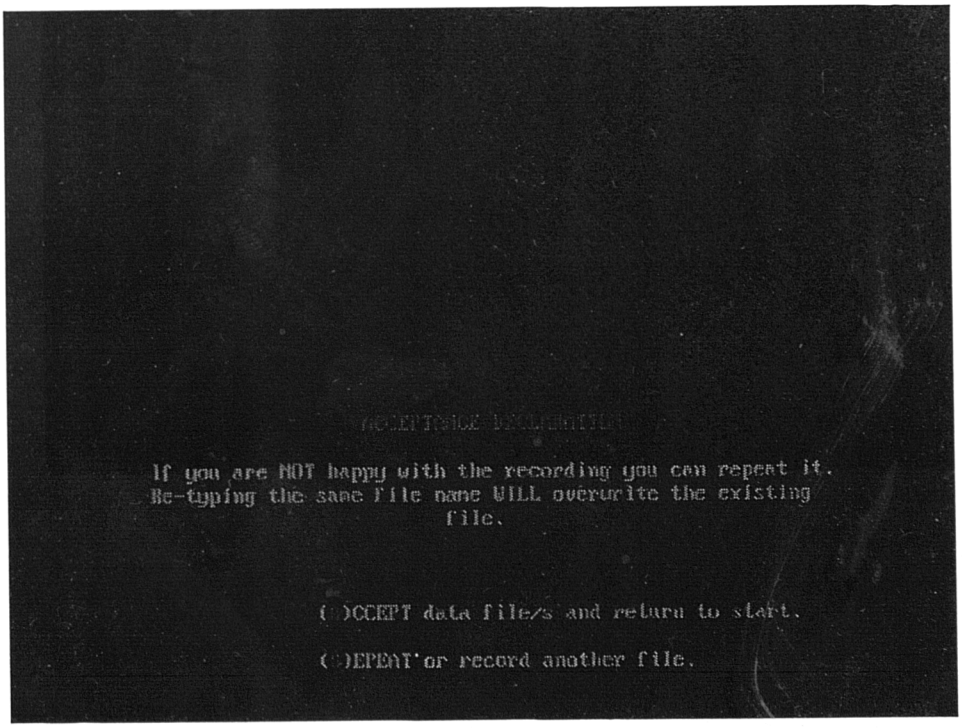

Please enter file format, recording & ADC information
TIME WINDOW SIZE FOR DATA FILE (FFT FRAMES, 200Max)... 200
FFT WINDOW PER FILE OR CONTINUOUS DATA (F/C)..... Continuous
DATA FILE LABELLING REQUIRED? (Y/N)..... YES
VOLTAGE (Y/N)..... YES
CURRENT (Y/N)..... YES
TRAVEL SPEED (Y/N).. YES


Please enter file format, recording & ADC information
TIME WINDOW SIZE FOR DATA FILE (FFT FRAMES, 200Max)... 200
FFT WINDOW PER FILE OR CONTINUOUS DATA (F/C)..... Continuous
DATA FILE LABELLING REQUIRED? (Y/N)..... YES
Give path to store data.
An extention of .NWD is automatic.
Enter Output drive, directory and file name.
Default is the given name in home directory :-
train1.nwd

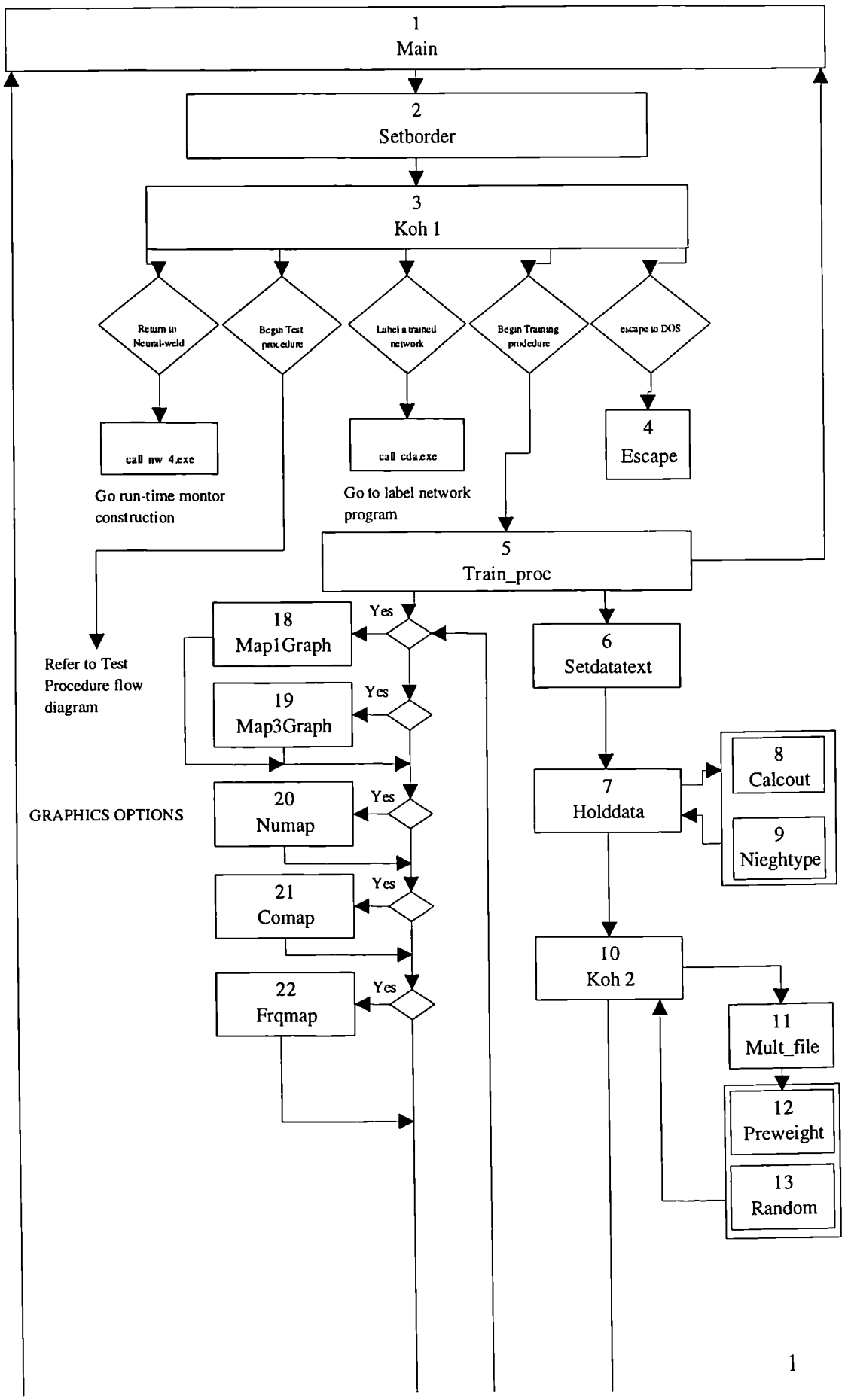

ON-LINE FAST FOURIER TRANSFORM (7th ORDER)
M
A
G
0
2.5
5
7.5
10Khz
F R E Q U E N C Y
0.00 V
921.05 A
0.00 mm\s
1. Ensure you have a valid FFT response on the graph above.
2. Start the weld process then press the 'RECORD' key.
3. Wait for the 'END OF RECORD' audio-visual signal BEFORE
stopping the weld.
(S)TART recording a file.
(R)ETURN to start of program.

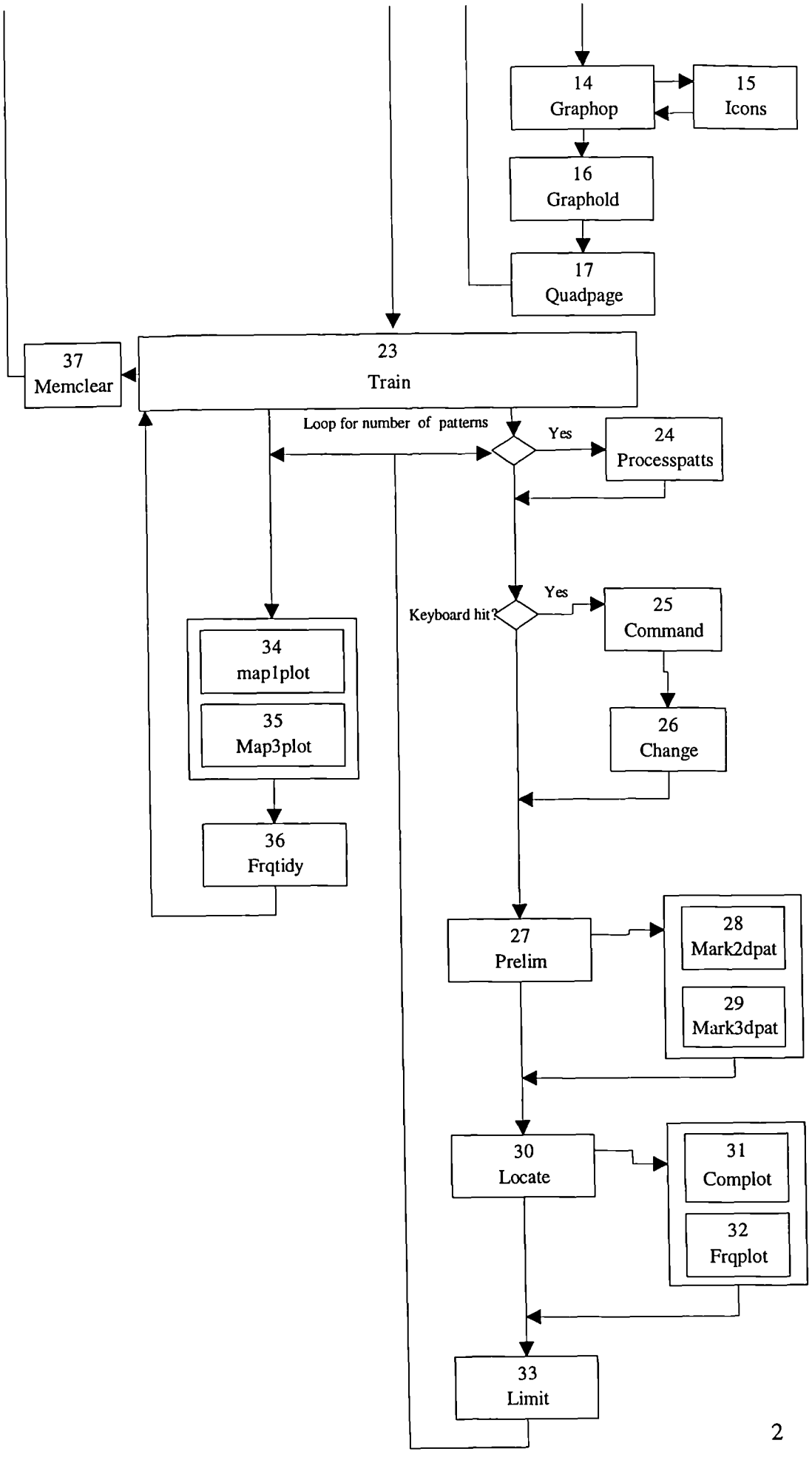

ON-LINE FAST FOURIER TRANSFORM (7th ORDER)
M
A
G
0
2.5
5
7.5
10Khz
F R E Q U E N C Y
0.00 V
894.74 A
0.00 mm\s
RECORDING IN PROGRESS.
PRESS ANY KEY TO HALT.
FFT frames = 63 x 1


ON-LINE FAST FOURIER TRANSFORM (7th ORDER)
MAG
0
2.5
5
7.5
10Khz
FREQUENCY
0.00 V
947.37 A
0.00 mm\s
RECORDING HALTED.
175 FFT frames stored
Enter to Continue

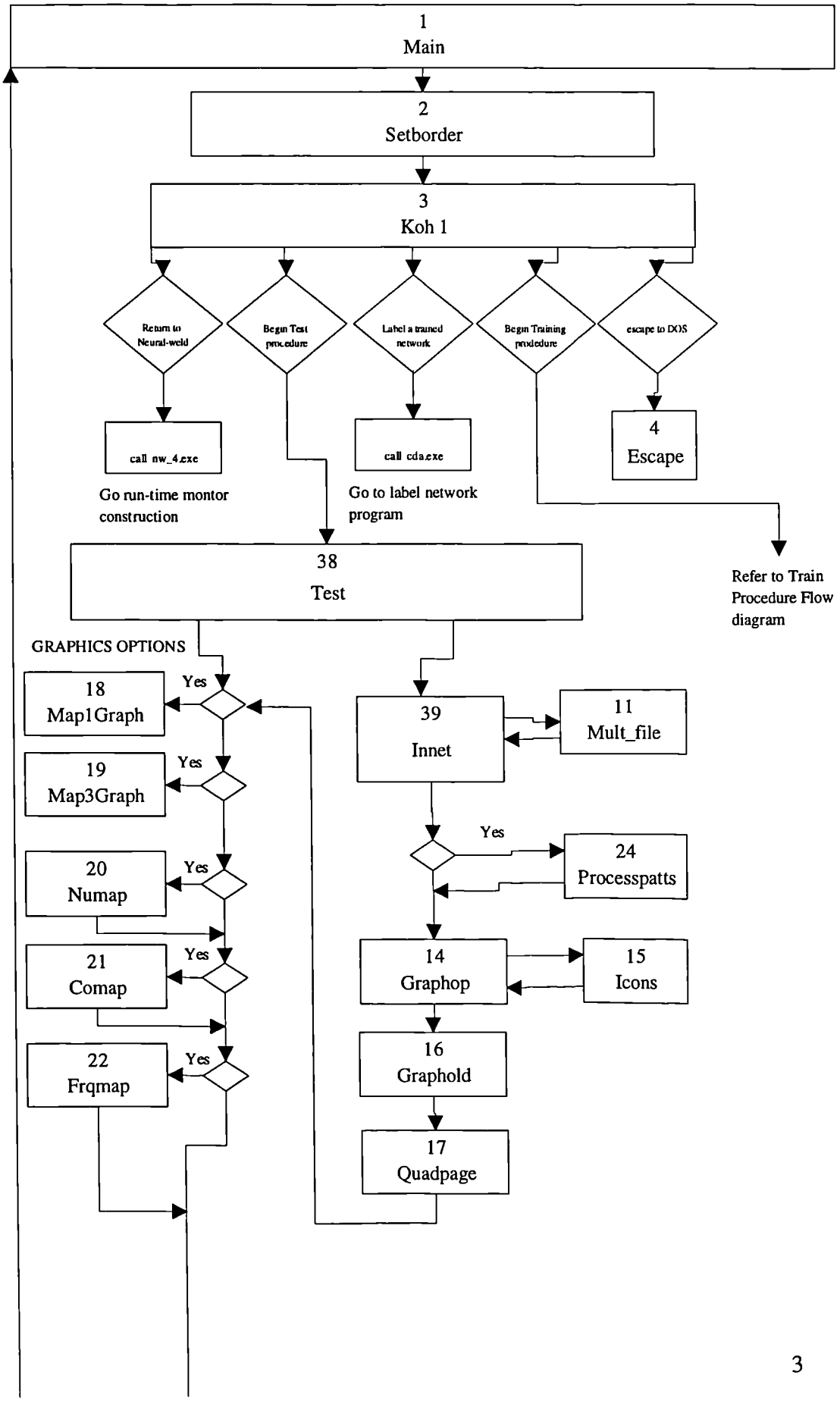

If you are NOT happy with the recording you can repeat it.
Re-typing the same file name WILL overwrite the existing
file.
( )CCEPT data file/s and return to start.
( )EPEAT or record another file.

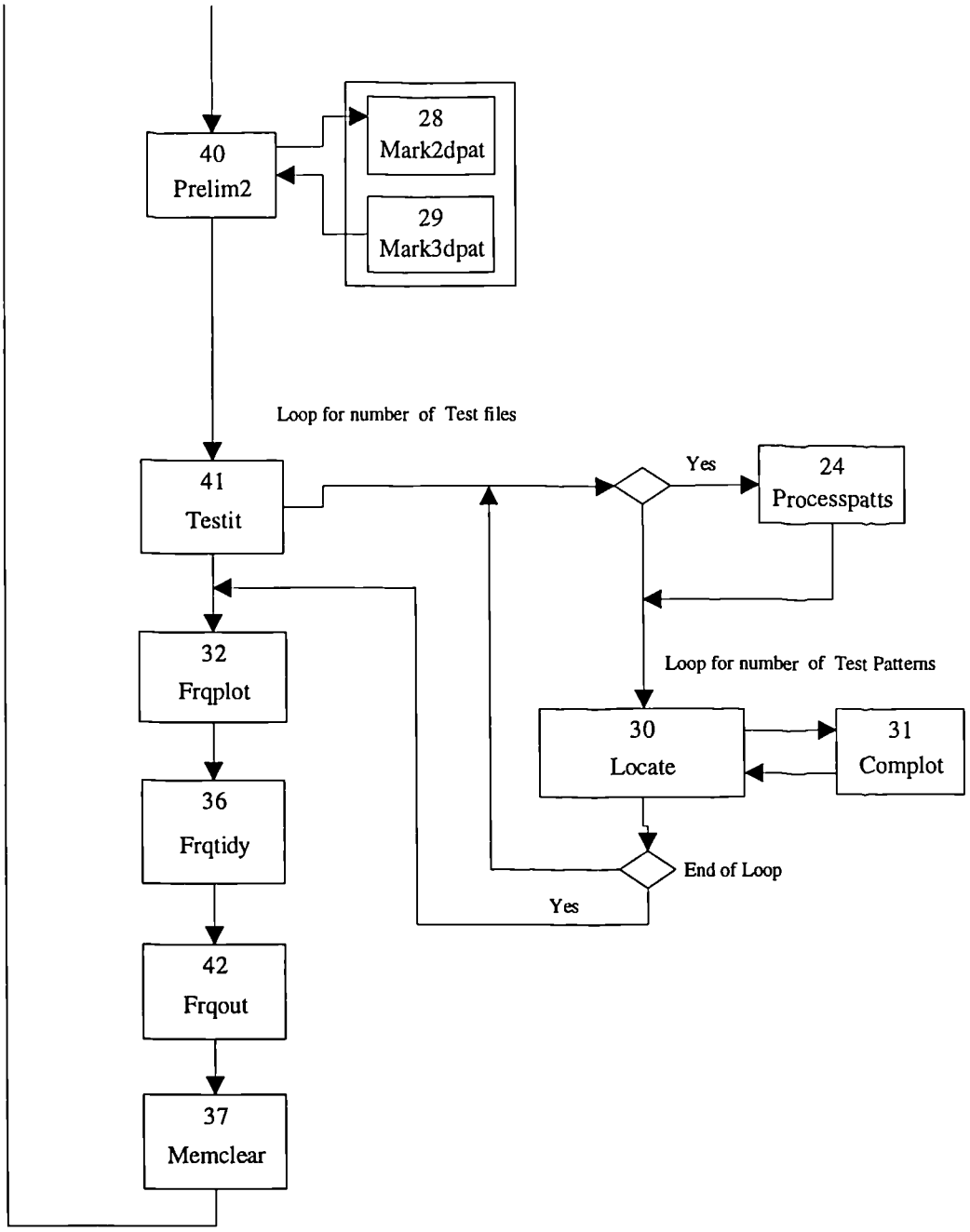
Flow Chart for Self Organising Feature Map (SOFM_3.c)




14
Graphop
15
Icons
16
Graphold
17
Quadpage
37
Memclear
23
Train
Loop for number of patterns
Yes
24
Processpatts
Keyboard hit?
Yes
25
Command
34
map1plot
35
Map3plot
26
Change
36
Frqtidy
27
Prelim
28
Mark2dpat
29
Mark3dpat
30
Locate
31
Complot
32
Frqplot
33
Limit

Flow Chart for Self Organising Feature Map (SOFM_3.c)




40
Prelim2
28
Mark2dpat
29
Mark3dpat
Loop for number of Test files
41
Testit
Yes
24
Processpatts
32
Frqplot
Loop for number of Test Patterns
30
Locate
31
Complot
36
Frqtidy
End of Loop
Yes
42
Frqout
37
Memclear

FUNCTIONS KEY

2 void setborder(short color);	Sets clour & format for screen border
3 void koh1(void);	Sets title page & switches to test, train, NW or escape
5 void train_proc(void);	Resets any training parameters and calls training procedure save all necessary files post training
6 void setdatatext(void);	Sets graphics page for training parameters
7 void holddata(void);	Takes and sets training parameters, calls and sets appropriate memory allocation
8 void calcout(void);	Calculates No of outputs nodes to form output layer
9 void neightype(void);	Switch for LIMIT or WRAP (presently redundant) style neighbourhood
10 void koh2(void);	Takes and sets training data file parameters
11void mult_file(void);	Multi-test/train file input
12 void preweight(void);	Takes pretrained network for continual training
13 void random(void);	Randomizes contents of synaptic weight matrix
14 void graphop(void);	Text page for viewing graphics option
15 void icons(void);	Draws graphics icons
16 void graphold(void);	Takes and sets viewing graphics options
17 void quadpage(void);	Divides screen into four for view graphics
18 void map1graph (void);	Draws basic 2D graph background with titles, sets a graphics origin
19 void map3graph (void);	Draws basic 3D graph background with titles, sets a graphics origin
20 void numap (void);	Sets background text for numerical data with titles
21 void comap (void);	Sets background with titles for input pattern/weight comparison
22 void frqmap (void);	Sets backgound with titles for firing frequency viewer
23 void train(void);	Full procedure for training algorithm
24 void processpatts(void);	Normalises input pattern matrix
25void command(void);	Switch for online training parameter control
26void change(void);	Changes online training parameters
27void prelim(void);	Views initial input data before training
28 void mark2dpat(void);	Draws initial density of 2D training/test data
29void mark3dpat(void);	Draws initial density of 3D training/test data
30 void locate(void);	Computes Euclidean errors and No of winning neuron
31 void complot(void);	Plots online input pattern and weight component comparison
32 void frqplot(void);	Plots online firing frequency of output nodes
33 void limit(void);	Locates winner in output layer, defines neighbourhood, adapts synaptic weights
34 void map1plot(void);	Plots online 2D synaptic weights as mesh
35 void map3plot(void);	Plots online 3D synaptic weights as mesh
36 void frqtidy(void);	Redraws firing frequency plot with mesh
37 void memclear(void);	Calls all procedures for freeing allocated memory
38 void test(void);	Preliminary test procedure and graphics setup
39 void innet(void);	Loads all pretrained network data
40 void prelim2(void);	Views initial input data before testing
41 void testit(void);	Full test iteration on data file
42 void frqout (void);	Open and save outlayer frequency map

J. R. McCardle 1997

```
/*****************************************************************************
*                                                                            *
*  SOFM.C (Self Organising Feature Map).                                     *
*  This program is the Kohonen Self Organising Feature Map algorithm for     *
*  use with Neural-Weld N.A.M. (Neural Acoustic Monitor). A good reference   *
*  is the program source code written and documented by Dobbins & Eberhart   *
*  (version 1.0 RWD 28 Jan 1990), included in their book "Neural Network     *
*  PC Tools".                                                                *
*  This version however, is dedicated to the Neural-weld package and whilst  *
*  retaining some of theory of the above version, is substantially           *
*  different in operation.                                                   *
*                                                                            *
*  Compiled by J.R. McCardle, Neural Applications Group, Brunel University   *
*  Department of Design.                                                     *
*****************************************************************************/

/* Inclusion headers */

#include   <stdio.h>
#include   <stdlib.h>
#include   <math.h>
#include   <conio.h>
#include   <ctype.h>
#include   <string.h>
#include   <graph.h>
#include   <time.h>
#include   <dos.h>
#include   <process.h>
#include   <share.h>
#include   <errno.h>
#include   <malloc.h>

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Program construction prototypes and Typedefs */

typedef float *PTR_FLOAT;
typedef PTR_FLOAT VECTOR;
typedef PTR_FLOAT *MATRIX;

int vectormem(VECTOR *ptr_vector, int Ncolumns);            /* Allocates memory for vectors */
void columnmem(PTR_FLOAT matrix[], int Nrows, int Ncolumns);/* Allocates memory for columns in..*/
                                                            /* matrix */
int matrixmem(MATRIX *ptr_matrix, int Nrows, int Ncolumns); /* Allocates memory for variable matrix */
void matrixdel(MATRIX matrix, int Nrows, int Ncolumns);     /* Releases all memory allocated to..*/
                                                            /* matrices */

void setborder(short color);  /* Sets clour & format for screen border */
void Enterkey(void);          /* Looks for & registers return key press */

void koh1(void);              /* Sets title page & switches to test, train, NW or escape */
void koh2(void);              /* Takes and sets training data file parameters */

void setdatatext(void);       /* Sets graphics page for training parameters */
void holddata(void);          /* Takes and sets training parameters, calls and sets appropriate..*/
                              /* memory allocation */

void calcout(void);           /* Calculates No of outputs nodes to form output layer */
void neightype(void);         /* Switch for LIMIT or WRAP (presently redundant) style neighbourhood */

void graphop(void);           /* Text page for viewing graphics option */
void icons(void);             /* Draws graphics icons */
void graphold(void);          /* Takes and sets viewing graphics options */
void quadpage(void);          /* Divides screen into four for view graphics */

void map1graph (void);        /* Draws basic 2D graph background with titles, sets a graphics origin */
void map3graph (void);        /* Draws basic 3D graph background with titles, sets a graphics origin */
void ndimengraph (void);      /* Reserved for N-dimensional space interpretation */
void numap (void);            /* Sets background text for numerical data with titles */
void comap (void);            /* Sets background with titles for input pattern/weight comparison */
void frqmap (void);           /* Sets backgound with titles for firing frequency viewer */
void frqout (void);           /* Open and save outlayer frequency map */

void processpatts(void);      /* Normalises input pattern matrix */
void preweight(void);         /* Takes pretrained network for continual training */
void random(void);            /* Randomizes contents of synaptic weight matrix */

void train_proc(void);        /* Resets any training parameters and calls training procedure,..*/
                              /* save all necessary files post training */
void train(void);             /* Full procedure for training algorithm */

void test(void);              /* Preliminary test procedure and graphics setup */
void testit(void);            /* Full test iteration on data file */
void innet(void);             /* Loads all pretrained network data */
void mult_file(void);         /* Multi-test/train file input */


void command(void);           /* switch for online training parameter control */
void change(void);            /* Changes online training parameters */
void prelim(void);            /* Views initial input data before training */
void prelim2(void);           /* Views initial input data before testing */

void mark2dpat(void);         /* Draws initial density of 2D training/test data */
void mark3dpat(void);         /* Draws initial density of 3D training/test data */

void locate(void);            /* Computes Euclidean errors and No of winning neuron */
void limit(void);             /* Locates winner in output layer, defines neighbourhood, adapts synaptic
weights */


void memclear(void);          /* Calls all procedures for freeing allocated memory */
void yesno(void);             /* Registers a yes or no answer as TRUE/FALSE */
```

```
void map1plot(void);          /* Plots online 2D synaptic weights as mesh */
void map3plot(void);          /* Plots online 3D synaptic weights as mesh */
void comlplot(void);          /* Plots online input pattern and weight component comparison */
void frqplot(void);           /* Plots online firing frequency of output nodes */
void frqtidy(void);           /* Redraws firing frequency plot with mesh */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Global defines */

#define INCOLOR         0       /* black */
#define BACKGNDCOLOR    3       /* cyan */
#define AXISCOLOR       1       /* blue */
#define GRAPHCOLOR      4       /* red */
#define TITLECOLOR      14      /* yellow */
#define LABELCOLOR      15      /* white */

#define ALWAYS    1            /* decision flags */
#define TRUE      1
#define FALSE     0

/* Composition of "printat" */
#define printat(x, y, z)        _settextposition(x, y);\
                _outtext(z);


#define YCENTER 240             /* Center pixel coordinates of graphics */
#define XCENTER 320             /* screen                  */

     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
          /* DYNAMIC ARRAY STORAGE DEFINES */

MATRIX inlayer;
MATRIX outlayer;
MATRIX syn_weights;

VECTOR pattern_no;
VECTOR previn;
VECTOR prevweight;
VECTOR interlay;
VECTOR interweigh;
VECTOR fire;
VECTOR order;
VECTOR euc_err;

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Global variables */

short   xorig,yorig;    /* Origins for graphics pages */

unsigned int tra;               /* train or test flags */
unsigned int tes;
unsigned int traf;              /* No of train and/or test files */
unsigned int tesf;


unsigned int side;              /* Side dimension of output layer */
unsigned int end;               /* End dimension of output layer */
unsigned int  rep;              /* Repeat procedure flag */
unsigned long p;                /* inpatts flag */
unsigned long o;                /* outnodes flag */
unsigned long n;                /* innodes flag */
unsigned int win;               /* winning neuron */
unsigned int neighbside;        /* neighbourhood dimensions of ratio 5:8 */
unsigned int left, right, up, down; /* limits of output layer for neighbourhood */
unsigned int wrap;              /* wrap around neighbourhood flag */
unsigned int lim;               /* limited neighbourhood flag */
unsigned int map3;              /* Flags for graphics options */
unsigned int map1;
unsigned int num;
unsigned int comp;
unsigned int frq;
unsigned int tog;               /* component vector toggle */
unsigned int norm;              /* normalisation routine */
unsigned int ans;               /* Yes or no flag */
unsigned int manual;            /* manual step or autorun flags */
unsigned int automatic;
unsigned int tesfno;            /* No of present test file */
unsigned int trafno;            /* No of present train file */

unsigned long innodes;          /* Number of input nodes */
unsigned long inpatts;          /* Number of input patterns */
unsigned long outnodes;         /* Number of ouput nodes */
unsigned long epochs;           /* Number of training epochs */
unsigned long ordepochs;        /* Number of ordering epochs */
unsigned long e;                /* epochs flag */
unsigned long itsupdate;        /* Number of iterations before neighbouhood & learning rate decrement */
unsigned long its;              /* present iteration */
unsigned long itsdone;          /* iterations completed */
unsigned long gritsdone;        /* graphics iterations */
unsigned long gupdate;          /* Epochs before graphics update */

float inneighperc;      /* Initial size of ouput neural neighbourhood */
float lcoefval;         /* Value of learning coefficient */
float lcoefdec;         /* Factor of learning coefficient decrement */
float big;              /* largest possible % neighbourhood */
float defneigh;         /* Default neibourhood size as a fraction of output layer */
float defval;           /* Default learning coefficient */
float defdec;           /* Default learning coefficient decrement */
float Small_Euc_Dist;   /* Smallest Euclidean distance gives winning neuron */
float bar;      /* error vector total for error bar plot */

FILE *infile, *outweights, *inweights;     /* Pointer for recorded pattern file */
```

```
FILE *outfrq;                                                    /* and output save files */

char inpath[_MAX_PATH], outpathweights[_MAX_PATH], inpathweights[_MAX_PATH];
char outpathfrq[_MAX_PATH];

char testfiles[10][_MAX_PATH];
char trainfiles[10][_MAX_PATH];

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*****************************************************************************/

void main(void) {


/* TO ENSURE RUN_TIME FLOATING POINT LIB IS LOADED ONLY ! thanks Microsoft !!!*/

    defneigh = 1.0F;    /* Set default neighbourhood */
    defval = 0.60F;     /* Set default learning coeff */
    defdec = 0.20F;     /* Set default learning coeff decrement */


_setvideomode(_MAXRESMODE);

_setbkcolor(_BLACK);

setborder(11);

  while(ALWAYS) {
    tra = 0;    /* reset flags */
    tes = 0;

    _setvieworg(0,0);
    koh1();

    /* run train procedure */

    if(tra) {
        train_proc();
     }

    /* otherwise do test routine */

    else{
         test();
      }

   } /* end of always loop */

} /* End of main() */

/*****************************************************************************/

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void koh1(void) {

int input[3];
int ok;
ok = 0;



{
_settextcolor(GRAPHCOLOR);
printat(yorig + 2, xorig + 25, " KOHONEN UNSUPERVISED TRAINING");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 5, xorig, " The Kohonen Self Organising Feature Map will be trained on the data stored")
;
printat(yorig + 6, xorig, " in the declared input files.");
printat(yorig + 7, xorig, " Data files can be created with NEURAL-WELD or other methods that generate");
printat(yorig + 8, xorig, " numerical ASCII text.");

printat(yorig + 11, xorig, " Pre-trained networks can be loaded and executed, on line, in NEURAL-WELD.")
;
printat(yorig + 13, xorig, " Otherwise the test procedure can be used to subject a trained");
printat(yorig + 14, xorig, " network with test data files.");

}

{
_settextcolor(3);
printat(yorig + 17, xorig + 24, " (C)ONTINUE and begin training.");
printat(yorig + 19, xorig + 24, " (R)ETURN to Neural-Weld.");
printat(yorig + 21, xorig + 24, " (T)EST trained network.");
printat(yorig + 23, xorig + 24, " (E)SCAPE to DOS.");

/* Little red circles */
_setcolor(4);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 260, xorig + 187, yorig + 267);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 292, xorig + 187, yorig + 299);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 324, xorig + 187, yorig + 331);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 356, xorig + 187, yorig + 363);

          /*--------------------------------------*/

                  /* Switch modes */

while(input[0] = getch())
{
```

```
    switch(input[0]) {
    case 'c':
    case 'C': memclear();
          _clearscreen(_GCLEARSCREEN);       /* continue with train */
          tra = 1;                           /* set train procedure flag */
          ok = 1;
          break;

    case 'e':
    case 'E': memclear();                       /* Escape to DOS */
          _clearscreen(_GCLEARSCREEN);
          _setvideomode(_DEFAULTMODE);
          _displaycursor(_GCURSORON);
          _settextposition(25,81);
          exit(0);

    case 'r':
    case 'R': memclear();
          _clearscreen(_GCLEARSCREEN);
          execl("nw_4.exe", NULL);          /* Call Neural-Weld procedure */
          exit(1);

    case 't':
    case 'T': memclear();
          _clearscreen(_GCLEARSCREEN);      /* continue with test */
          tes = 1;
          ok = 1;
          break;

           }
    if (ok == 1) break;
}
}

}   /* end of koh1() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void train_proc(void) {

setdatatext();

REDO:
holddata();

vectormem(&euc_err, (short)inpatts);  /* set array memory for error bar plot */

_settextcolor(GRAPHCOLOR);
printat( 27, 21, " Do you wish to change the parameters ?");
printat( 28, 22, "             (Y)es or (N)o");

yesno();

if(ans) {
    printat( 27, 22, "                                          ");
    printat( 28, 22, "                            ");
    memclear();
    goto REDO;
}


printat( 27, 22, "                                          ");
printat( 28, 22, "                              ");

printat( 28, 20, " OK! now press ENTER to declare I/O files.");

Enterkey();

flushall();

_clearscreen(_GCLEARSCREEN);
koh2();

_clearscreen(_GCLEARSCREEN);
graphop();
graphold();

_clearscreen(_GCLEARSCREEN);
quadpage();

_settextcolor(TITLECOLOR);
printat(30, 19,"(P)ause    (H)alt    (T)oggle    (C)hange ");

if(map1 && !map3) {
    map1graph();
}

if(map3 && !map1) {
    map3graph();
}

if(num) {
    numap();
}

if(comp) {
    vectormem(&previn, (short)innodes);     /* set dynamic arrays for swap files */
    vectormem(&prevweight, (short)innodes);
    vectormem(&interlay, (short)innodes);
    vectormem(&interweigh, (short)innodes);
    comap();
```

```
}

if(frq) {
       vectormem(&fire, (short)outnodes);  /* set dynamic array for firing frq */
       frqmap();
}

train();

printat(15, 24, "                              ");
printat(16, 30, "                       ");

    /* Save constructed weight matrix */

_settextcolor(TITLECOLOR);
printat( 15, 21, " OK! now press ENTER to save all files.");
Enterkey();

printat( 15, 21, "                                          ");

printat(15, 27, "SAVING NETWORK WEIGHT MATRIX");
printat(16, 27, "        PLEASE WAIT.");

outweights = fopen(outpathweights, "wb");

/* print file header */
fprintf(outweights, "%ld\n%ld\n%d\n%d\n", outnodes, innodes, side, end);

for(o = 0; o < outnodes; o++) {        /* print data to file comma separated */
    for(n = 0; n < innodes; n++) {
          fprintf(outweights, "%f,", syn_weights[o][n]);
    }
    fprintf(outweights, "\n");
}
fclose(outweights);

printat(15, 27, "                              ");

printat(14, 27, "TRAINING IS NOW COMPLETE AND");
printat(15, 27, "      ALL FILES SAVED.");
printat(16, 24, "Press ENTER to return to MENU PAGE");

Enterkey();
_clearscreen(_GCLEARSCREEN);    /* clearscreen and reset origin */
_setvieworg(0, 0);


} /* end of train_proc() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void train(void)
{
unsigned char dat[12];
char mes[45];


unsigned long i, c, b;
float r, d, a;

            /*--------------------------------------*/
/******************** BEGINNING OF TRAINING INTERATIONS *****************/
            /*--------------------------------------*/
               /* 1. Parameter initialisation */

its = 0;
itsdone = 0;
gritsdone = 0;

neighbside = (side * inneighperc) + 0.5; /* compute initial neighbourhood dimensions */

        /*--------------------------------------*/
            /* 2. Start iteration loops */


for(e = 0; e < epochs + ordepochs; e++) {
/*  && lcoefval > 0.1 could use || to maintain either value */

 if( e == epochs) {                    /* change network data if ordering count commenced */
     _settextcolor(TITLECOLOR);
     printat(15, 26, "                              ");
     printat(15, 30, "   ORDERING COUNT   ");

     lcoefval = 0.1F;        /* minimum learning coefficient */
     lcoefdec = 1.0F;        /* no change to lcoefval */
     neighbside = 0;         /* winner only to update */

     if(num) {                                    /* change numerical output */
     printat(11, 67, "           ");
     _settextposition(11, 67);
     _settextcolor(TITLECOLOR);
     sprintf(dat, "%f", lcoefdec);
     _outtext(dat);
      }
}

 if(num) {
      printat(8, 67, "           ");          /* update numerical output */
      _settextcolor(TITLECOLOR);
      _settextposition(8, 57);
      sprintf(dat, "(T=%d)", epochs + ordepochs);
      _outtext(dat);
```

```
      _settextposition(8, 67);
      sprintf(dat, "%d", e);
      _outtext(dat);

      printat(10, 67, "          ");
      _settextposition(10, 67);
      _settextcolor(TITLECOLOR);
      sprintf(dat, "%f", lcoefval);
      _outtext(dat);
     }


  for(trafno = 0; trafno < traf; trafno++) {

/* open and read train file */
     if( (infile = fopen( &trainfiles[trafno][0], "rb")) == NULL ) { /* error message */
    _settextcolor(GRAPHCOLOR);
    printf("\a"); /* Alarm */
    sprintf( mes, " Can't open pattern file, error number: %d", errno);
    printat(17, 18, mes);
    printat(18, 16, " Drive and/or Directory has to exist to use !");
    printat(20, 20, " Press ENTER to repeat declaration.");

    Enterkey();

/* Erase error message */
    printat(17, 18, "                                                    ");
    printat(18, 16, "                                                    ");
    printat(20, 20, "                                                    ");
    printat(yorig + 9, xorig + 62, "                    "); /* Erase path */

    exit(0);    /* escape to dos */
      }

     for(i = 0; i < inpatts; i++) {
    for(c = 0; c < innodes; c++) {
        fscanf(infile, "%f,", &r);
        inlayer[i][c] = r;
        }
    fscanf(infile, "\n");
    }

     if(norm) {       /* perform normalisation if required */
    processpatts();
       }

     fclose(infile);


prelim();  /* view input pattern dispertion and file No */

            /*--------------------------------------*/

    for(p = 0; p < inpatts; p++) {


            /*--------------------------------------*/
              /* 3. Locate winning output neuron */

       locate();


             /*--------------------------------------*/
 /* 4. Define winner position, neighbourhood and limits around winning neuron */

       limit();

             /*--------------------------------------*/
       /* 5. Test for number of completed pattern iterations */


       its++; /* increment iteration count */

   if(num) {                                /* update numerical output */
        printat(7,67, "          ");
        _settextposition(7, 67);
        _settextcolor(TITLECOLOR);
        sprintf(dat, "%d", its);
        _outtext(dat);

        _settextposition(6, 57);
        sprintf(dat, "(T=%d)",inpatts);
        _outtext(dat);
        _settextposition(6, 67);
        sprintf(dat, "%ld", p);
        _outtext(dat);
      }

   if(kbhit()) {        /* check keyboard interupt */
      command();
    }

  } /* end of pattern presentation */
/* compute error bar for pattern */

  a = 0.0F;
  d = 0.0F;
    for(i = 0; i < inpatts; i++) {
    a = euc_err[i];
    d += (a * a);
     }

 bar = sqrt(d);
```

```
if(num) {                   /* update error bar output */
        printat(13,67, "          ");
        _settextposition(13, 67);
        _settextcolor(TITLECOLOR);
        sprintf(dat, "%f", bar);
        _outtext(dat);
       }

for(i = 0; i < inpatts; i++){    /* reset error bar matrix for next epoch */
    euc_err[i] = 0;
     }

 ) /* end of run  of test files list */

    itsdone++;              /* increment iterations flags */
    gritsdone++;

         /*--------------------------------------*/
          /* 6. Test for end of iteration and break or loop */
         /* Reduce learning coefficient and neighbourhood size */

    if(map1 || map3) {                              /* update graphics */
          if( gritsdone == gupdate ) {
          if(map1){
             map1plot();
           }
          else{
             map3plot();
           }
          gritsdone = 0;
           }
     }

    if(itsdone == itsupdate) {          /* re parameterise */

       lcoefval *= lcoefdec;

       if(lcoefval < 0.1F) {
          lcoefval = 0.1F;
       }

       if(neighbside > 0) {
          neighbside -= 1;   /*remove one neuron from neighbourhood dimension */
       }

       itsdone = 0;      /* reset flag */
    }

if(frq) {            /* call frequency firing map tidy up */
    frqtidy();       /* after each epoch */
    for(b = 0; b < outnodes; b++){   /* reset firing frq for next epoch */
    fire[b] = 0;
     }
}


) /* end of epoch loop */

/************************** END OF TRAINING ROUTINE **********************/
           /*--------------------------------------*/

printf("\a");  /* Beep at end of training */

) /* end of train() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void prelim(void) {

unsigned char dat[12];


if(map1) {       /* view training pattern dispertion */
    mark2dpat();
    if(e == 0){     /* do if first epoch only */
    map1graph();
    mark2dpat();
    _settextcolor(TITLECOLOR);
    printat( 15, 26, "TRAINING PATTERN DISPERTION");
    Enterkey();
    printat( 15, 26, "                              ");
    map1plot();
    printat( 15, 28, "INITIAL SYNAPTIC WEIGHTS");
    Enterkey();
    printat( 15, 28, "                              ");
     }
}

if(map3) {
    mark3dpat();
    if(e == 0){ /* do if first epoch only */
       map3graph();
       mark3dpat();
      _settextcolor(TITLECOLOR);
       printat( 15, 26, "TRAINING PATTERN DISPERTION");
       Enterkey();
       printat( 15, 26, "                               ");
       map3plot();
       printat( 15, 28, "INITIAL SYNAPTIC WEIGHTS");
       Enterkey();
```

```
      printat( 15, 28, "                          ");
    }
  )

if(num) {
    printat(4, 67, "            ");
    _settextcolor(TITLECOLOR);
    _settextposition(4, 67);
    sprintf(dat, "%s", &trainfiles[trafno][0]);
    _outtext(dat);


    _settextposition(5, 67);
    sprintf(dat, "%ld / %ld", innodes, outnodes);
    _outtext(dat);

    printat(11, 67, "            ");
    _settextposition(11, 67);
    sprintf(dat, "%f", lcoefdec);
    _outtext(dat);
}
if(e < epochs){            /* print until ordering count */
    printat( 15, 26, "                          ");
    _settextcolor(TITLECOLOR);
    printat(15, 30, "TRAINING IN PROGRESS");
 )

} /* end of prelim() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void locate(void)
             /* 4. Locate winning output neuron */
{
unsigned char dat[3];

float d;
unsigned long keep_small;
float Euc_Dist;       /* Actual Euclidean distance */


      Small_Euc_Dist = 0.0F;

      for(o = 0; o < outnodes; o++) {

          Euc_Dist = 0.0F;

          for(n = 0; n < innodes; n++) {
          d = inlayer[p][n] - syn_weights[o][n];
          Euc_Dist += d * d;
          }

          if(o == 0 || Euc_Dist <= Small_Euc_Dist) {
          Small_Euc_Dist = Euc_Dist;
          win = (short)o;
          }
     )

for(keep_small = 0; keep_small < p; keep_small++){ /* Set error bar matrix for pattern */
    euc_err[p] = Small_Euc_Dist;
     }

  if(comp) {                        /* call component vector plot */
       complot();
       }

  if(tra && frq) {                        /* call firing frequency map plot */
       frqplot();
       }

  if(num) {                            /* update numerical output */
       printat(12, 67, "      ");
       _settextposition(12, 67);
       _settextcolor(TITLECOLOR);
       sprintf(dat, "%d", win);
       _outtext(dat);
   )


} /* end of locate */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void limit(void)
{

unsigned char dat[5];
float   neighbourhood; /* actual neighbourhood as a percentage of output layer */

unsigned int alpha; /* first output neuronal index of neighbourhood */
unsigned int winside; /* position of winning neuron in output slab */
unsigned int winend;
unsigned int c;
unsigned int per;
unsigned int x;
        /* FOR 2D OUTPUT SLAB OF DIMENSIONS SIDE x END */

{ /* Begin procedure */
```

```
    if(win < side) {
       winside = win;
       winend = 0;
     }
    else {
       winside = win % side;
       winend = win / side;
     }

    left = winside;           /* gives total number of available neurons in output */
    up = winend;                        /* layer for use by neighbourhood */
    right = (side - winside) - 1;
    down = (end - winend) - 1;

  /* set limits on neighbourhood */

    left = min(left, neighbside / 2);
    right = min(right, neighbside / 2);
    up = min(up, neighbside / 2);
    down = min(down, neighbside / 2);

  /* identify beginning of neighbourhood in terms of output neuronal index */

    x = win - (up * side);
    alpha = x - left;             /* alpha is neuronal index of neighbourhood start */

          /*--------------------------------------*/
               /* 5a. Adapt synaptic weights */

    for(c = 0, per = 0; c < up + down + 1; c++, alpha += side) {
       for(o = alpha; o < alpha + left + right + 1; o++, per++) {
        for(n = 0; n < innodes; n++) {
        syn_weights[o][n] += lcoefval * (inlayer[p][n] - syn_weights[o][n]);
        }
    }
    }



if(num) {
    /* calculate neighbourhood as a percentage of output layer */

    neighbourhood = per * 100.0F / (float)outnodes;

    printat(9, 67, "           ");
    _settextcolor(TITLECOLOR);
    if( per == 1 ) {
        printat(9, 67, "Winner");
        }
    else {
         sprintf(dat, "%.1f %%", neighbourhood);
        _settextposition(9, 67);
        _outtext(dat);
       }
  }


 } /* End of procedure */

} /* end of limit() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void command(void) {

int cmd[3];

cmd[0] = getch();

switch(cmd[0]) {
    case 'p':
    case 'P': Enterkey();
          break;

    case 'h':
    case 'H': _settextcolor(GRAPHCOLOR);
          printat(16, 30, "HALTING AT NEXT EPOCH");
          e = epochs + ordepochs;
          break;

    case 't':
    case 'T': if(tog) {
        tog = 0;
        }
          else {
        tog = 1;
        }
          break;

    case 'c':
    case 'C': change();
          _settextcolor(TITLECOLOR);
          printat(30, 19,"(P)ause    (H)alt    (T)oggle    (C)hange ");
          break;

default: break;
           }        /* end of switch */

} /* end of command() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void change(void) {

int cmd[3];
int ok = 0;
char mess[10];

printat(30, 19,"                                                          ");
_settextcolor(TITLECOLOR);
printat(30, 19,"(L)earn     (D)ec     (N)eighb     (R)eturn ");

while(cmd[0] = getch()) {
switch(cmd[0]) {
        case 'r':
        case 'R': ok = 1;
              break;

        case 'l':
        case 'L': _settextposition(10, 65);
              _settextcolor(GRAPHCOLOR);
              _outtext("\x10");
              _settextposition(10, 67);
              scanf("%f", &lcoefval);
              sprintf(mess, "%f", lcoefval);
              _settextcolor(TITLECOLOR);
              _outtext(mess);
              _settextcolor(3);
              printat(10, 65, "=");
              break;

        case 'd':
        case 'D': _settextposition(11, 65);
              _settextcolor(GRAPHCOLOR);
              _outtext("\x10");
              _settextposition(11, 67);
              scanf("%f", &lcoefdec);
              sprintf(mess, "%f", lcoefdec);
              _settextcolor(TITLECOLOR);
              _outtext(mess);
              _settextcolor(3);
              printat(11, 65, "=");
              break;


        case 'n':
        case 'N': _settextposition(9, 65);
              _settextcolor(GRAPHCOLOR);
              _outtext("\x10");
              _settextposition(9, 67);
              scanf("%f", &inneighperc);
              sprintf(mess, "%.1f %%", inneighperc);
              _settextcolor(TITLECOLOR);
              _outtext(mess);
              inneighperc /= 100.0;
              neighbside = (side * inneighperc) + 0.5; /* compute new neighbourhood dimensions */
              _settextcolor(3);
              printat(9, 65, "=");
              break;

        default: break;

    }   /* end of switch */

    if (ok == 1) break;
 }

} /* end of change */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void setdatatext(void)


         /*--------------------------------------*/

               /* Set data page1 */

{
_settextcolor(GRAPHCOLOR);
printat(yorig + 2, xorig + 30, " TRAINING PARAMETERS");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 4, xorig + 11, " Please Enter the desired network parameters at the prompts.");

_settextcolor(3);
printat(yorig + 7, xorig + 5, " NUMBER OF INPUT NEURONS (N-WELD = 64).................");
printat(yorig + 9, xorig + 5, " OUTPUT LAYER CONFIGURATION (X x Y).......... ........");
printat(yorig + 11, xorig + 5, " NUMBER OF INPUT PATTERNS PER TRAIN FILE..............");
printat(yorig + 13, xorig + 5, " NUMBER OF EPOCHS.....................................");
printat(yorig + 15, xorig + 5, " NUMBER OF EPOCHS FOR RE-PARAMETERISATION.............");
printat(yorig + 17, xorig + 5, " NUMBER OF EPOCHS FOR ORDERING COUNT..................");
printat(yorig + 19, xorig + 5, " INITIAL NEIGHBOURHOOD PERCENTAGE (MAXIMUM =       )..");
printat(yorig + 21, xorig + 5, " WRAP AROUND OR LIMITED NEIGHBOURHOOD ( W / L ).........");
printat(yorig + 23, xorig + 5, " LEARNING COEFFICIENT VALUE (0.1 - 1.00)...............");
printat(yorig + 25, xorig + 5, " LEARNING COEFFICIENT DECREASE FACTOR (0.00 - 1.00)....");
```

```
/* Little red circles */
_setcolor(4);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 100, xorig + 27, yorig + 107);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 132, xorig + 27, yorig + 139);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 164, xorig + 27, yorig + 171);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 196, xorig + 27, yorig + 203);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 228, xorig + 27, yorig + 235);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 260, xorig + 27, yorig + 267);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 292, xorig + 27, yorig + 299);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 324, xorig + 27, yorig + 331);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 356, xorig + 27, yorig + 363);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 388, xorig + 27, yorig + 395);

) /* End of setdatatext() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void holddata(void)
          /*--------------------------------------*/

          /* Set input lines and read input data */

{
char mess[10];   /* output data strings for screen */
unsigned int swap;


do {

{
rep = 0;    /* reset repeat flag */

/* Erase any previous error message */
printat(27, 22, "                                          ");
printat(28, 18, "                                                  ");
printat(29, 25, "                                  ");

/* Clear any erroneous inputs */
printat(yorig + 7, xorig + 62, "                    ");  /* 18 chars */
printat(yorig + 9, xorig + 62, "                    ");
printat(yorig + 11, xorig + 62, "                    ");
printat(yorig + 13, xorig + 62, "                    ");
printat(yorig + 15, xorig + 62, "                    ");
printat(yorig + 17, xorig + 62, "                    ");
printat(yorig + 19, xorig + 62, "                    ");
printat(yorig + 19, xorig + 50, "        ");
printat(yorig + 21, xorig + 62, "                    ");
printat(yorig + 23, xorig + 62, "                    ");
printat(yorig + 25, xorig + 62, "                    ");

}

          /*--------------------------------------*/

/* Ask for number of input nodes */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 7, xorig + 60);
_outtext("\x10");                          /* ASCII delta to right character */
_settextposition(yorig + 7, xorig + 62);
 scanf("%ld", &innodes);
sprintf(mess,"%ld", innodes);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 7, xorig + 60, " ");     /* Erase delta cursor */


         /*--------------------------------------*/

/* Ask for ouput layer configuration */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 9, xorig + 60);
_outtext("\x10");                           /* ASCII delta to right character */
_settextposition(yorig + 9, xorig + 62);
scanf("%d x %d", &side, &end);

if(end > side) {          /* end should never be greater than side */
    swap = end;
    end = side;
    side = swap;
    }

sprintf(mess,"%d x %d", side, end);
_settextcolor(TITLECOLOR);
_outtext(mess);

          /*--------------------------------------*/

/* Compute total output layer and maximum neighbourhood size */


calcout();
printat(yorig + 9, xorig + 60, " ");     /* Erase delta cursor */

         /*--------------------------------------*/

/* Ask for number of input patterns */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 11, xorig + 60);
_outtext("\x10");
```

```
_settextposition(yorig + 11, xorig + 62);
scanf("%ld", &inpatts);
sprintf(mess,"%ld", inpatts);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 11, xorig + 60, " ");     /* Erase delta cursor */


      /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
            /* ALLOCATE DYNAMIC ARRAY STORAGE */

vectormem(&pattern_no, (short)inpatts);                  /* Allocate memory for vector */
if (rep == 1) {
          _settextcolor(TITLECOLOR);
          printat(29, 32, "TOO MANY PATTERNS");
          matrixdel(inlayer, (short)inpatts, (short)innodes + 1);/* Free memory already allocated */
          free(pattern_no);
          Enterkey();
          goto SKIP;                                     /* Abort next memory allocation */
          }

matrixmem(&inlayer, (short)inpatts, ((short)innodes + 1));      /* Allocate memory for matrix */
if (rep == 1) {                    /* +1 for vector augmentation and 2d on 3d plot availability */
           _settextcolor(TITLECOLOR);
           printat(29, 25, "TOO MANY PATTERNS OR INPUT NODES");
           matrixdel(inlayer, (short)inpatts, ((short)innodes +1));/* Free memory already allocated */
           free(pattern_no);
           Enterkey();
           goto SKIP;                                    /* Abort next memory allocation */
           }

matrixmem(&outlayer, (short)inpatts, (short)outnodes);   /* Allocate memory for matrix */
if (rep == 1) {
          _settextcolor(TITLECOLOR);
          printat(29, 25, "TOO MANY PATTERNS OR OUTPUT NODES");
          matrixdel(inlayer, (short)inpatts, ((short)innodes +1));/* Free memory already allocated */
          matrixdel(outlayer, (short)inpatts, (short)outnodes);
          free(pattern_no);
          Enterkey();
          goto SKIP;                                     /* Abort next memory allocation */
          }

matrixmem(&syn_weights, (short)outnodes, ((short)innodes + 1));/* Allocate memory for matrix */
if (rep == 1) {
          _settextcolor(TITLECOLOR);
          printat(29, 27, "TOO MANY INPUT OR OUTPUT NODES");
          matrixdel(inlayer, (short)inpatts, ((short)innodes +1));/* Free memory already allocated */
          matrixdel(outlayer, (short)inpatts, (short)outnodes);
          matrixdel(syn_weights, (short)outnodes, ((short)innodes +1));
          free(pattern_no);
          Enterkey();
          }

     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
SKIP:
continue;

}while(rep == 1);

        /*---------------------------------------*/

/* Ask for number of training epochs */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 13, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 13, xorig + 62);
scanf("%ld", &epochs);
sprintf(mess,"%ld", epochs);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 13, xorig + 60, " ");

         /*---------------------------------------*/

/* Ask for number of re-parameterisation epochs */

do{

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 15, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 15, xorig + 62);
scanf("%ld", &itsupdate);

if(itsupdate > epochs) {            /* Cannot be larger than no of epochs */
               printf("\a");        /* Alarm */
               _settextposition(yorig + 15, xorig + 62);
               printf("          ");                 /* Erase value */
               }

} while(itsupdate > epochs);           /* Repeat until acceptable value */

sprintf(mess,"%ld", itsupdate);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 15, xorig + 60, " ");

         /*---------------------------------------*/

/* Ask for number of ordering epochs */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 17, xorig + 60);
```

```
_outtext("\x10");
_settextposition(yorig + 17, xorig + 62);
scanf("%ld", &ordepochs);
sprintf(mess,"%ld", ordepochs);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 17, xorig + 60, " ");  /* Erase delta cursor */

          /*--------------------------------------*/

/* Ask for percentage of initial neighbourhood */

do {
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 19, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 19, xorig + 62);
scanf("%f", &inneighperc);

 if(inneighperc > big) {                                  /* Maximum 100 */
               printf("\a");                          /* Alarm */
               _settextposition(yorig + 19, xorig + 62);
               printf("          ");                  /* Erase value */
               }

} while(inneighperc > big);            /* Repeat until acceptable value */


sprintf(mess,"%.1f %%", inneighperc);
_settextcolor(TITLECOLOR);
_outtext(mess);                /* Output value to screen */

inneighperc /= 100.0;                  /* Convert to fraction < 1 */

printat(yorig + 19, xorig + 60, " ");  /* Erase delta cursor */

          /*--------------------------------------*/

neightype();

           /*--------------------------------------*/

/* Ask for value of learning coefficient */

do {
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 23, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 23, xorig + 62);
scanf("%f", &lcoefval);

  if((lcoefval > 1.0F) || (lcoefval < 0.1F)) {    /* Must be less than 1 and greater than 0.01 */
               printf("\a");                  /* Alarm */
               _settextposition(yorig + 23, xorig + 62);
               printf("          ");
               }

} while((lcoefval > 1.0F) || (lcoefval < 0.1F) );   /* repeat until acceptable value */

sprintf(mess, "%f", lcoefval);
_settextcolor(TITLECOLOR);
_outtext(mess);                /* ouput value to screen */
printat(yorig + 23, xorig + 60, " ");

           /*--------------------------------------*/

/* Ask value of learning coefficient decrement */

do {
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 25, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 25, xorig + 62);
scanf("%f", &lcoefdec);

  if( lcoefdec > 1.0F ) {                          /* Must be less than 1 */
            printf("\a");
            _settextposition(yorig + 25, xorig + 62);
            printf("          ");
            }

} while( lcoefdec > 1.0F );          /* Repeat until acceptable value */

sprintf(mess,"%f", lcoefdec);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 25, xorig + 60, " ");  /* Erase delta cursor */


} /* End of holddata() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

     /* Recalculate No of output nodes and largest % neighbourhood */

void calcout(void) {

float temp;

     /* output data strings for screen */
char new[24];
char maxneigh [7];
```

```
    outnodes = side * end;      /* Calculate total number of output nodes */

     _settextposition(yorig + 9, xorig + 62);
     sprintf(new, "Layer = %dx%d=%ld", side, end, outnodes);  /* Show output layer dimensions */
    _outtext(new);


/* print maximum percentage of neighbourhood */
       big = end * end;
       big /= outnodes;                                    /* could also = side / end * 100 */
       big *= 100.0;
       _settextposition(yorig + 19, xorig + 50);
       sprintf(maxneigh, "%.1f %%", big);
       _outtext(maxneigh);

} /* end of calcout() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void neightype(void)
{
int ok;
int input[1];

/* Ask for wrap around or boundary limited neighbourhood */
     {
      _settextcolor(GRAPHCOLOR);
      _settextposition(yorig + 21, xorig + 60);
      _outtext("\x10");
      _settextposition(yorig + 21, xorig + 62);

    ok = FALSE;
    while(input[0] = getch())
       {

        switch(input[0]) {
           case 'w':
           case 'W':
                _settextcolor(TITLECOLOR);
                _outtext("WRAP");                  /* ouput value to screen */
                wrap = TRUE;
                ok = TRUE;
                break;

           case 'l':
           case 'L':
                _settextcolor(TITLECOLOR);
                _outtext("LIMIT");                 /* ouput value to screen */
                lim = TRUE;
                ok = TRUE;
                break;
        }
     if( ok == TRUE) break;
    }
      printat(yorig + 21, xorig + 60, " ");  /* Erase delta cursor */
    }

}  /* end of neightype */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void koh2(void) {

int  c;
int  i;

float r;

char drive[_MAX_DRIVE], dir[_MAX_DIR];
char fname[_MAX_FNAME], ext[_MAX_EXT];
char mess[45];
char dat[12];

{
_settextcolor(GRAPHCOLOR);
printat(yorig + 2, xorig + 30, "FILE DECLARATIONS");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 4, xorig + 9, " Please Enter the desired file names & paths at the prompts.");
printat(yorig + 5, xorig + 12, " Extentions are awarded automatically for OUTPUT files.");

_settextcolor(3);
printat(yorig + 7, xorig + 5, " NUMBER OF INPUT PATTERN TRAIN FILES ..................");
printat(yorig + 9, xorig + 5, " EUCLIDEAN NORMALISATION REQUIRED (Y/N) ................");
printat(yorig + 11, xorig + 5, " FILE TO STORE SYNAPTIC WEIGHTS (*.nww automatic).....");


/* Little red circles */
_setcolor(4);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 100, xorig + 27, yorig + 107);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 132, xorig + 27, yorig + 139);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 164, xorig + 27, yorig + 171);

}

         /*--------------------------------------*/

/* Ask for number of train files */
```

```
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 7, xorig + 60);
_outtext("\x10");                          /* ASCII delta to right character */
_settextposition(yorig + 7, xorig + 62);
scanf("%d", &traf);
sprintf(dat,"%d", traf);
_settextcolor(TITLECOLOR);
_outtext(dat);
printat(yorig + 7, xorig + 60, " ");       /* Erase delta cursor */
}

/* check for normalisation */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 9, xorig + 60);
_outtext("\x10");                          /* ASCII delta to right character */

yesno();

norm = ans;

_settextcolor(TITLECOLOR);
_settextposition(yorig + 9, xorig + 62);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 9, xorig + 60, " ");     /* Erase delta cursor */

}

flushall();

/* Ask for output file name for synaptic weight matrix */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 11, xorig + 60);
_outtext("\x10");                          /* ASCII delta to right character */

do{
_settextposition(yorig + 11, xorig + 62);
gets(outpathweights);
_settextcolor(TITLECOLOR);
_splitpath( outpathweights, drive, dir, fname, ext);     /* create output path */
strcpy ( ext, "nww" );
_makepath( outpathweights, drive, dir, fname, ext );
_outtext(outpathweights);

if( (outweights = _fsopen( outpathweights, "wb", SH_DENYWR )) == NULL ) /* error message */

  {
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     sprintf( mess, " Can't open weights file, error number: %d", errno);
     printat(17, 18, mess);
     printat(18, 16, " Drive and/or Directory has to exist to use !");
     printat(20, 20, " Press ENTER to repeat declaration.");

     Enterkey();

     /* Erase error message */
     printat(17, 18, "                                                  ");
     printat(18, 16, "                                                  ");
     printat(20, 20, "                                                  ");
     printat(yorig + 9, xorig + 64, "                       "); /* Erase path */


   }
} while( outweights == NULL );  /* Loop declaration until valid */

fclose(outweights);
printat(yorig + 11, xorig + 60, " ");  /* Erase delta prompt */

}

         /*--------------------------------------*/

if(traf > 1) {
 mult_file();    /* set page for multiple file input list */
 }

else{
    _settextcolor(3);

     printat(yorig + 13, xorig + 5, " FILE NAME OF TRAIN PATTERNS [DIMS EXPECTED =    ].....");
    _setcolor(4);
    _ellipse(_GFILLINTERIOR, xorig + 20, yorig + 196, xorig + 27, yorig + 203);

    _settextcolor(GRAPHCOLOR);
    _settextposition(yorig + 13, xorig + 61);
    _outtext("\x10");

    sprintf(mess,"%ld", innodes);
    _settextcolor(TITLECOLOR);
    _settextposition(yorig + 13, xorig + 51);
    _outtext(mess);

    for(trafno = 0; trafno < traf; trafno++) {
    _settextposition(yorig + 13, xorig + 62);
    gets(&trainfiles[trafno][0]);
```

```
    _settextcolor(TITLECOLOR);
    sprintf(mess,"%s", &trainfiles[trafno][0]);
    _outtext(mess);
      }

    printat(yorig + 13, xorig + 61, " ");   /* Erase delta prompt */

} /* end of else */

preweight();

_settextcolor(GRAPHCOLOR);
printat(22, 30, "FILE OPTIONS ACCEPTED");
printat(24, 25, "Press ENTER for graphics options.");

Enterkey();


} /* End of koh2 */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

               /* TEST PROCEDURE */

void test(void) {

unsigned int ok;
unsigned long i, c, b;

int cmd[3];
float r, d, a;

char mess[45];
unsigned char dat[12];

{

  innet();  /* fetch test data parameters */

  vectormem(&euc_err, (short)inpatts);  /* set array memory for error bar plot */

  _settextcolor(GRAPHCOLOR);
  printat(22, 30, "FILE OPTIONS ACCEPTED");
  printat(24, 25, "Press ENTER for graphics options.");

  Enterkey();

  _clearscreen(_GCLEARSCREEN);
  graphop();
  graphold();

  _clearscreen(_GCLEARSCREEN);
  quadpage();

  if(map1 && !map3) {
     map1graph();
  }

  if(map3 && !map1) {
     map3graph();
  }

  if(num) {
     numap();
  }

  if(comp) {
    vectormem(&previn, ((short)innodes + 1));     /* set dynamic arrays for swap files */
    vectormem(&prevweight, ((short)innodes + 1));
    vectormem(&interlay, ((short)innodes + 1));
    vectormem(&interweigh, ((short)innodes + 1));
    comap();
  }

  if(frq) {
       vectormem(&fire, (short)outnodes);      /* set dynamic array for firing frq */
       vectormem(&order, (short)inpatts);
       frqmap();
  }

  prelim2();

  _settextcolor(TITLECOLOR);
  printat(30, 32,"(A)uto     (M)anual ");

  ok = FALSE;
  while (cmd[0] = getch()) {            /* select automatic or manual test */
     switch (cmd[0]) {
         case 'a':
         case 'A': automatic = TRUE;
                   manual = FALSE;
                   _settextcolor(GRAPHCOLOR);
                   printat(30, 32,"(A)uto");
                   ok = 1;
                   break;

         case 'm':
         case 'M': manual = TRUE;
                   automatic = FALSE;
                   _settextcolor(GRAPHCOLOR);
                   printat(30, 42,"(M)anual ");
                   ok = 1;
```

```
                break;
            }               /* end of switch */
     if(ok == 1) break;

   } /* end of while switch loop */


if(tesf > 1) {
   for( tesfno = 0; tesfno < tesf; tesfno++) {

    if( (infile = fopen( &testfiles[tesfno][0], "rb")) == NULL ) { /* error message */
       _settextcolor(GRAPHCOLOR);
       printf("\a");              /* Alarm */
       sprintf( mess, " Can't open pattern file, error number: %d", errno);
       printat(17, 18, mess);
       printat(18, 16, " Drive and/or Directory has to exist to use !");
       printat(20, 20, " Press ENTER to repeat declaration.");

       Enterkey();

       /* Erase error message */
       printat(17, 18, "                                                    ");
       printat(18, 16, "                                                    ");
       printat(20, 20, "                                                    ");
       printat(yorig + 15, xorig + 62, "                      "); /* Erase path */

     }

     for(i = 0; i < inpatts; i++) {
         for(c = 0; c < innodes; c++) {
         fscanf(infile, "%f,", &r);
         inlayer[i][c] = r;
          }
         fscanf(infile, "\n");
      }

     if(norm) {          /* perform normalisation if required */
         processpatts();
      }

     fclose(infile);

     if(frq) {
        frqmap();
      }

     testit();

/* compute error bar for pattern */

  a = 0.0F;
  d = 0.0F;
    for(i = 0; i < inpatts; i++) {
    a = euc_err[i];
    d += (a * a);
     }

 bar = sqrt(d);

 if(num) {                  /* update error bar output */
        printat(13,67, "         ");
        _settextposition(13, 67);
        _settextcolor(TITLECOLOR);
        sprintf(dat, "%f", bar);
        _outtext(dat);
       }

for(i = 0; i < inpatts; i++){    /* reset error bar matrix for next epoch */
    euc_err[i] = 0;
     }


     if(frq) {
        frqtidy();
      }

     frqout();

     for(b = 0; b < outnodes; b++) { /* reset firing frequency matrix */
         fire[b] = 0;
      }
     for(b = 0; b < inpatts; b++) { /* reset firing order matrix     */
         order[b] = 0;
      }

    } /* end of for tesfno loop */
} /* end of if tesf */

 if(tesf == 1) {

   testit();

 /* compute error bar for pattern */

   a = 0.0F;
   d = 0.0F;
     for(i = 0; i < inpatts; i++) {
     a = euc_err[i];
     d += (a * a);
      }

  bar = sqrt(d);

  if(num) {                  /* update error bar output */
```

```
        printat(13,67, "         ");
        _settextposition(13, 67);
        _settextcolor(TITLECOLOR);
        sprintf(dat, "%f", bar);
        _outtext(dat);
       }

for(i = 0; i < inpatts; i++){    /* reset error bar matrix for next epoch */
    euc_err[i] = 0;
     }

     if(frq) {
      frqtidy();
    }
   frqout();
}

_settextcolor(TITLECOLOR);
printf("\a");   /* alarm */
printat(30, 19,"Iteration Test complete. Enter to return...");

Enterkey();

_clearscreen(_GCLEARSCREEN);

}

} /* end of test() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void testit(void) {

unsigned char dat[12];

float t = 0.23271F; /* tan 13.1 deg */
float xa, xb, ya, yb, Z, X, Y;          /*coordinates for input pattern */
float xa1, xb1, ya1, yb1, Z1, X1, Y1;   /*coordinates for winning neuron */

{

for( p = 0; p < inpatts; p++ ) { /* draw individual input patterns */

    locate();      /* find closest neuron */

    if(map1) {
       _setvieworg(5, 50);                              /* reset origin */
       _setcolor(LABELCOLOR);    /* change input pattern colours to white */

       for(n = 0; n < 1; ) {
        xa = inlayer[p][n];
        n++;
        ya = inlayer[p][n];
      }
      xa *= 300;
      ya *= 170;
    _ellipse(_GFILLINTERIOR, (int)xa - 1, (int)ya - 1, (int)xa + 1, (int)ya + 1);

      _setvieworg(5, 50);                          /* reset origin */
      _setcolor(GRAPHCOLOR);            /* draw winning neuron in red */

      for(n = 0; n < 1; ) {
        xb = syn_weights[win][n];
        n++;
        yb = syn_weights[win][n];
      }
      xb *= 300;
      yb *= 170;
    _ellipse(_GFILLINTERIOR, (int)xb - 2, (int)yb - 2, (int)xb + 2, (int)yb + 2);

    _setcolor(LABELCOLOR);         /* join input pattern to winning neuron */
    _moveto((int)xa, (int)ya);
    _lineto((int)xb, (int)yb);

  _setcolor(0);  /* change input pattern back to cyan */
  _ellipse(_GFILLINTERIOR, (int)xa - 1, (int)ya - 1, (int)xa + 1, (int)ya + 1);
  _setcolor(TITLECOLOR);        /* change neuron back to yellow */
  _ellipse(_GFILLINTERIOR, (int)xb - 2, (int)yb - 2, (int)xb + 2, (int)yb + 2);

   } /* end of map1 procedure */


if(map3) {
     _setvieworg(5, 50);                                /* reset origin */
     _setcolor(LABELCOLOR);      /* change input pattern colours to white */

    for(n = 0; n < 2; ) {
        xa = inlayer[p][n];
        n++;
        ya = inlayer[p][n];
        n++;
        Z = inlayer[p][n];
     }

     xa *= 150;    /* scale inputs */
     ya *= 150;
     Z *= 100;

     xb = 150 - xa;
     yb = 150 + ya;

     X = yb - xa;
     Y = 100 + (t * ( xa + ya ) - Z) ;
```

```
    _ellipse(_GFILLINTERIOR, (int)X - 1, (int)Y - 1, (int)X + 1, (int)Y + 1);

    _setvieworg(5, 50);                    /* reset origin */
    _setcolor(GRAPHCOLOR);                /* draw winning neuron in red */

     for(n = 0; n < 2; ) {
        xa1 = syn_weights[win][n];
        n++;
        ya1 = syn_weights[win][n];
        n++;
        Z1 =  syn_weights[win][n];
     }

     xa1 *= 150;     /* scale inputs */
     ya1 *= 150;
     Z1 *= 100;

     xb1 = 150 - xa1;
     yb1 = 150 + ya1;

     X1 = yb1 - xa1;
     Y1 = 100 + (t * ( xa1 + ya1 ) - Z1) ;


    _ellipse(_GFILLINTERIOR, (int)X1 - 2, (int)Y1 - 2, (int)X1 + 2, (int)Y1 + 2);

    _setcolor(LABELCOLOR);          /* join input pattern to winning neuron */
    _moveto((int)X, (int)Y);
    _lineto((int)X1, (int)Y1);

  _setcolor(AXISCOLOR);  /* change input pattern back to blue */
  _ellipse(_GFILLINTERIOR, (int)X - 1, (int)Y - 1, (int)X + 1, (int)Y + 1);
  _setcolor(TITLECOLOR);         /* change neuron back to yellow */
  _ellipse(_GFILLINTERIOR, (int)X1 - 2, (int)Y1 - 2, (int)X1 + 2, (int)Y1 + 2);

   } /* end of if map3 */


if(num) {                                        /* print pattern No */
     printat(6, 67, "            ");
     _settextcolor(TITLECOLOR);
     _settextposition(6, 57);
     sprintf(dat, "(T=%d)",inpatts);
     _outtext(dat);
     _settextposition(6, 67);
     sprintf(dat, "%d", p);
     _outtext(dat);
       }

if(frq) {
   frqplot();
 }

if(manual) {
  Enterkey();
  }

}
}

}  /* end of pattern iteration test testit() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


void innet(void) {    /* load pre-trained network data */

unsigned long  c, i;
float r;
char mess[45];


_settextcolor(GRAPHCOLOR);
printat(yorig + 2, xorig + 30, "FILE DECLARATIONS");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 4, xorig, " N.B. ENSURE THE TEST DATA AND NETWORK MATRIX ARE OF THE SAME
DIMENSIONALITY !");

_settextcolor(3);

printat(yorig + 7, xorig + 5, " FILE NAME FOR PRE-TRAINED WEIGHT MATRIX ( ?.nww ).....");
printat(yorig + 9, xorig + 5, " NORMALISATION REQUIRED ON TEST PATTERNS ? (Y/N) .....");
printat(yorig + 11, xorig + 5, " NUMBER OF INPUT TEST PATTERNS PER FILE.....");
printat(yorig + 13, xorig + 5, " NUMBER OF DATA FILES TO TEST .....");


_settextcolor(GRAPHCOLOR);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 100, xorig + 27, yorig + 107);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 132, xorig + 27, yorig + 139);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 164, xorig + 27, yorig + 171);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 196, xorig + 27, yorig + 203);


 /* ask for pre-trained weight matrix */
 _settextcolor(GRAPHCOLOR);
 _settextposition(yorig + 7, xorig + 60);
 _outtext("\x10");
 _settextposition(yorig + 7, xorig + 62);
```

```
flushall();   /* clear buffer */

   do{
     _settextposition(yorig + 7, xorig + 62);
     gets(inpathweights);
     _settextcolor(TITLECOLOR);
     _outtext(inpathweights);

     if( (inweights = fopen( inpathweights, "rb")) == NULL ) {   /* error message */
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     sprintf( mess, " Can't open weights file, error number: %d", errno);
     printat(17, 18, mess);
     printat(18, 16, " Drive and/or Directory has to exist to use !");
     printat(20, 20, " Press ENTER to repeat declaration.");

     Enterkey();

     /* Erase error message */
    printat(17, 18, "                                                ");
    printat(18, 16, "                                                ");
    printat(20, 20, "                                                ");
    printat(yorig + 7, xorig + 62, "                    "); /* Erase path */

    }

 } while( inweights == NULL);   /* Loop declaration until valid */


_settextcolor(GRAPHCOLOR);
printat(22, 27, "PROCESSING DATA PLEASE WAIT");

fscanf(inweights, "%ld\n%ld\n%d\n%d\n", &outnodes, &innodes, &side, &end);   * read header */

matrixmem(&syn_weights, (short)outnodes, ((short)innodes + 1)); /* set matrix */

for(i = 0; i < outnodes; i++) {      /* read data */
    for(c = 0; c < innodes; c++) {
    fscanf(inweights, "%f,", &r);
    syn_weights[i][c] = r;
      }
      fscanf(inweights, "\n");
 }

fclose(inweights);

printat(22, 27, "                            ");

printat(yorig + 7, xorig + 60, " ");  /* Erase delta prompt */


/* check for normalisation */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 9, xorig + 60);
_outtext("\x10");                        /* ASCII delta to right character */

yesno();

norm = ans;

_settextcolor(TITLECOLOR);
_settextposition(yorig + 9, xorig + 62);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 9, xorig + 60, " ");    /* Erase delta cursor */

}

/* Ask for number of input patterns */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 11, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 11, xorig + 62);
scanf("%ld", &inpatts);
sprintf(mess,"%ld", inpatts);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 11, xorig + 60, " ");     /* Erase delta cursor */


vectormem(&pattern_no, (short)inpatts);
matrixmem(&inlayer, (short)inpatts, ((short)innodes + 1));  /* set matrices */
matrixmem(&outlayer, (short)inpatts, (short)outnodes);


/* Ask for number of data files */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 13, xorig + 60);
_outtext("\x10");                        /* ASCII delta to right character */
_settextposition(yorig + 13, xorig + 62);
scanf("%d", &tesf);
sprintf(mess,"%d", tesf);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 13, xorig + 60, " ");     /* Erase delta cursor */
}
```

```
if(tesf == 1) {

/* Ask for drive and file declaration */
/* and load input pattern matrix */

flushall();  /* clear all buffers */
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 228, xorig + 27, yorig + 235);
_settextcolor(3);
printat(yorig + 15, xorig + 5, " FILE NAME OF TEST PATTERNS [DIMS EXPECTED =    ].....");

sprintf(mess,"%ld", innodes);
_settextcolor(TITLECOLOR);
_settextposition(yorig + 15, xorig + 50);
_outtext(mess);

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 15, xorig + 60);
_outtext("\x10");                        /* ASCII delta to right character */

do {

_settextposition(yorig + 15, xorig + 62);
gets(inpath);
_settextcolor(TITLECOLOR);
_outtext(inpath);

  if( (infile = fopen( inpath, "rb")) == NULL )  /* error message */

  {
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     sprintf( mess, " Can't open pattern file, error number: %d", errno);
     printat(17, 18, mess);
     printat(18, 16, " Drive and/or Directory has to exist to use !");
     printat(20, 20, " Press ENTER to repeat declaration.");

     Enterkey();

     /* Erase error message */
     printat(17, 18, "                                                  ");
     printat(18, 16, "                                                  ");
     printat(20, 20, "                                                  ");
     printat(yorig + 15, xorig + 62, "                      "); /* Erase path */

   }

} while( infile == NULL);       /* Loop declaration until valid */

_settextcolor(GRAPHCOLOR);
printat(22, 27, "PROCESSING DATA PLEASE WAIT");

for(i = 0; i < inpatts; i++) {
  for(c = 0; c < innodes; c++) {
    fscanf(infile, "%f,", &r);
    inlayer[i][c] = r;
      }
       fscanf(infile, "\n");
  }

if(norm) {           /* perform normalisation if required */
   processpatts();
}

fclose(infile);

printat(22, 27, "                           ");

printat(yorig + 15, xorig + 60, " ");   /* Erase delta prompt */
}

if(tesf > 1) {
     mult_file();
     }

} /* end of innet() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void mult_file(void) {

unsigned int a, yg, yt;
char q[2];
char mess[45];


{
 /* draw file input box and title */
_setcolor(8);
_rectangle(_GFILLINTERIOR, XCENTER - 150, YCENTER - 200, XCENTER + 170, YCENTER + 200);
_setcolor(0);
_rectangle(_GFILLINTERIOR, XCENTER - 160, YCENTER - 210, XCENTER + 160, YCENTER + 190);
_setcolor(3);
_rectangle(_GBORDER, XCENTER - 160, YCENTER - 210, XCENTER + 160, YCENTER + 190);

_settextcolor(GRAPHCOLOR);
printat(4, 31, "INPUT FILE LIST");
_settextcolor(3);
printat(5, 23, " PATTERN DIMENSION EXPECTED =    ");
sprintf(mess,"%ld", innodes);
_settextcolor(TITLECOLOR);
_settextposition(5, 53);
```

```
_outtext(mess);


 /* for No of tesf or traf number slots */
if(tes) {
    for(a = 1, yt = 7; a <= tesf; a++, yt += 2) {
       _settextcolor(GRAPHCOLOR);
       _settextposition(yt, 25);
       sprintf(q, "%d.", a);
       _outtext(q);
      }
  }

  if(tra) {
    for(a = 1, yt = 7; a <= traf; a++, yt += 2) {
       _settextcolor(GRAPHCOLOR);
       _settextposition(yt, 25);
       sprintf(q, "%d.", a);
       _outtext(q);
      }

  }

  /* take in file names and paths and label as strings */

flushall();

if(tes) {
    for(tesfno = 0, yt = 7; tesfno < tesf; tesfno++, yt += 2) {
    _settextcolor(GRAPHCOLOR);
    _settextposition(yt, 28);
    _outtext("\x10");          /* ASCII delta to right character */
    _settextposition(yt, 30);
    gets(&testfiles[tesfno][0]);
    _settextcolor(TITLECOLOR);
    sprintf(mess,"%s", &testfiles[tesfno][0]);
    _outtext(mess);
    printat(yt, 28, " ");
      }
 }

if(tra) {
    for(trafno = 0, yt = 7; trafno < traf; trafno++, yt += 2) {
    _settextcolor(GRAPHCOLOR);
    _settextposition(yt, 28);
    _outtext("\x10");          /* ASCII delta to right character */
    _settextposition(yt, 30);
    gets(&trainfiles[trafno][0]);
    _settextcolor(TITLECOLOR);
    sprintf(mess,"%s", &trainfiles[trafno][0]);
    _outtext(mess);
    printat(yt, 28, " ");
      }
 }


}

} /* end of mult_file() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void prelim2(void) {

unsigned char dat[12];


if(map1) {           /* view training pattern dispertion */
    mark2dpat();
    _settextcolor(TITLECOLOR);
    printat( 15, 27, "TEST PATTERN DISPERTION");
    Enterkey();
    printat( 15, 26, "                              ");
    map1plot();
    printat( 15, 28, "TRAINED SYNAPTIC WEIGHTS");
    Enterkey();
    printat( 15, 28, "                              ");
    tes = 1; /* set test procedure flag here to initiate winning neuron marker */
}

if(map3) {
    mark3dpat();
    _settextcolor(TITLECOLOR);
    printat( 15, 27, "TEST PATTERN DISPERTION");
    Enterkey();
    printat( 15, 26, "                              ");
    map3plot();
    printat( 15, 28, "TRAINED SYNAPTIC WEIGHTS");
    Enterkey();
    printat( 15, 28, "                              ");
    tes = 1;
    }

if(num) {
    _settextcolor(3);                  /* headings */
    printat(4, 42, "TEST FILE              =");
    _settextcolor(GRAPHCOLOR);
    printat(8, 67, "Test Run");
    printat(9, 67, "Test Run");
    printat(10, 67, "Test Run");
    printat(11, 67, "Test Run");
```

```
    _settextcolor(TITLECOLOR);
    _settextposition(5, 67);
    sprintf(dat, "%ld / %ld", innodes, outnodes);
    _outtext(dat);

    printat(7, 67, "           ");
    _settextposition(7, 67);
    sprintf(dat, "%d", inpatts);
    _outtext(dat);

}

_settextcolor(TITLECOLOR);
printat(15, 30, "TESTING IN PROGRESS");

} /* end of prelim2() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


/* ROUTINES FOR MEMORY ALLOCATION */

int vectormem(VECTOR *ptr_vector, int Ncolumns)
{
{    *ptr_vector = (VECTOR) calloc(Ncolumns, sizeof(float));

    if( *ptr_vector == NULL)
    {                                          /* sets memory allocation for vectors */
     _settextcolor(GRAPHCOLOR);
   printf("\a");
   printat(27, 22, "MEMORY TOO SMALL FOR VECTOR ALLOCATION");
   printat(28, 18, "Press ENTER and try smaller number of patterns.");
   rep = 1;
   Enterkey();
    }

}
}  /* end of vectormem */
        /* - - - - - - - - - - - - - - - - */

int matrixmem(MATRIX *ptr_matrix, int Nrows, int Ncolumns)
{
{   *ptr_matrix = (MATRIX) calloc(Nrows, sizeof(float));

   if(*ptr_matrix == NULL)
   {                                           /* sets memory allocation for matrices */
   _settextcolor(GRAPHCOLOR);
   printf("\a");
   printat(27, 22, "MEMORY TOO SMALL FOR MATRIX ALLOCATION");
   printat(28, 18, "Press ENTER and try smaller nodal combinations.");
   rep = 1;
   Enterkey();
   }

columnmem(*ptr_matrix, Nrows, Ncolumns);

}
}   /* end of matrixmem */

         /* - - - - - - - - - - - - - - - - */

void columnmem(PTR_FLOAT matrix[], int Nrows, int Ncolumns)
{
int i;
      for(i = 0; i < Nrows; i++) {              /* sets up column matrix */
       if(rep != 1) {
      vectormem(&matrix[i], Ncolumns);
       }
}
}    /* end of columnmem */

        /* - - - - - - - - - - - - - - - - */

void matrixdel(MATRIX matrix, int Nrows, int Ncolumns)
{
int i, a;
      for(i = 0; i < Nrows; i++){                /* deletes all matrices */
     for(a = 0; a < Ncolumns; a++){
      free(matrix);
     }
    free(matrix[i]);
      }
}    /* end of matrixdel */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void graphop(void)
{

int a, x, y;

_settextcolor(GRAPHCOLOR);
printat(yorig + 2, xorig + 32, " GRAPHICS OPTIONS");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 4, xorig + 11, " NOTE : Any graphics choice will increase the training time.");

_settextcolor(3);
```

```
printat(yorig + 8, xorig + 5, " MAP FOR 1 OR 2 DIMENSIONAL DATA (Y/N).....");
printat(yorig + 13, xorig + 5, " MAP FOR 3 DIMENSIONAL DATA (Y/N).....");
printat(yorig + 18, xorig + 5, " NETWORK NUMERICAL OUTPUT (Y/N).....");
printat(yorig + 23, xorig + 5, " VECTOR COMPONENT ERROR (Y/N).....");
printat(yorig + 28, xorig + 5, " FREQUENCY ACTIVATION MAP (Y/N).....");


/* Little red circles */
_setcolor(4);
for(a = 0, x = 20, y = 116; a < 5; a++, y += 80) {
    _ellipse(_GFILLINTERIOR, xorig + x, yorig + y, xorig + (x + 7), yorig + (y + 7));
}

icons();

} /* End of graphop() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void icons(void) {

int a, b, x, y, x1, y1;

/* icon 1 */

{
_setcolor(8);
_rectangle(_GFILLINTERIOR, xorig + 505, yorig + 75, xorig + 600, yorig + 135);
_setcolor(3);
_rectangle(_GFILLINTERIOR, xorig + 500, yorig + 70, xorig + 595, yorig + 130);
_setcolor(AXISCOLOR);
_rectangle(_GBORDER, xorig + 500, yorig + 70, xorig + 595, yorig + 130);

_setcolor(TITLECOLOR);
for(a = 0, y = 82; a < 3; y += 17, a++) {
     for(b = 0, x = 512; b < 5; x += 17, b++) {
     _ellipse(_GFILLINTERIOR, xorig + x, yorig + y, xorig + (x + 3), yorig + (y + 3));
       }
}

_setcolor(4);
for(a = 0, x = 516; a < 4; x +=17, a++) {
    for(b = 0, y = 83; b < 3; y += 17, b++) {
    _moveto(xorig + x, yorig + y);
    _lineto(xorig + (x + 11), yorig + y);
     }
 }

for(a = 0, x = 513; a < 5; x +=17, a++) {
    for(b = 0, y = 86; b < 2; y += 17, b++) {
    _moveto(xorig + x, yorig + y);
    _lineto(xorig + x, yorig + (y + 11));
     }
 }

}

/* icon2 */
{
_setcolor(3);
_moveto(xorig + 550, yorig + 150);
_lineto(xorig + 500, yorig + 165);
_lineto(xorig + 550, yorig + 180);
_lineto(xorig + 600, yorig + 165);
_lineto(xorig + 550, yorig + 150);
_lineto(xorig + 550, yorig + 210);
_lineto(xorig + 600, yorig + 195);
_lineto(xorig + 550, yorig + 180);
_lineto(xorig + 500, yorig + 195);
_lineto(xorig + 550, yorig + 210);

_moveto(xorig + 500, yorig + 165);
_lineto(xorig + 500, yorig + 195);
_moveto(xorig + 600, yorig + 195);
_lineto(xorig + 600, yorig + 165);

_setcolor(TITLECOLOR);


for(a = 0, x1 = 505, y1 = 182; a < 5; a++, x1 += 10, y1 += 3) {
   for(b = 0, x = x1, y = y1; b < 6; b++, x += 10, y -= 3) {
     _ellipse(_GFILLINTERIOR, xorig + x, yorig + y, xorig + (x + 3), yorig + (y + 3));
    }
}
}

/* icon 3 */
{
_setcolor(8);
_rectangle(_GFILLINTERIOR, xorig + 505, yorig + 230, xorig + 600, yorig + 290);
_setcolor(3);
_rectangle(_GFILLINTERIOR, xorig + 500, yorig + 225, xorig + 595, yorig + 285);
_setcolor(AXISCOLOR);
_rectangle(_GBORDER, xorig + 500, yorig + 225, xorig + 595, yorig + 285);

_moveto(xorig + 520, yorig + 235);
_setcolor(GRAPHCOLOR);
_lineto(xorig + 575, yorig + 235);

_moveto(xorig + 510, yorig + 240);
_setcolor(TITLECOLOR);
_lineto(xorig + 585, yorig + 240);
```

```
_setcolor(AXISCOLOR);
for(a = 0, x = 505, y = 250; a < 5; y += 7, a++) {
    _moveto(xorig + x, yorig + y);
    _lineto(xorig + (x + 50), yorig + y);
 }

_setcolor(4);
for(a = 0, x = 560, y = 248; a < 5; y += 7, a++) {
    _ellipse(_GFILLINTERIOR, xorig + x, yorig + y, xorig + (x + 4), yorig + (y + 4));
}

}

/* icon 4 */
{
_setcolor(8);
_rectangle(_GFILLINTERIOR, xorig + 505, yorig + 310, xorig + 600, yorig + 370);
_setcolor(3);
_rectangle(_GFILLINTERIOR, xorig + 500, yorig + 305, xorig + 595, yorig + 365);
_setcolor(AXISCOLOR);
_rectangle(_GBORDER, xorig + 500, yorig + 305, xorig + 595, yorig + 365);

_moveto(xorig + 585, yorig + 355);
_setcolor(AXISCOLOR);
_lineto(xorig + 510, yorig + 355);
_lineto(xorig + 510, yorig + 315);

_setcolor(TITLECOLOR);
for(a = 0, x = 515, y = 334; a < 14; x += 5, a++, y++) {
    _moveto(xorig + x, yorig + y);
    _lineto(xorig + x, yorig + 354);
     }

}


/* icon 5 */
{
_setcolor(8);
_rectangle(_GFILLINTERIOR, xorig + 505, yorig + 390, xorig + 600, yorig + 450);
_setcolor(3);
_rectangle(_GFILLINTERIOR, xorig + 500, yorig + 385, xorig + 595, yorig + 445);
_setcolor(AXISCOLOR);
_rectangle(_GBORDER, xorig + 500, yorig + 385, xorig + 595, yorig + 445);

_moveto(xorig + 510, yorig + 435);
_setcolor(AXISCOLOR);
_lineto(xorig + 585, yorig + 435);
_moveto(xorig + 547, yorig + 435);
_lineto(xorig + 547, yorig + 395);


_setcolor(GRAPHCOLOR);
_ellipse(_GFILLINTERIOR, xorig + 560, yorig + 415, xorig + 567, yorig + 422);

_moveto(xorig + 546, yorig + 434);
_setcolor(TITLECOLOR);
_lineto(xorig + 530, yorig + 425);
_lineto(xorig + 525, yorig + 410);
_lineto(xorig + 535, yorig + 400);
_lineto(xorig + 553, yorig + 397);
_lineto(xorig + 562, yorig + 405);
_lineto(xorig + 563, yorig + 415);
}

} /* End of icons() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void graphold(void) {

char mess[10];   /* output data strings for screen */

/* 1 & 2D option */
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 8, xorig + 50);
_outtext("\x10");                          /* ASCII delta to right character */
_settextposition(yorig + 8, xorig + 52);

yesno();

map1 = ans;

_settextcolor(TITLECOLOR);
if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 8, xorig + 50, " ");    /* Erase delta cursor */


/* 3D option */
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 13, xorig + 50);
_outtext("\x10");                          /* ASCII delta to right character */
_settextposition(yorig + 13, xorig + 52);

if(!map1){                     /* if map1 selected cannot have map3 */
```

```
      yesno();
      map3 = ans;
      }

    else { ans = FALSE;}

_settextcolor(TITLECOLOR);
if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 13, xorig + 50, " ");   /* Erase delta cursor */


/* numerical option */
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 18, xorig + 50);
_outtext("\x10");                       /* ASCII delta to right character */
_settextposition(yorig + 18, xorig + 52);

yesno();

num = ans;

_settextcolor(TITLECOLOR);
if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 18, xorig + 50, " ");   /* Erase delta cursor */


/* component option */
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 23, xorig + 50);
_outtext("\x10");                       /* ASCII delta to right character */
_settextposition(yorig + 23, xorig + 52);

yesno();

comp = ans;

_settextcolor(TITLECOLOR);
if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 23, xorig + 50, " ");   /* Erase delta cursor */


/* total vector option */
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 28, xorig + 50);
_outtext("\x10");                       /* ASCII delta to right character */
_settextposition(yorig + 28, xorig + 52);

yesno();

frq = ans;

_settextcolor(TITLECOLOR);
if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 28, xorig + 50, " ");   /* Erase delta cursor */

Enterkey();

    if( (map1 || map3 || comp || frq) && tra) {
{
_setcolor(8);
_rectangle(_GFILLINTERIOR, XCENTER - 150, YCENTER - 40, XCENTER + 170, YCENTER + 60);
_setcolor(0);
_rectangle(_GFILLINTERIOR, XCENTER - 160, YCENTER - 50, XCENTER + 160, YCENTER + 50);
_setcolor(3);
_rectangle(_GBORDER, XCENTER - 160, YCENTER - 50, XCENTER + 160, YCENTER + 50);

_settextcolor(3);
printat(yorig + 14, xorig + 24, " ENTER EPOCHS FOR GRAPHICS UPDATE...");

_setcolor(GRAPHCOLOR);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 212, xorig + 187, yorig + 219);

do{

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 16, xorig + 38);
_outtext("\x10");
_settextposition(yorig + 16, xorig + 40);
scanf("%ld", &gupdate);
```

```
if(gupdate > epochs) {                          /* Cannot be larger than no of epochs */
              printf("\a");                     /* Alarm */
              _settextposition(yorig + 16, xorig + 40);
              printf("          ");             /* Erase value */
              }

} while(gupdate > epochs);              /* Repeat until acceptable value */

sprintf(mess,"%ld", gupdate);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 16, xorig + 38, " ");

}

_settextcolor(GRAPHCOLOR);
printat(yorig + 18, xorig + 27, " Press Enter to continue .....");

Enterkey();
}

} /* end of graphold() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void processpatts(void) {

/***********************************************
 *   Compute Euclidean Norm for pattern matrix.  *
 *  of form :- P_new = P_old / [E(P_old)^2] ^1/2 *
 ***********************************************/
{
unsigned long  c, i;

double  d = 0.0;
double  a = 0.0;

for(i = 0; i < inpatts; i++) {

    for(c = 0; c < innodes; c++) {
     a = inlayer[i][c];
     d += (a * a);
     }

 d = sqrt(d);

    for(c = 0; c < innodes; c++) {
     inlayer[i][c] /= d;
     }
 d = 0;
}

}
}  /* End of pattern normalisation routine */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Ask for pre recorded weight matrix or set randomized/normalized start */
/* for initial weight matrix */

void preweight(void) {

float r;     /* contents for matrix */
int ok;
unsigned long   i, c;
int input[3];
char mes[45];

{
_settextcolor(GRAPHCOLOR);
printat(22, 21, "Continue training an established network");
printat(23, 21, "  with a pre-saved weight matrix file ?");
printat(24, 21, "             (Y)es or (N)o");

ok = 0;
while(input[0] = getch())
{

    switch(input[0]) {
    case 'y':
    case 'Y': _settextcolor(3);
           printat(yorig + 15, xorig + 5, " FILE NAME FOR PRE-TRAINED WEIGHT MATRIX..... ");
          _ellipse(_GFILLINTERIOR, xorig + 20, yorig + 196, xorig + 27, yorig + 203);
          _settextcolor(GRAPHCOLOR);
          _settextposition(yorig + 15, xorig + 60);
          _outtext("\x10");
          _settextposition(yorig + 15, xorig + 62);
          printat(22, 18, "                                                  ");
          printat(23, 18, "                                                  ");
          printat(24, 18, "                                 ");

   do{

_settextposition(yorig + 15, xorig + 62);
gets(inpathweights);
_settextcolor(TITLECOLOR);
_outtext(inpathweights);

  if( (inweights = fopen( inpathweights, "rb")) == NULL )        /* error message */
```

```
  {
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     sprintf( mes, " Can't open weights file, error number: %d", errno);
     printat(17, 18, mes);
     printat(18, 16, " Drive and/or Directory has to exist to use !");
     printat(20, 20, " Press ENTER to repeat declaration.");

     Enterkey();

     /* Erase error message */
     printat(17, 18, "                                                  ");
     printat(18, 16, "                                                  ");
     printat(20, 20, "                                                  ");
     printat(yorig + 9, xorig + 62, "                      "); /* Erase path */

   )

} while( inweights == NULL);    /* Loop declaration until valid */

_settextcolor(GRAPHCOLOR);
printat(22, 27, "PROCESSING DATA PLEASE WAIT");

for(i = 0; i < outnodes; i++) {
  for(c = 0; c < innodes; c++) {
    fscanf(inweights, "%f,", &r);
    syn_weights[i][c] = r;
      }
   fscanf(inweights, "\n");
}

fclose(inweights);

printat(22, 27, "                            ");

printat(yorig + 9, xorig + 60, " ");  /* Erase delta prompt */

ok = 1;

break;

    case 'n':
    case 'N': printat(22, 18, "                                                  ");
          printat(23, 18, "                                                ");
          printat(24, 18, "                                 ");
          random();
          ok = 1;
          break;

} /* End of switch */

if( ok == 1 ) break;
}
}
} /* End of preweight() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void random(void) {

/* Fill synaptic weight matrix with random floats between 0 - 1 */
/* and perform Euclidean Normalisation. Vector magnitudes lie on unit sphere */


unsigned long   s; /* integer count for total output layer */
unsigned long   i; /* integer count for input layer end */


float r;     /* contents for matrix */
double  d = 0.0;
double  a = 0.0;

{
_settextcolor(GRAPHCOLOR);
printat(20, 25, "RANDOMIZING INITIAL WEIGHT MATRIX");

if(norm) {
      printat(21, 25, "               &");
      printat(22, 29, "COMPUTING EUCLIDEAN NORM");
}

for(s = 0; s < outnodes; s++) {
    for(i = 0; i < innodes; i++) {
        r = (float)rand() / RAND_MAX;
        syn_weights[s][i] = r;
        }
    }


if(norm) {

 /************************************************
 *   Compute Euclidean Norm for weight matrix.    *
 *  of form :- W_new = W_old / [E(W_old)^2] ^1/2  *
 ************************************************/

     for(s = 0; s < outnodes; s++) {

      for(i = 0; i < innodes; i++) {
          a = syn_weights[s][i];
          d += (a * a);                 /* do sum of squared vector components */
        )
```

```
      d = sqrt(d);

      for(i = 0; i < innodes; i++) {
          syn_weights[s][i] /= d;   /* divide components by rooted summed squares */
        }

      d = 0;
      }
 }

printat(20, 25, "                                    ");
printat(21, 25, "                ");
printat(22, 29, "                        ");

}
}  /* End of random() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void memclear(void)
{
     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
            /* FREE ALL MEMORY ALLOCTIONS */

matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));
matrixdel(outlayer, (short)inpatts, (short)outnodes);
matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));

free(pattern_no);
free(previn);
free(prevweight);
free(interlay);
free(interweigh);
free(fire);
free(order);
free(euc_err);

}  /* end of memclear() */

     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*************************************************************************/
              /* GRAPHICS ROUTINES */
/*************************************************************************/

void quadpage(void) {
_setcolor(8);
_rectangle(_GFILLINTERIOR, 11, 11, XCENTER - 1, 235);
_setcolor(0);
_rectangle(_GFILLINTERIOR, 1, 1, XCENTER - 11, 225);
_setcolor(3);
_rectangle(_GBORDER, 1, 1, XCENTER - 11, 225);

_setcolor(8);
_rectangle(_GFILLINTERIOR, XCENTER + 15, 11, 639, 235);
_setcolor(0);
_rectangle(_GFILLINTERIOR, XCENTER + 5, 1, 629, 225);
_setcolor(3);
_rectangle(_GBORDER, XCENTER + 5, 1, 629, 225);

_setcolor(8);
_rectangle(_GFILLINTERIOR, 11, YCENTER + 10, XCENTER - 1, 479);
_setcolor(0);
_rectangle(_GFILLINTERIOR, 1, YCENTER, XCENTER - 11, 469);
_setcolor(3);
_rectangle(_GBORDER, 1, YCENTER, XCENTER - 11, 469);

_setcolor(8);
_rectangle(_GFILLINTERIOR, XCENTER + 15, YCENTER + 10, 639, 479);
_setcolor(0);
_rectangle(_GFILLINTERIOR, XCENTER + 5, YCENTER, 629, 469);
_setcolor(3);
_rectangle(_GBORDER, XCENTER + 5, YCENTER, 629, 469);


} /* end of quadpage() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


void map1graph (void) {

int a, x, y;

_setvieworg(0, 0);

_setcolor(0);
_rectangle(_GFILLINTERIOR, 1, 1, XCENTER - 11, 225);
_setcolor(3);
_rectangle(_GBORDER, 1, 1, XCENTER - 11, 225);

_settextcolor(GRAPHCOLOR);
printat(2, 12, "DIMENSIONAL MAP");

/* draw grid */
_setcolor(3);
```

```
_rectangle(_GFILLINTERIOR,5,50,305,220);

_setcolor(0);
for( a = 0, x = 35; a < 9; a++, x += 30) {
    _moveto(x, 50);
    _lineto(x, 220);
}

for( a = 0, y = 67; a < 9; a++, y += 17) {
    _moveto(5, y);
    _lineto(305, y);
}

_settextcolor(TITLECOLOR);

} /* end of map1graph */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void map3graph (void) {

_setvieworg(0, 0);

_setcolor(0);
_rectangle(_GFILLINTERIOR, 1, 1, XCENTER - 11, 225);

_setcolor(3);
_rectangle(_GBORDER, 1, 1, XCENTER - 11, 225);

_settextcolor(GRAPHCOLOR);
printat(2, 12, "DIMENSIONAL MAP");

_setvieworg(5, 50);

/* draw 3d box */

_setcolor(3);
_moveto(150, 0);
_lineto(150, 100);
_lineto(0, 135);
_lineto(150, 170);
_lineto(300, 135);
_lineto(150, 100);

_moveto(0, 135);
_lineto(0, 35);
_lineto(150, 0);
_lineto(300, 35);
_lineto(300, 135);

_settextcolor(TITLECOLOR);

} /* end of map3graph() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void numap(void) {


_settextcolor(GRAPHCOLOR);                /* title */
printat(2, 55, "NETWORK DATA");

_settextcolor(3);                       /* headings */
printat(4, 42, "TRAINING FILE          =");
printat(5, 42, "INPUT/OUTPUT NODES     =");
printat(6, 42, "PATTERN NUMBER         =");
printat(7, 42, "TOTAL PATTERNS         =");
printat(8, 42, "ITERATIONS DONE        =");
printat(9, 42, "NEIGHBOURHOOD %        =");
printat(10, 42, "LEARNING COEFFICIENT   =");
printat(11, 42, "COEFFICIENT DECREMENT  =");
printat(12, 42, "WINNING NEURON         =");
printat(13, 42, "ERROR BAR              =");

} /* end of numap() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void comap (void) {

int a, y;


_setvieworg(0, 0);                          /* write title */
_settextcolor(GRAPHCOLOR);
printat(17, 12, "COMPONENT ERROR");

_setcolor(8);               /* draw graph */
_rectangle(_GFILLINTERIOR,5,289,305,459);
_setcolor(0);
_rectangle(_GFILLINTERIOR,8,289,305,456);

_setcolor(3);                /* draw grid lines */
for( a = 0, y = 306; a < 9; a++, y += 17) {
    _moveto(5, y);
    _lineto(305, y);
}
_setcolor(3);                /* graph border */
_rectangle(_GBORDER,5,289,305,459);

tog = 1;    /* set toggle to true */
```

```
} /* end of comap () */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void frqmap(void) {

unsigned int x, y, a, b, c;
unsigned int x1, y1;
unsigned int tempx, tempy;
unsigned int tempX, tempY;
char mess[25];

{

_setvieworg(0,0);
_settextcolor(GRAPHCOLOR);                    /* title */
printat(17, 53, "FRQ - ACTIVATION");
_settextcolor(TITLECOLOR);

_settextposition(4, 67);

if(tes){
   if(tesf == 1){
      sprintf(mess, "%s", inpath); /* print running data file name on numeric map space */
    }

   if(tesf > 1){
      sprintf(mess, "%s", &testfiles[tesfno][0]);
    }
   _outtext(mess);
}

_setcolor(0);              /* draw bkgnd rectangle*/
_rectangle(_GFILLINTERIOR, 327, 289, 627, 459);

tempX = 150 / end;
tempY = 35 / end;
tempx = 150 / side; /* available pixels to allocate outlayer nodes */
tempy = 35 / side;

_setcolor(3);         /* draw cyan base grid & vertical axis */

_moveto(477, 289);
_lineto(477, 389);

for(b = 0, x1 = 477, y1 = 389; b < end; b++, x1 -= tempX, y1 += tempY) {
    _moveto(x1, y1);                              /* draw end lines \\\ */
    _lineto(x1 + tempx * (side - 1), y1 + tempy * (side - 1));
    }


for(a = 0, x1 = 477, y1 = 389; a < side; a++, x1 += tempx, y1 += tempy ) {
    _moveto(x1, y1);                              /* draw side lines /// */
    _lineto(x1 - tempX * (end - 1), y1 + tempY * (end - 1));
    }

_setcolor(GRAPHCOLOR);  /* draw red nodes */
for(b = 0, x1 = 477, y1 = 389; b < end; b++, x1 -= tempX, y1 += tempY) { /* increment end */
    for(a = 0, x = x1, y = y1; a < side; a++, x += tempx, y += tempy) {  /* increment side */
      _ellipse(_GFILLINTERIOR, x, y, x + 1, y + 1);
     }
 }

_settextcolor(TITLECOLOR);

}

} /* end of frqmap() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


void mark2dpat(void) {  /* plot and view input distribution */

unsigned long z;
float x, y;

{
    _setvieworg(5, 50);             /* reset origin */


    _setcolor(0);                   /* mark input pattern */
    for(z = 0; z < inpatts; z++) {
    for(n = 0; n < 1; ) {
        x = inlayer[z][n];
        n++;
        y = inlayer[z][n];
      }
      x *= 300;
      y *= 170;
    _ellipse(_GFILLINTERIOR, (int)x - 1, (int)y - 1, (int)x + 1, (int)y + 1);

     }
}
} /* end of mark2dpat() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void mark3dpat(void) {
```

```
float t = 0.23271F; /* tan 13.1 deg */
float xa, xb, ya, yb, Z, X, Y;
unsigned long z;

_setvieworg(5, 50);                 /* reset origin */


_setcolor(11);          /* mark input pattern */
    for(z = 0; z < inpatts; z++) {
    for(n = 0; n < 2; ) {
        xa = inlayer[z][n];
        n++;
        ya = inlayer[z][n];
        n++;
        Z = inlayer[z][n];
     }


     xa *= 150;     /* scale inputs */
     ya *= 150;
     Z *= 100;

     xb = 150 - xa;
     yb = 150 + ya;

     X = yb - xa;
     Y = 100 + (t * ( xa + ya ) - Z) ;


    _ellipse(_GFILLINTERIOR, (int)X - 1, (int)Y - 1, (int)X, (int)Y);

     }

} /* end of mark3dpat() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*************************************************************************/
            /* GRAPH PLOTS */
/*************************************************************************/

void map1plot(void) {

unsigned int a, b, c;
float x, y;

    map1graph();                    /* erase old map */
    _setvieworg(5, 50);             /* reset origin */
    mark2dpat();            /* mark input pattern */


    _setcolor(TITLECOLOR);              /* draw output layer nodes */
    for(o = 0; o < outnodes; o++) {
    for(n = 0; n < 1; ) {
        x = syn_weights[o][n];
        n++;
        y = syn_weights[o][n];
      }
      x *= 300;
      y *= 170;
    _ellipse(_GFILLINTERIOR, (int)x - 2, (int)y - 2, (int)x + 2, (int)y + 2);

     }

/* draw horizantal joint lines */

_setcolor(GRAPHCOLOR);
for(b = 0, c = 0; b < end; b++, c +=side) {
    for(a = c; a < c + side; a++) {
    for(n = 0; n < 1; ) {
        x = syn_weights[a][n];
        n++;
        y = syn_weights[a][n];
      }
      x *= 300;
      y *= 170;

      if(a == c) {
     _moveto((int)x, (int)y);
      }
      _lineto((int)x, (int)y);
    }
}

/* draw vertical joint lines */

for(b = 0; b < side; b++) {
    for(a = b, c = 0; c < end; a += side, c++) {
    for(n = 0; n < 1; ) {
        x = syn_weights[a][n];
        n++;
        y = syn_weights[a][n];
      }
      x *= 300;
      y *= 170;

      if(a == b) {
     _moveto((int)x, (int)y);
      }
      _lineto((int)x, (int)y);
    }
}
```

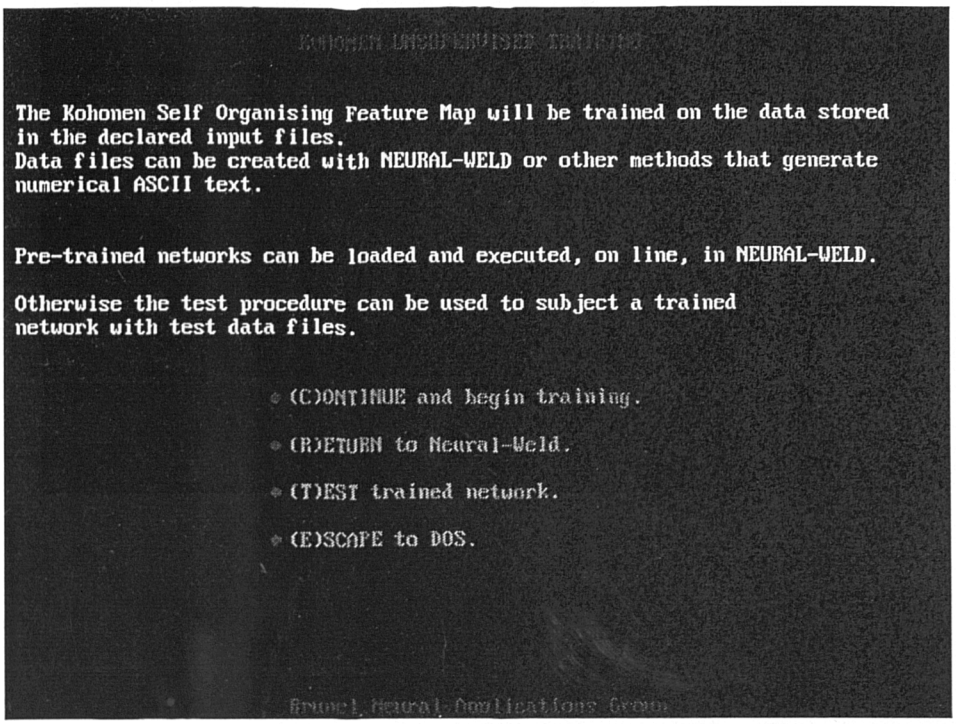
```
} /* end of map1plot() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


void map3plot(void) {

float t = 0.23271F; /* tan 13.1 deg */
float xa, xb, ya, yb, Z, X, Y;
unsigned int  a, b, c;

map3graph();         /* erase old neurons by redrawing bkgnd */
mark3dpat();         /* mark input pattern */

    _setcolor(TITLECOLOR);              /* draw output layer nodes */
    for(o = 0; o < outnodes; o++) {
     for(n = 0; n < 2; ) {
        xa = syn_weights[o][n];
        n++;
        ya = syn_weights[o][n];
        n++;
        Z =  syn_weights[o][n];
     }

     xa *= 150;     /* scale inputs */
     ya *= 150;
     Z *= 100;

     xb = 150 - xa;
     yb = 150 + ya;

     X = yb - xa;
     Y = 100 + (t * ( xa + ya ) - Z) ;


    _ellipse(_GFILLINTERIOR, (int)X - 2, (int)Y - 2, (int)X + 2, (int)Y + 2);

     }

/* draw horizantal joint lines */

_setcolor(GRAPHCOLOR);
for(b = 0, c = 0; b < end; b++, c +=side) {
    for(a = c; a < c + side; a++) {
    for(n = 0; n < 2; ) {
        xa = syn_weights[a][n];
        n++;
        ya = syn_weights[a][n];
        n++;
        Z =  syn_weights[a][n];
     }

     xa *= 150;     /* scale inputs */
     ya *= 150;
     Z *= 100;

     xb = 150 - xa;
     yb = 150 + ya;

     X = yb - xa;
     Y = 100 + (t * ( xa + ya ) - Z) ;

      if(a == c) {
     _moveto((int)X, (int)Y);
      }
      _lineto((int)X, (int)Y);
    }
}

/* draw vertical joint lines */

for(b = 0; b < side; b++) {
    for(a = b, c = 0; c < end; a += side, c++) {
    for(n = 0; n < 2; ) {
        xa = syn_weights[a][n];
        n++;
        ya = syn_weights[a][n];
        n++;
        Z =  syn_weights[a][n];
      }

     xa *= 150;     /* scale inputs */
     ya *= 150;
     Z *= 100;

     xb = 150 - xa;
     yb = 150 + ya;

     X = yb - xa;
     Y = 100 + (t * ( xa + ya ) - Z) ;

      if(a == b) {
     _moveto((int)X, (int)Y);
      }
      _lineto((int)X, (int)Y);
    }
}


} /* end of map3plot() */
```

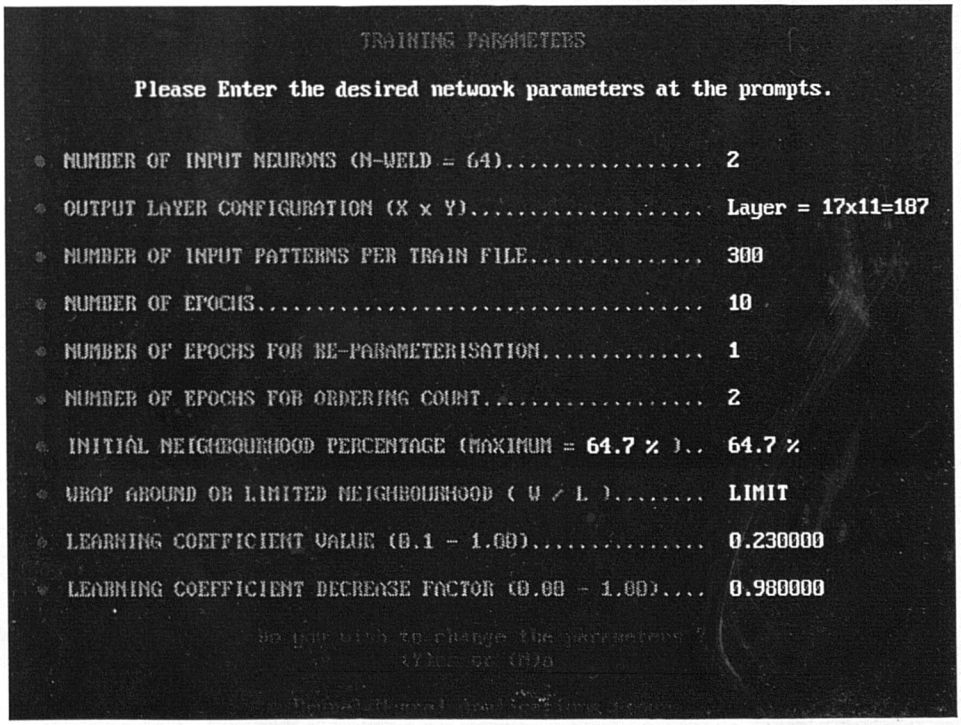
```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void complot(void) {

char dat[12];

float *inlayerptr, *syn_weightsptr, *previnptr, *prevweightptr;
float *interlayptr, *interweighptr;
float templ, temp2;

int x, y, x2;
unsigned long f, a;

{

y = 167;                    /* available 'Y' pixels on graph */

inlayerptr = inlayer[p];              /* set pointers to data arrays */
syn_weightsptr = syn_weights[win];

interlayptr = interlay;
interweighptr = interweigh;

for(f = 0; f < innodes; f++) {        /* copy data arrays for erase */
    templ = (*inlayerptr++);
    temp2 = (*syn_weightsptr++);
    *interlayptr++ = templ * y;
    *interweighptr++ = temp2 * y;
    }

interlayptr = interlay;               /* pont to current data */
interweighptr = interweigh;

previnptr = previn;                   /* point to previous data */
prevweightptr = prevweight;


/* plot data */
_setvieworg(8, 289);


x2 = (297 / (short)innodes) - 2;        /* width of column */

   for(n = 0, x = 1; n < innodes; n++, x += 297 / innodes) {

     if(tog) {                         /* swap synaptic pattern with data pattern */
       _setcolor(0);                   /* on graphics screen */
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*prevweightptr);

       _setcolor(0);
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*previnptr);

       *(previnptr++) = *interlayptr;                /* set to current data array */
       *(prevweightptr++) = *interweighptr;

    _setcolor(TITLECOLOR);
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*(interweighptr)++);

       _setcolor(GRAPHCOLOR);
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*(interlayptr)++);
     }
     else {
       _setcolor(0);
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*previnptr);

       _setcolor(0);
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*prevweightptr);

       *(previnptr++) = *interlayptr;                /* set to current data array */
       *(prevweightptr++) = *interweighptr;

    _setcolor(GRAPHCOLOR);
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*(interlayptr)++);

       _setcolor(TITLECOLOR);
       _rectangle(_GFILLINTERIOR, x, y, x + x2, y - (int)*(interweighptr)++);
     }
    }

_setvieworg(0, 0);
_setcolor(3);                 /* redraw grid lines */
for( a = 0, y = 306; a < 9; a++, y += 17) {
    _moveto(5, y);
    _lineto(305, y);

printat(18, 8, "Euclidean Error=");       /* print vector Euclidean error */
_settextposition(18, 25);
_settextcolor(TITLECOLOR);
sprintf(dat, "%f", Small_Euc_Dist);
_outtext(dat);

}

}

} /* end of complot() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void frqplot(void) {
```

```
int winend, winside;
int x, y, a, b, c, z;
int x1, y1;
int tempx, tempy;
int tempX, tempY;

{

x1 = 477; /* starting pixel */
y1 = 389;

tempX = 150 / end;
tempY = 35 / end;
tempx = 150 / side; /* available pixels to allocate outlayer nodes */
tempy = 35 / side;

if(win < side) {          /* locate active neuron within output slab */
    winside = win;
    winend = 0;
    }
else {
    winside = win % side;
    winend = win / side;
    }

if(tes) {
     order[p] = win; /* list order of firing neurons if test proc only */
   }

fire[win] += 2; /* increment frequency vector by step of 2*/

z = (int)fire[win];     /* y coordinate change factor */

x1 -= winend * tempX; /* translate position in slab to pixel coordinates */
y1 += winend * tempY;
x = x1;
y = y1;
x += winside * tempx;
y += winside * tempy;

_setvieworg(0,0);

_setcolor(14);    /* change winning neuron to yellow & move up */
_ellipse(_GFILLINTERIOR, x, y - z, x + 1, y - z + 1);

}

} /* end of frqplot() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void frqtidy(void) {

unsigned int x, y, a, b, c, c1, z;
unsigned int x1, y1;
unsigned int tempx, tempy;
unsigned int tempX, tempY;

{
 frqmap();    /* refresh frq map start grid */

tempX = 150 / end;
tempY = 35 / end;
tempx = 150 / side; /* available pixels to allocate outlayer nodes */
tempy = 35 / side;

/* draw mesh by joining displaced frequency nodes */
_setvieworg(0,0);
_setcolor(14);  /* yellow lines */

for(b = 0, c = 0, x1 = 477, y1 = 389; b < end; b++, x1 -= tempX, y1 += tempY) {
    z = (int)fire[c]; /* start offset for line draw */
    _moveto(x1, y1 - z);                              /* draw end lines \\\ */
    for(a = 0, x = x1, y = y1; a < side; a++, c++, x += tempx, y += tempy) {    /* increment side */
    z = (int)fire[c];
    _lineto(x, y - z);
    }
  }

for(a = 0, c = 0, x1 = 477, y1 = 389; a < side; a++, c++, x1 += tempx, y1 += tempy) {
    z = (int)fire[c];
    _moveto(x1, y1 - z);                              /* draw end lines /// */
    for(b = 0, c1 = c + side, x = x1 - tempX, y = y1 + tempY; b < end - 1; b++, c1 += side,
    x -= tempX,y += tempY) {
    z = (int)fire[c1];
    _lineto(x, y - z);
    }
  }

_setcolor(GRAPHCOLOR);  /* red nodes */
for(b = 0, c = 0, x1 = 477, y1 = 389; b < end; b++, x1 -= tempX, y1 += tempY) { /* increment end */
    for(a = 0, x = x1, y = y1; a < side; a++, c++, x += tempx, y += tempy) {  /* increment side */
      z = (int)fire[c];
      _ellipse(_GFILLINTERIOR, x, y - z, x + 1, y - z + 1);
     }
 }

}

} /* end of frqtidy() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void frqout(void) {

unsigned int a, b, c;


char drive[_MAX_DRIVE], dir[_MAX_DIR];
char fname[_MAX_FNAME], ext[_MAX_EXT];

{
if(tesf == 1){
   strcpy(outpathfrq, inpath);
 }
if(tesf > 1){
   strcpy(outpathfrq, &testfiles[tesfno][0]);
 }

_splitpath( outpathfrq, drive, dir, fname, ext);
strcpy(ext, "nwf" );
_makepath( outpathfrq, drive, dir, fname, ext);

outfrq = fopen( outpathfrq, "wb");

fprintf( outfrq, "%s\n%ld\n", inpathweights, inpatts);

for( a = 0; a < outnodes; a++) {
    fprintf( outfrq, "%d ", (int)fire[a]);
  }

fclose(outfrq);   /* close 'written to' file */

_splitpath( outpathfrq, drive, dir, fname, ext); /* create and write matrix file */
strcpy(ext, "mtx" );
_makepath( outpathfrq, drive, dir, fname, ext);

outfrq = fopen( outpathfrq, "wb");

for(a=0, c=0; a<end; a++) {
    for(b=0; b<side; b++, c++) {
    fprintf( outfrq, "%d ", (int)fire[c]);
    }
    fprintf( outfrq, "\n");
    }

fclose(outfrq);   /* close 'written to' file */

_splitpath( outpathfrq, drive, dir, fname, ext); /* create and write 3D magnitude file */
strcpy(ext, "mag" );
_makepath( outpathfrq, drive, dir, fname, ext);

outfrq = fopen( outpathfrq, "wb");

for(a=0, c=0; a<end; a++) {
    for(b=0; b<side; b++, c++) {
    fprintf( outfrq, "%d,%d,%d,\n", a, b, (int)fire[c]);
    }
    }

fclose(outfrq);   /* close 'written to' file */

_splitpath( outpathfrq, drive, dir, fname, ext); /* create and write firing order file */
strcpy(ext, "ord" );
_makepath( outpathfrq, drive, dir, fname, ext);

outfrq = fopen( outpathfrq, "wb");

for( a = 0; a < inpatts; a++) {
    fprintf( outfrq, "%d,%d\n", a, (int)order[a]);
  }

fclose(outfrq);   /* close 'written to' file */

} /* end of procedure */

} /* end of frqout() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*     Wait for an enter key press         */

void Enterkey(void)
 {

int  input[3];
int  ok;

ok = FALSE;

   while(input[0] = getch()) {     /*  Do nothing until user hits enter    */
   switch(input[0]) {
    case '\x0d' : ok = TRUE;
          break;
            }
 if (ok == TRUE) break;

}
}  /*   End Enterkey() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
 /* Yes or No switch */

void yesno(void) {

int ok;
int input[1];

ok = FALSE;
    while(input[0] = getch())
       {

        switch(input[0]) {
           case 'y':
           case 'Y':ans = TRUE;
                ok = TRUE;
                break;

           case 'n':
           case 'N':ans = FALSE;
                ok = TRUE;
                break;
        }
     if( ok == TRUE) break;
    }

} /* end of yesno */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Set PC overscan register  */
void setborder(short color)
{
    union REGS regs;

    regs.x.ax = 0x01001;    /* Overscan register */
    regs.h.bh = color;
    int86(0x10, &regs, &regs);
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*********************************- FIN -************************************/
```


The Kohonen Self Organising Feature Map will be trained on the data stored in the declared input files.
Data files can be created with NEURAL-WELD or other methods that generate numerical ASCII text.
Pre-trained networks can be loaded and executed, on line, in NEURAL-WELD.
Otherwise the test procedure can be used to subject a trained network with test data files.
(C)ONTINUE and begin training.
(R)ETURN to Neural-Weld.
(T)EST trained network.
(E)SCAPE to DOS.


TRAINING PARAMETERS
Please Enter the desired network parameters at the prompts.
NUMBER OF INPUT NEURONS (N-WELD = 64)..... 2
OUTPUT LAYER CONFIGURATION (X x Y)..... Layer = 17x11=187
NUMBER OF INPUT PATTERNS PER TRAIN FILE..... 300
NUMBER OF EPOCHS..... 10
NUMBER OF EPOCHS FOR RE-PARAMETERISATION..... 1
NUMBER OF EPOCHS FOR ORDERING COUNT..... 2
INITIAL NEIGHBOURHOOD PERCENTAGE (MAXIMUM = 64.7 %).. 64.7 %
WRAP AROUND OR LIMITED NEIGHBOURHOOD (W / L)..... LIMIT
LEARNING COEFFICIENT VALUE (0.1 - 1.00)..... 0.230000
LEARNING COEFFICIENT DECREASE FACTOR (0.00 - 1.00).... 0.980000

FILE DECLARATIONS

Please Enter the desired file names & paths at the prompts.
Extentions are awarded automatically for OUTPUT files.

NUMBER OF INPUT PATTERN TRAIN FILES 1

EUCLIDEAN NORMALISATION REQUIRED (Y/N) ..... NO

FILE TO STORE SYNAPTIC WEIGHTS (*.nww automatic)..... 2d_300c.nww

FILE NAME OF TRAIN PATTERNS [DIMS EXPECTED = 2]..... 2d_300c.nwd

Continue training an established network
with a pre-saved weight matrix file ?
(Y)es or (N)o

Brunel Neural Applications Group

GRAPHICS OPTIONS

NOTE : Any graphics choice will increase the training time.

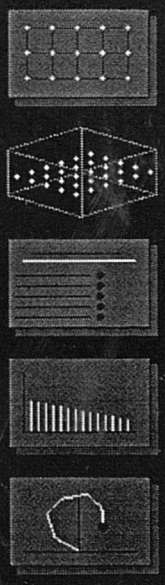
MAP FOR 1 OR 2 DIMENSIONAL DATA (Y/N)..... YES

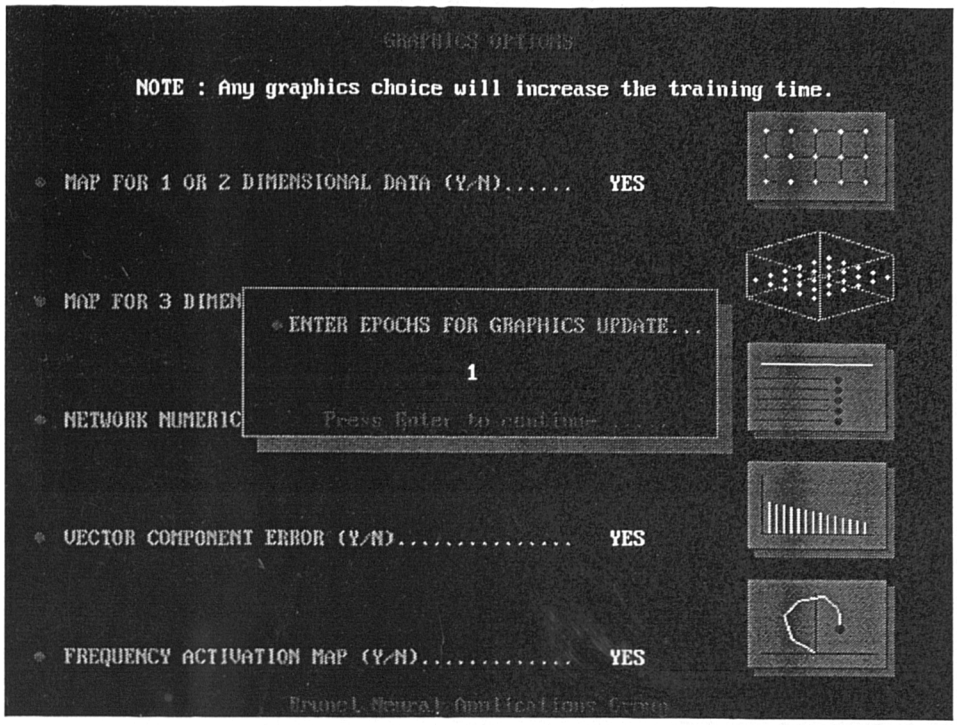
MAP FOR 3 DIMENSIONAL DATA (Y/N)..... NO

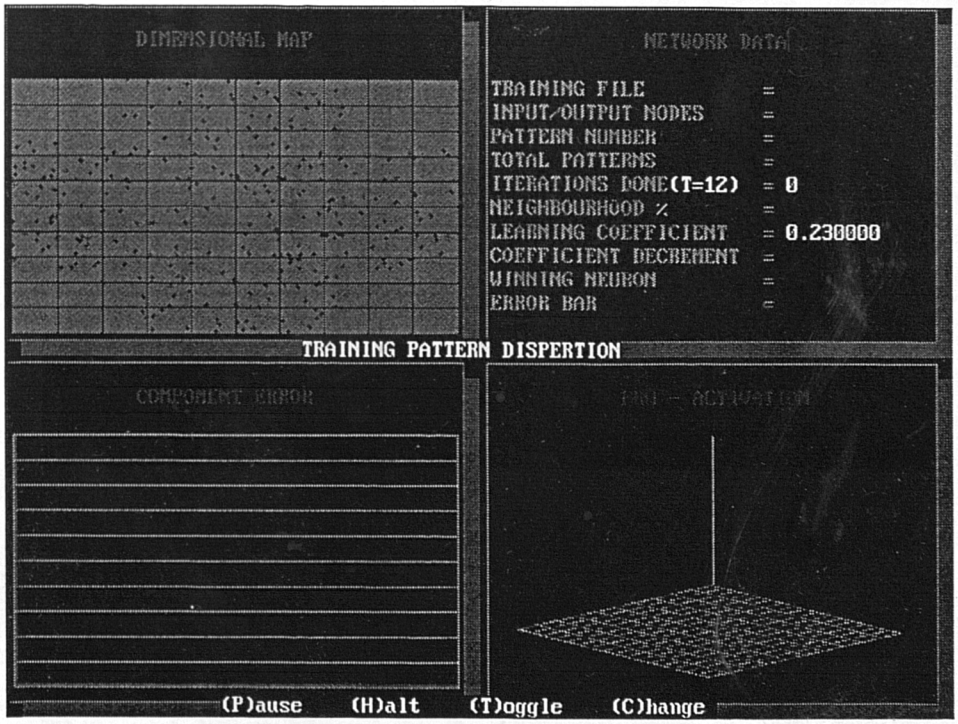
NETWORK NUMERICAL OUTPUT (Y/N)..... YES

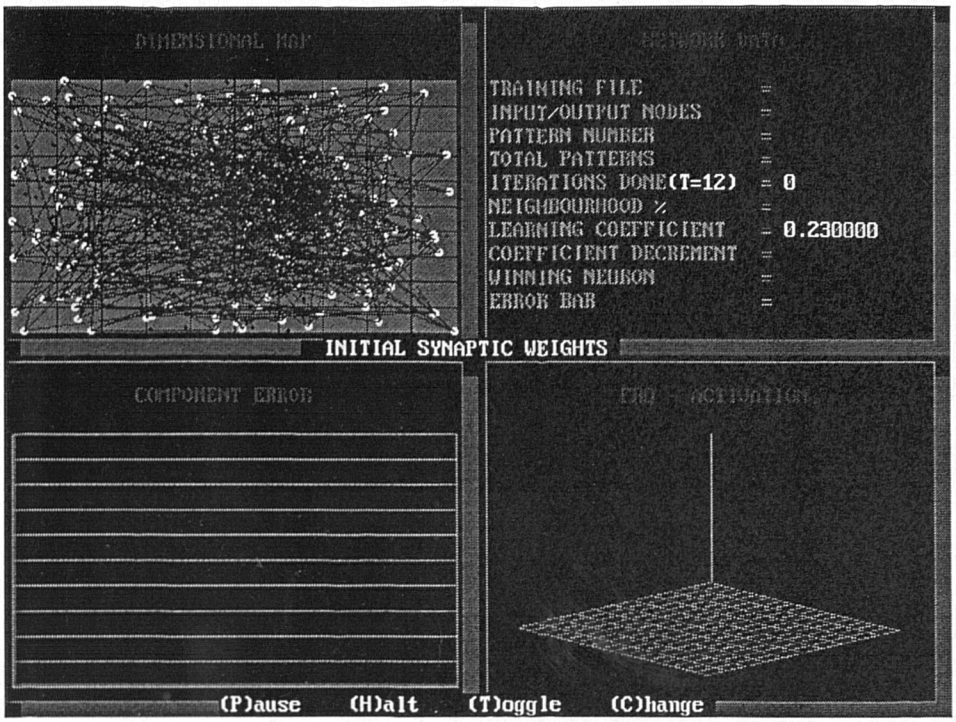
VECTOR COMPONENT ERROR (Y/N)..... YES

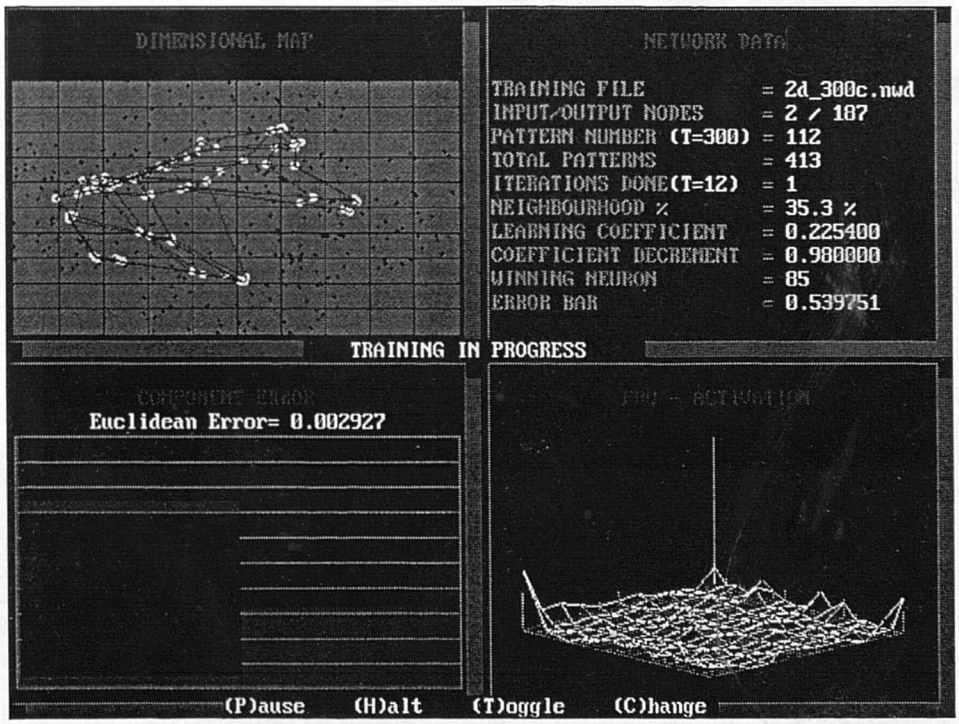
FREQUENCY ACTIVATION MAP (Y/N)..... YES

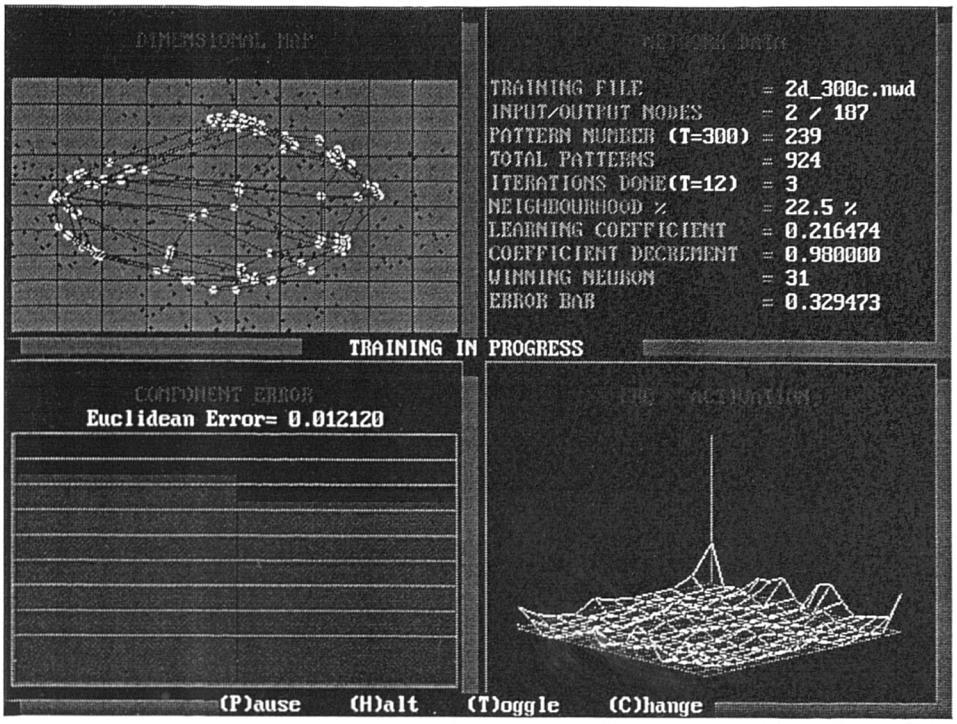


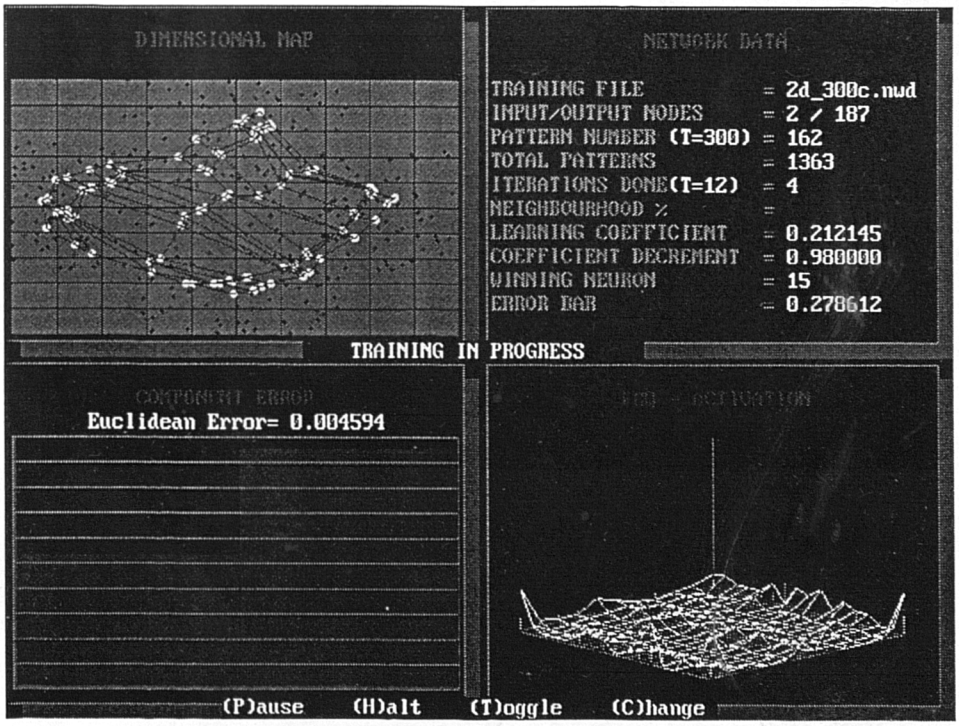

GRAPHICS OPTIONS
NOTE : Any graphics choice will increase the training time.
MAP FOR 1 OR 2 DIMENSIONAL DATA (Y/N)..... YES
MAP FOR 3 DIMEN
ENTER EPOCHS FOR GRAPHICS UPDATE...
1
NETWORK NUMERIC
VECTOR COMPONENT ERROR (Y/N)..... YES
FREQUENCY ACTIVATION MAP (Y/N)..... YES
Press Enter to continue
Kronel Neural Applications Group

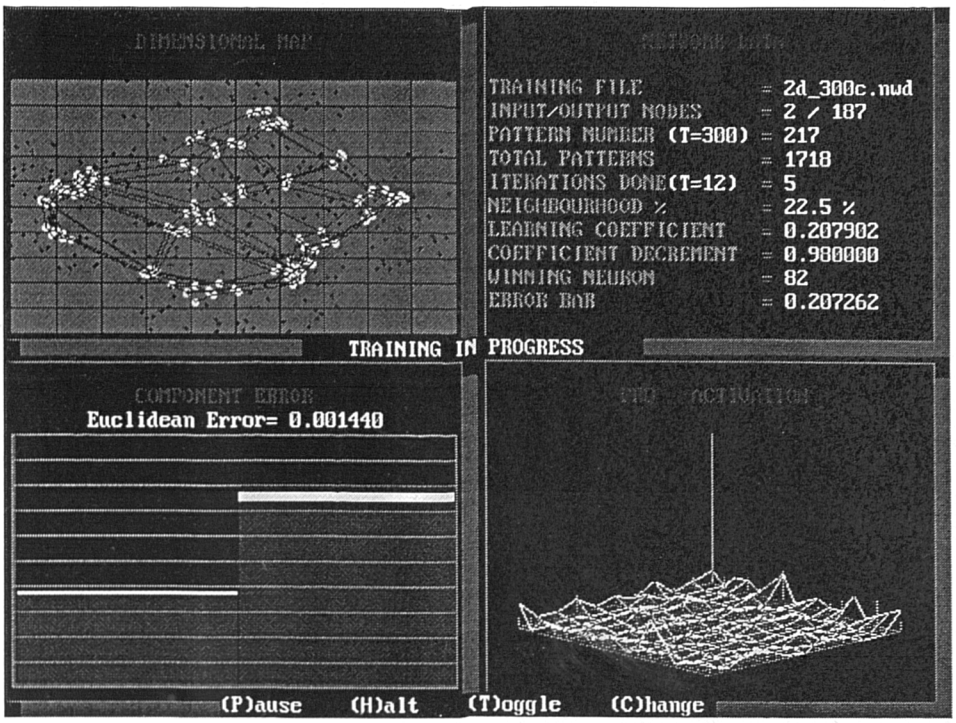

DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE =
INPUT/OUTPUT NODES =
PATTERN NUMBER =
TOTAL PATTERNS =
ITERATIONS DONE(T=12) = 0
NEIGHBOURHOOD % =
LEARNING COEFFICIENT = 0.230000
COEFFICIENT DECREMENT =
WINNING NEURON =
ERROR BAR =
TRAINING PATTERN DISPERTION
COMPONENT ERROR
FREQ - ACTIVATION
(P)ause (H)alt (T)oggle (C)hange

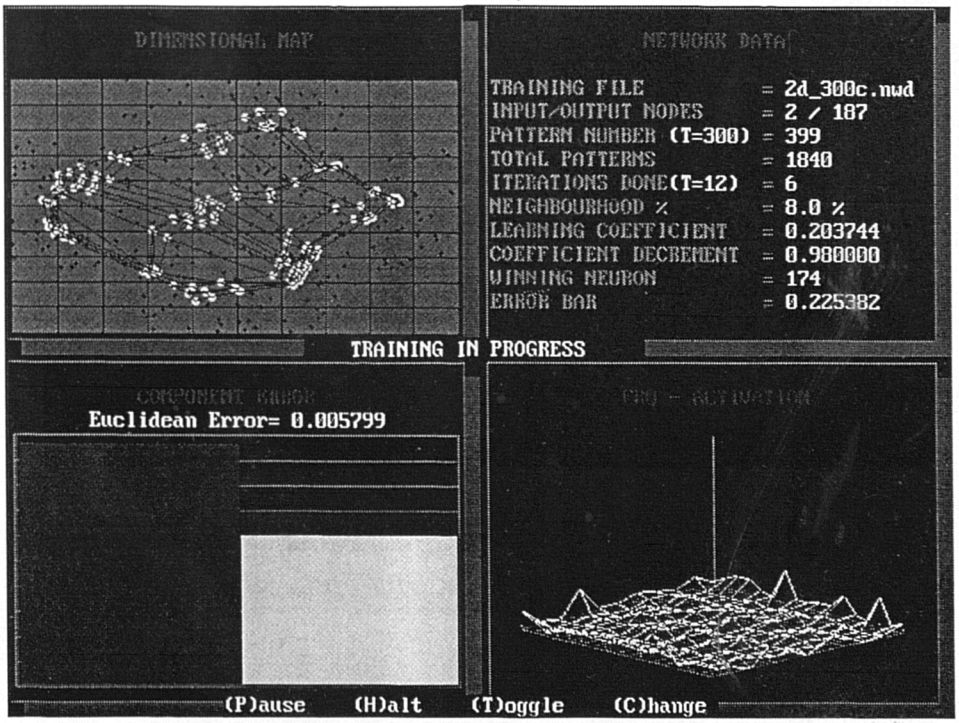

DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE =
INPUT/OUTPUT NODES =
PATTERN NUMBER =
TOTAL PATTERNS =
ITERATIONS DONE(T=12) = 0
NEIGHBOURHOOD % =
LEARNING COEFFICIENT = 0.230000
COEFFICIENT DECREMENT =
WINNING NEURON =
ERROR BAR =
INITIAL SYNAPTIC WEIGHTS
COMPONENT ERROR
FIRE - ACTIVATION
(P)ause (H)alt (T)oggle (C)hange

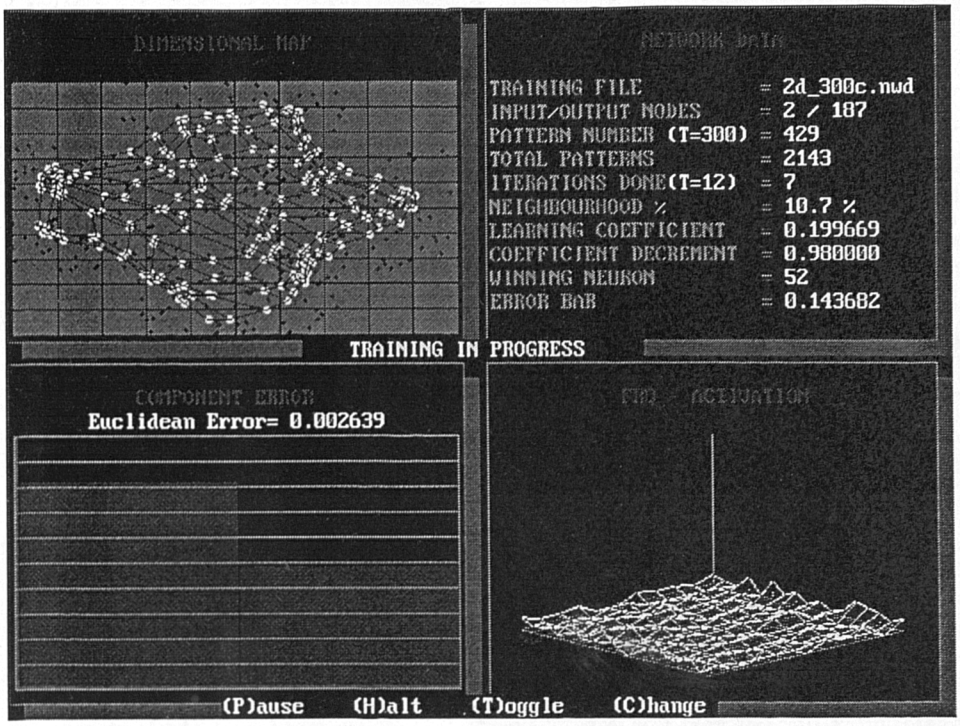

DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 112
TOTAL PATTERNS = 413
ITERATIONS DONE(T=12) = 1
NEIGHBOURHOOD % = 35.3 %
LEARNING COEFFICIENT = 0.225400
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON = 85
ERROR BAR = 0.539751
TRAINING IN PROGRESS
COMPONENT ERROR
Euclidean Error= 0.002927
FIRE - ACTIVATION
(P)ause (H)alt (T)oggle (C)hange

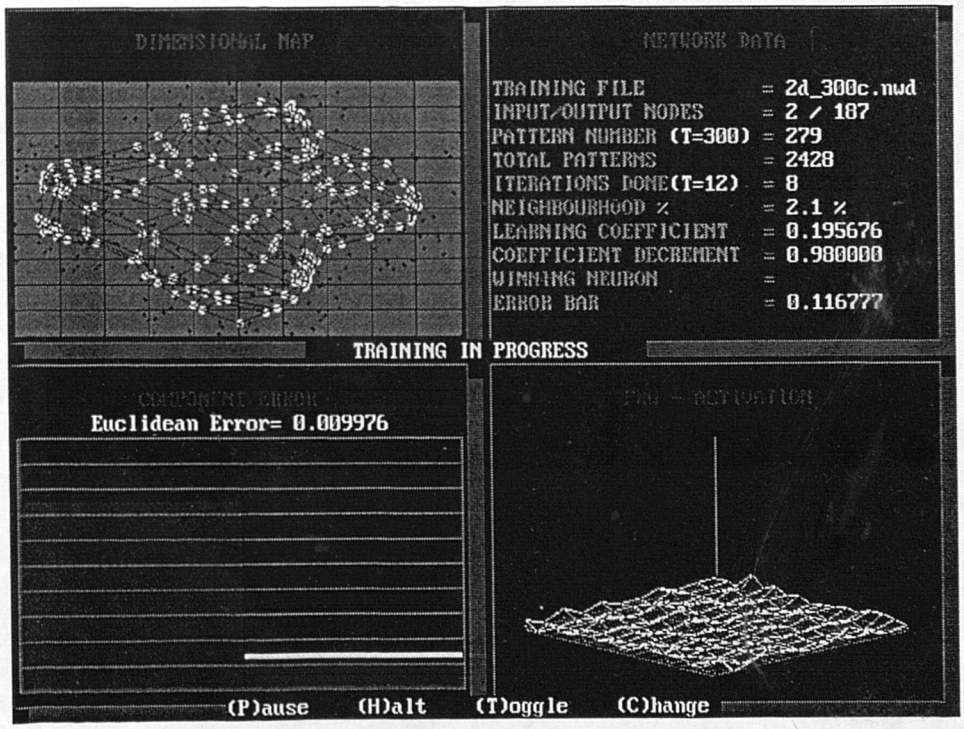

TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 239
TOTAL PATTERNS = 924
ITERATIONS DONE(T=12) = 3
NEIGHBOURHOOD % = 22.5 %
LEARNING COEFFICIENT = 0.216474
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON = 31
ERROR BAR = 0.329473
TRAINING IN PROGRESS
Euclidean Error= 0.012120
(P)ause (H)alt (T)oggle (C)hange

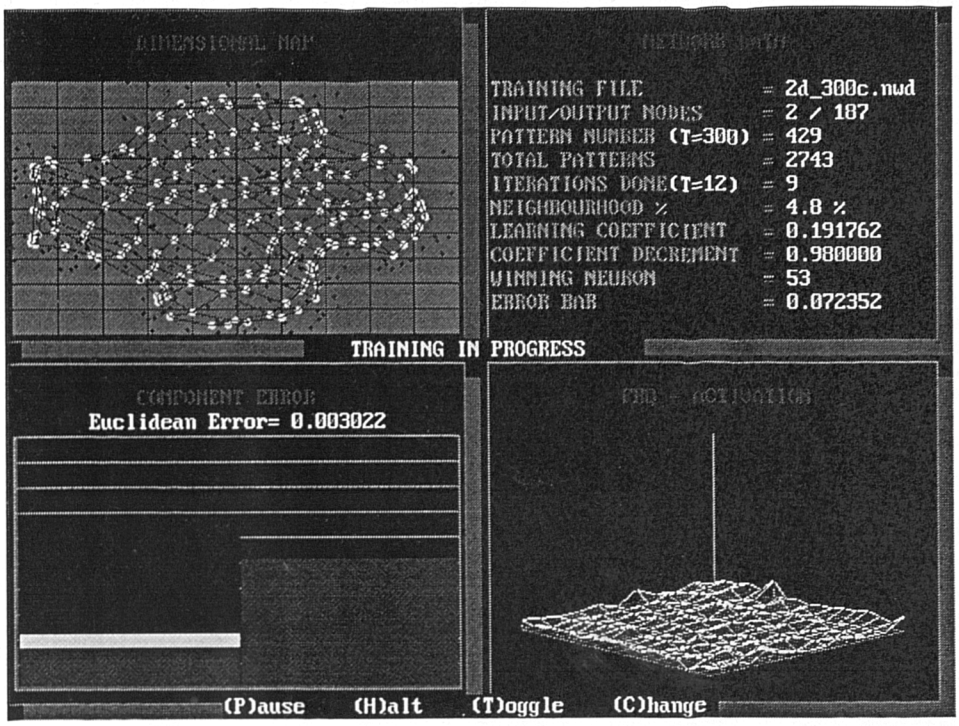

TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 162
TOTAL PATTERNS = 1363
ITERATIONS DONE(T=12) = 4
NEIGHBOURHOOD % =
LEARNING COEFFICIENT = 0.212145
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON = 15
ERROR BAR = 0.278612
TRAINING IN PROGRESS
Euclidean Error= 0.004594
(P)ause (H)alt (T)oggle (C)hange

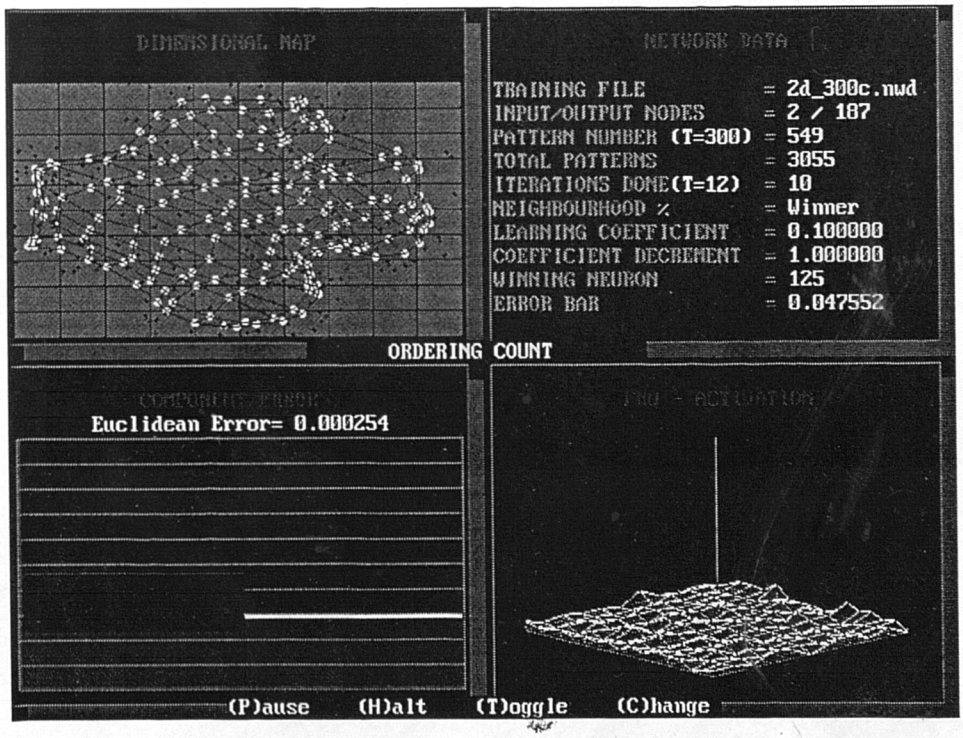

DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 217
TOTAL PATTERNS = 1718
ITERATIONS DONE(T=12) = 5
NEIGHBOURHOOD % = 22.5 %
LEARNING COEFFICIENT = 0.207902
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON = 82
ERROR BAR = 0.207262
TRAINING IN PROGRESS
COMPONENT ERROR
Euclidean Error= 0.001440
(P)ause (H)alt (T)oggle (C)hange

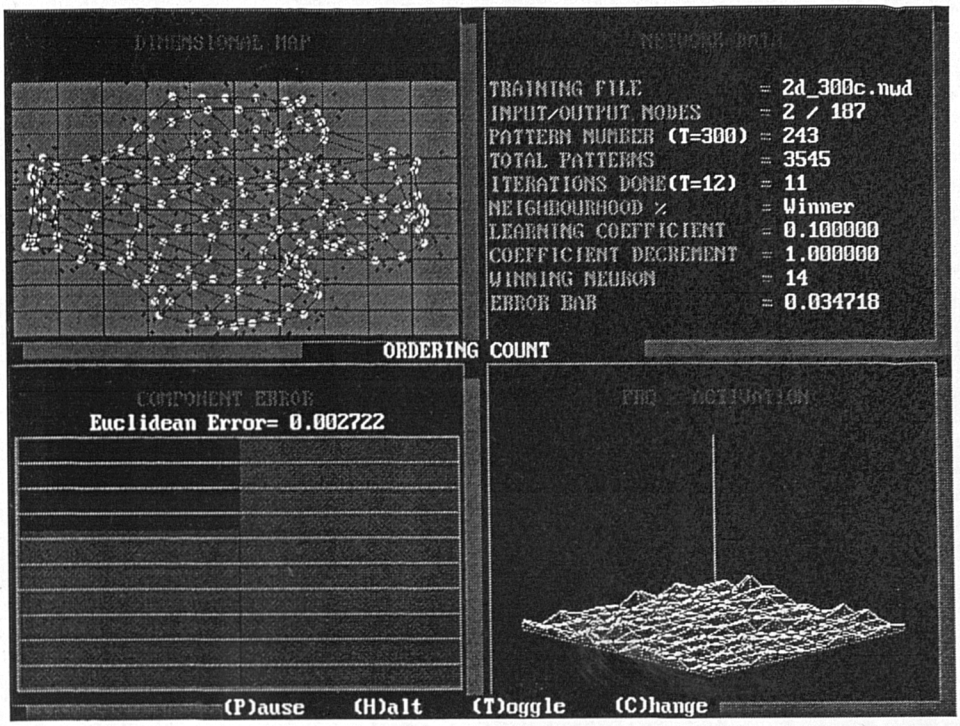

DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 399
TOTAL PATTERNS = 1840
ITERATIONS DONE(T=12) = 6
NEIGHBOURHOOD % = 8.0 %
LEARNING COEFFICIENT = 0.203744
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON = 174
ERROR BAR = 0.225382
TRAINING IN PROGRESS
COMPONENT ERROR
Euclidean Error= 0.005799
(P)ause (H)alt (T)oggle (C)hange

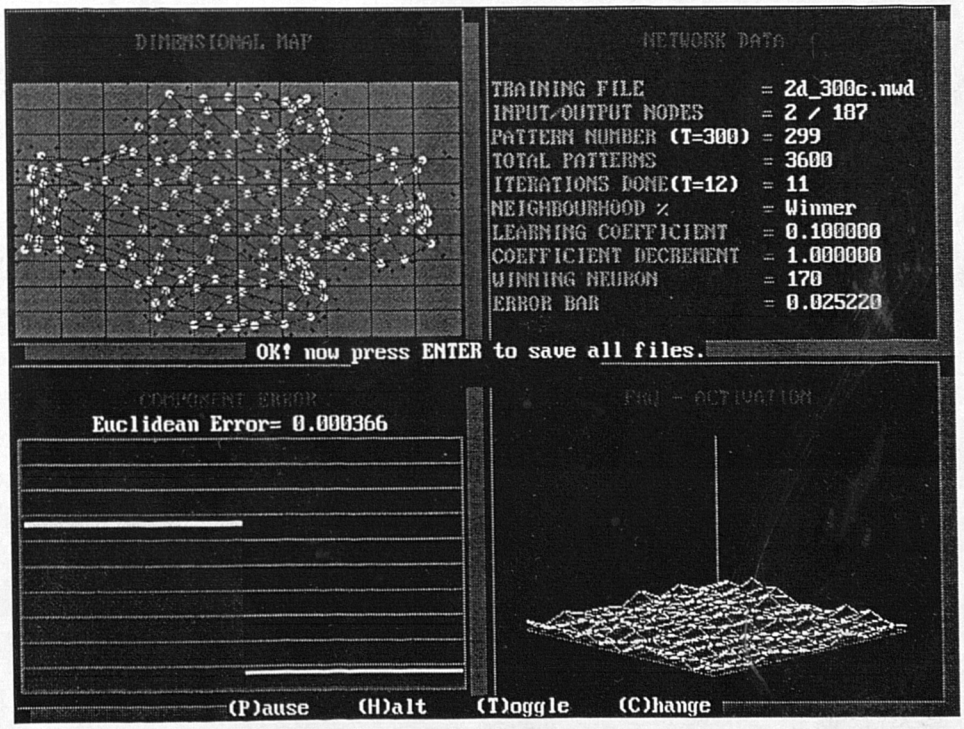

DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 429
TOTAL PATTERNS = 2143
ITERATIONS DONE(T=12) = 7
NEIGHBOURHOOD % = 10.7 %
LEARNING COEFFICIENT = 0.199669
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON = 52
ERROR BAR = 0.143682
TRAINING IN PROGRESS
COMPONENT ERROR
Euclidean Error= 0.002639
FIRE ACTIVATION
(P)ause (H)alt (T)oggle (C)hange


DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 279
TOTAL PATTERNS = 2428
ITERATIONS DONE(T=12) = 8
NEIGHBOURHOOD % = 2.1 %
LEARNING COEFFICIENT = 0.195676
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON =
ERROR BAR = 0.116777
TRAINING IN PROGRESS
COMPONENT ERROR
Euclidean Error= 0.009976
FIRE ACTIVATION
(P)ause (H)alt (T)oggle (C)hange


TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 429
TOTAL PATTERNS = 2743
ITERATIONS DONE(T=12) = 9
NEIGHBOURHOOD % = 4.8 %
LEARNING COEFFICIENT = 0.191762
COEFFICIENT DECREMENT = 0.980000
WINNING NEURON = 53
ERROR BAR = 0.072352
TRAINING IN PROGRESS
Euclidean Error= 0.003022
(P)ause (H)alt (T)oggle (C)hange


NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 549
TOTAL PATTERNS = 3055
ITERATIONS DONE(T=12) = 10
NEIGHBOURHOOD % = Winner
LEARNING COEFFICIENT = 0.100000
COEFFICIENT DECREMENT = 1.000000
WINNING NEURON = 125
ERROR BAR = 0.047552
ORDERING COUNT
Euclidean Error= 0.000254
(P)ause (H)alt (T)oggle (C)hange


DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 243
TOTAL PATTERNS = 3545
ITERATIONS DONE(T=12) = 11
NEIGHBOURHOOD % = Winner
LEARNING COEFFICIENT = 0.100000
COEFFICIENT DECREMENT = 1.000000
WINNING NEURON = 14
ERROR BAR = 0.034718
ORDERING COUNT
COMPONENT ERROR
Euclidean Error= 0.002722
FNU - ACTIVATION
(P)ause (H)alt (T)oggle (C)hange


DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d_300c.nwd
INPUT/OUTPUT NODES = 2 / 187
PATTERN NUMBER (T=300) = 299
TOTAL PATTERNS = 3600
ITERATIONS DONE(T=12) = 11
NEIGHBOURHOOD % = Winner
LEARNING COEFFICIENT = 0.100000
COEFFICIENT DECREMENT = 1.000000
WINNING NEURON = 170
ERROR BAR = 0.025220
OK! now press ENTER to save all files.
COMPONENT ERROR
Euclidean Error= 0.000366
FNU - ACTIVATION
(P)ause (H)alt (T)oggle (C)hange

FILE DECLARATIONS

N.B. ENSURE THE TEST DATA AND NETWORK MATRIX ARE OF THE SAME DIMENSIONALITY !

- FILE NAME FOR PRE-TRAINED WEIGHT MATRIX (?.nww)..... 2d_300c.nww
- NORMALISATION REQUIRED ON TEST PATTERNS ? (Y/N) ..... NO
- NUMBER OF INPUT TEST PATTERNS PER FILE..... 300
- NUMBER OF DATA FILES TO TEST 1
- FILE NAME OF TEST PATTERNS [DIMS EXPECTED = 2]..... 2d300cb.nwd

FILE OPTIONS ACCEPTED

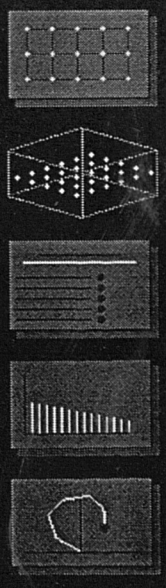
Press ENTER for graphics options.

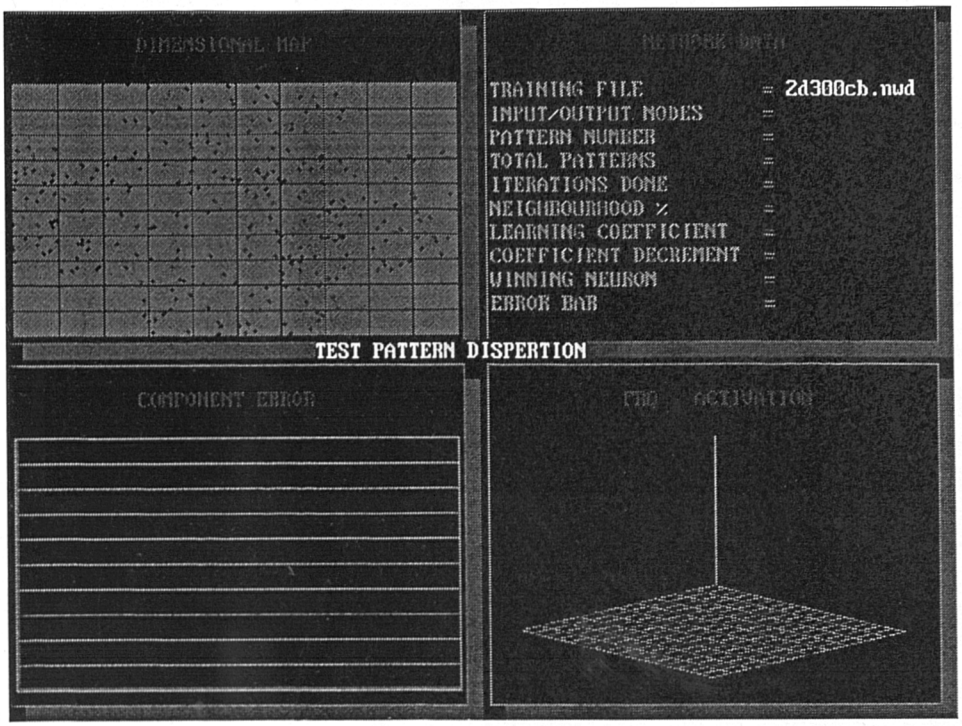
Brunel Neural Applications Group

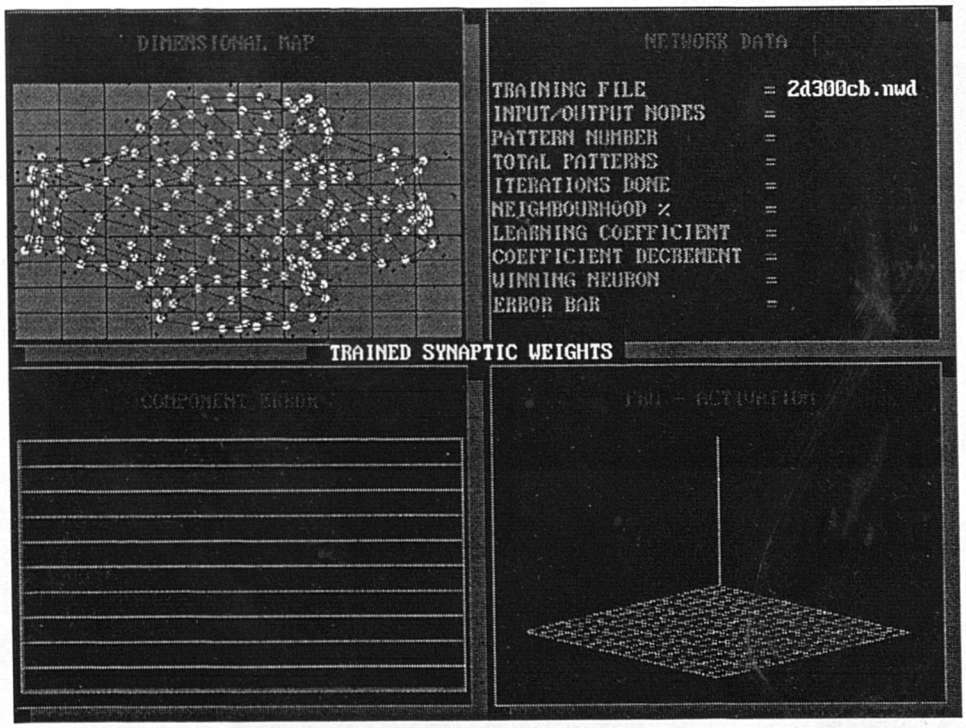
GRAPHICS OPTIONS

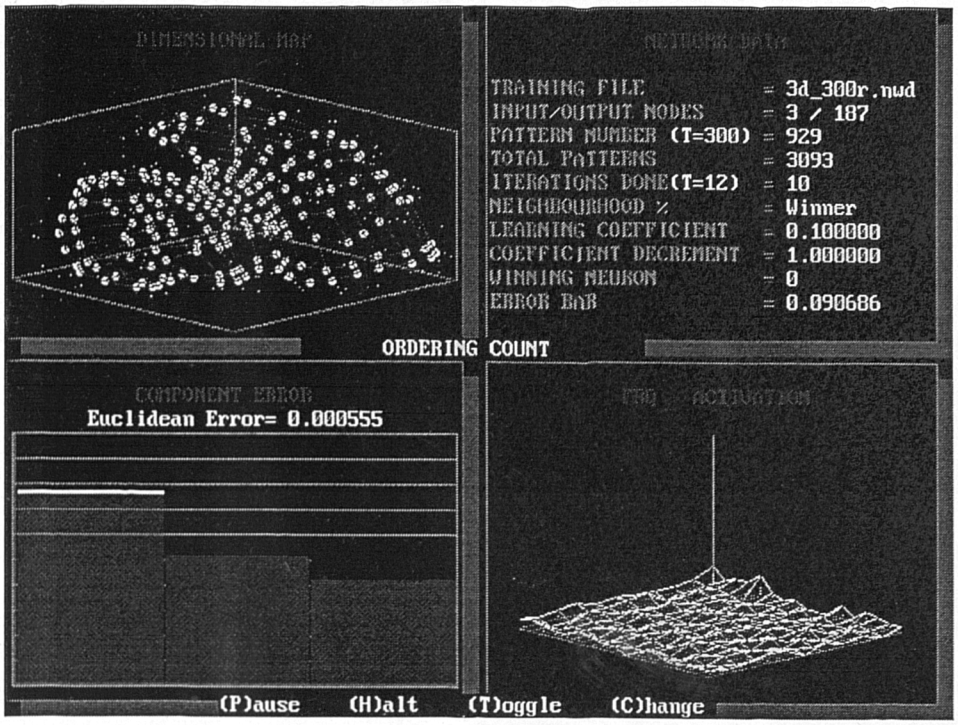
NOTE : Any graphics choice will increase the training time.

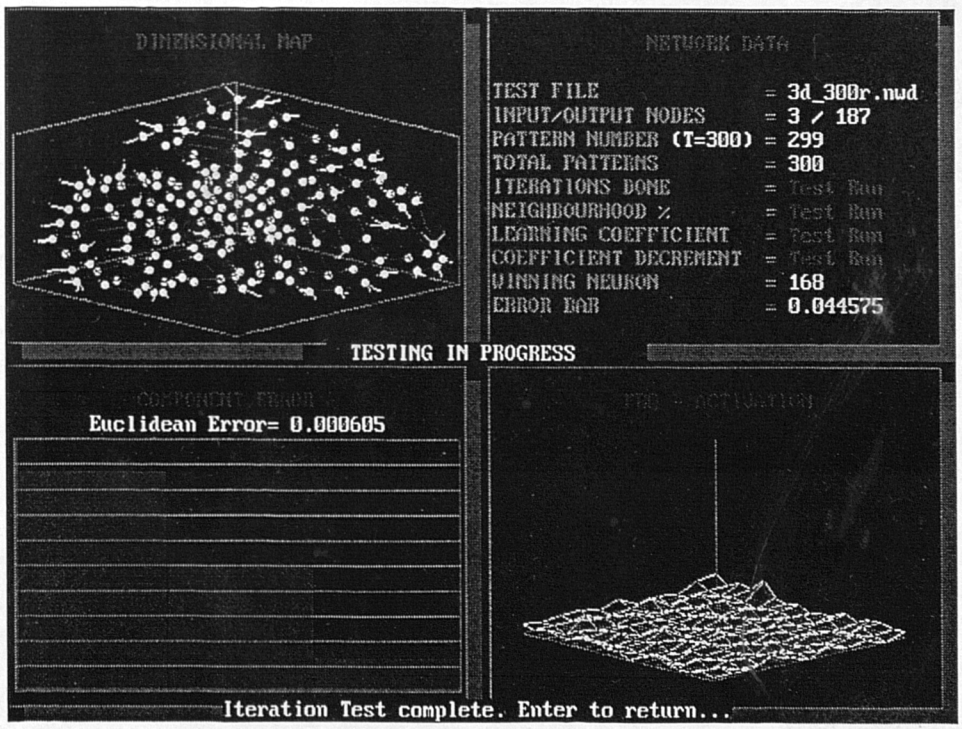
- MAP FOR 1 OR 2 DIMENSIONAL DATA (Y/N)..... YES
- MAP FOR 3 DIMENSIONAL DATA (Y/N)..... NO
- NETWORK NUMERICAL OUTPUT (Y/N)..... YES
- VECTOR COMPONENT ERROR (Y/N)..... YES
- FREQUENCY ACTIVATION MAP (Y/N)..... YES




DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d300cb.nwd
INPUT/OUTPUT NODES =
PATTERN NUMBER =
TOTAL PATTERNS =
ITERATIONS DONE =
NEIGHBOURHOOD % =
LEARNING COEFFICIENT =
COEFFICIENT DECREMENT =
WINNING NEURON =
ERROR BAR =
TEST PATTERN DISPERTION
COMPONENT ERROR
FREQ ACTIVATION


DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 2d300cb.nwd
INPUT/OUTPUT NODES =
PATTERN NUMBER =
TOTAL PATTERNS =
ITERATIONS DONE =
NEIGHBOURHOOD % =
LEARNING COEFFICIENT =
COEFFICIENT DECREMENT =
WINNING NEURON =
ERROR BAR =
TRAINED SYNAPTIC WEIGHTS
COMPONENT ERROR
FREQ ACTIVATION

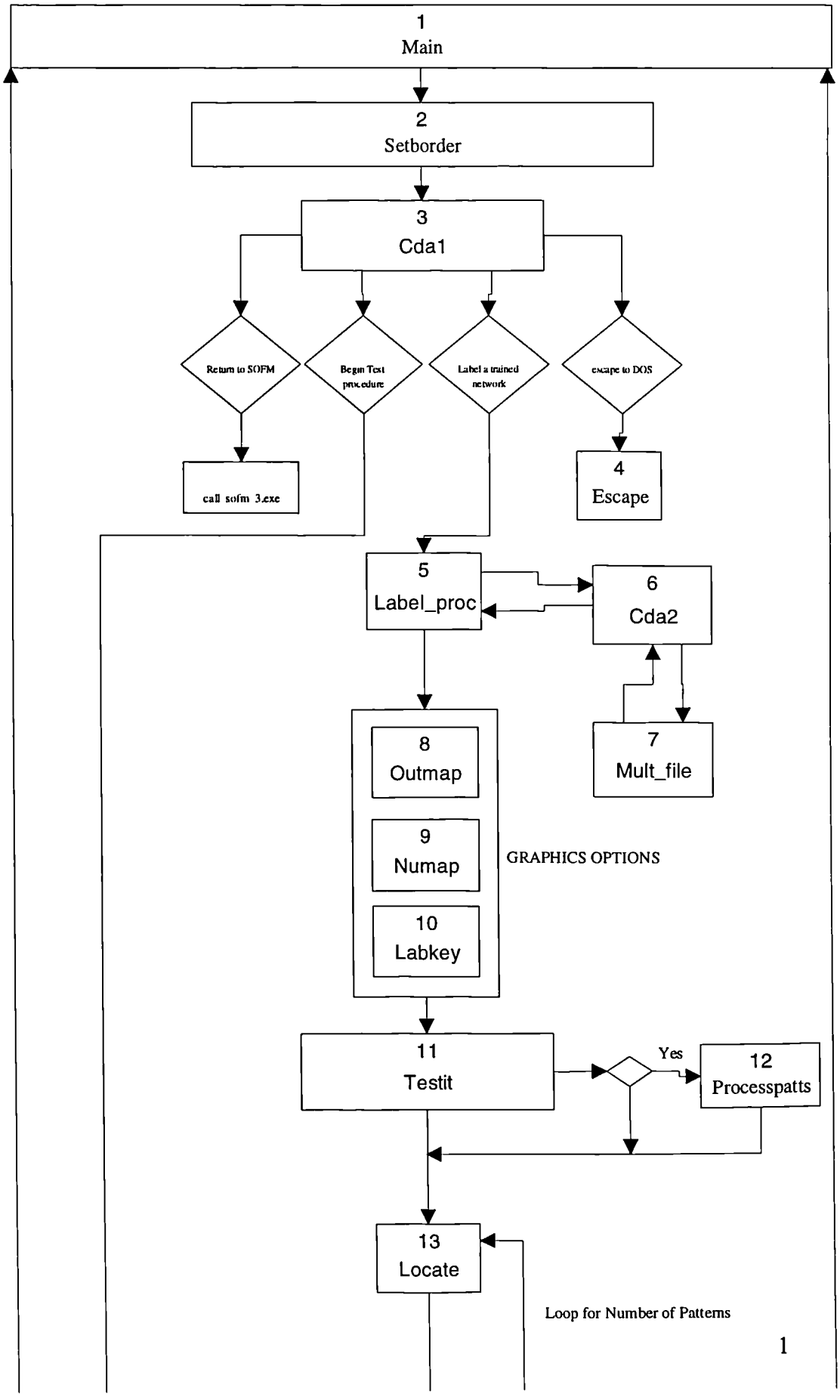

DIMENSIONAL MAP
NETWORK DATA
TRAINING FILE = 3d_300r.nwd
INPUT/OUTPUT NODES = 3 / 187
PATTERN NUMBER (T=300) = 929
TOTAL PATTERNS = 3093
ITERATIONS DONE(T=12) = 10
NEIGHBOURHOOD % = Winner
LEARNING COEFFICIENT = 0.100000
COEFFICIENT DECREMENT = 1.000000
WINNING NEURON = 0
ERROR BAR = 0.090686
ORDERING COUNT
COMPONENT ERROR
Euclidean Error= 0.000555
(P)ause (H)alt (T)oggle (C)hange

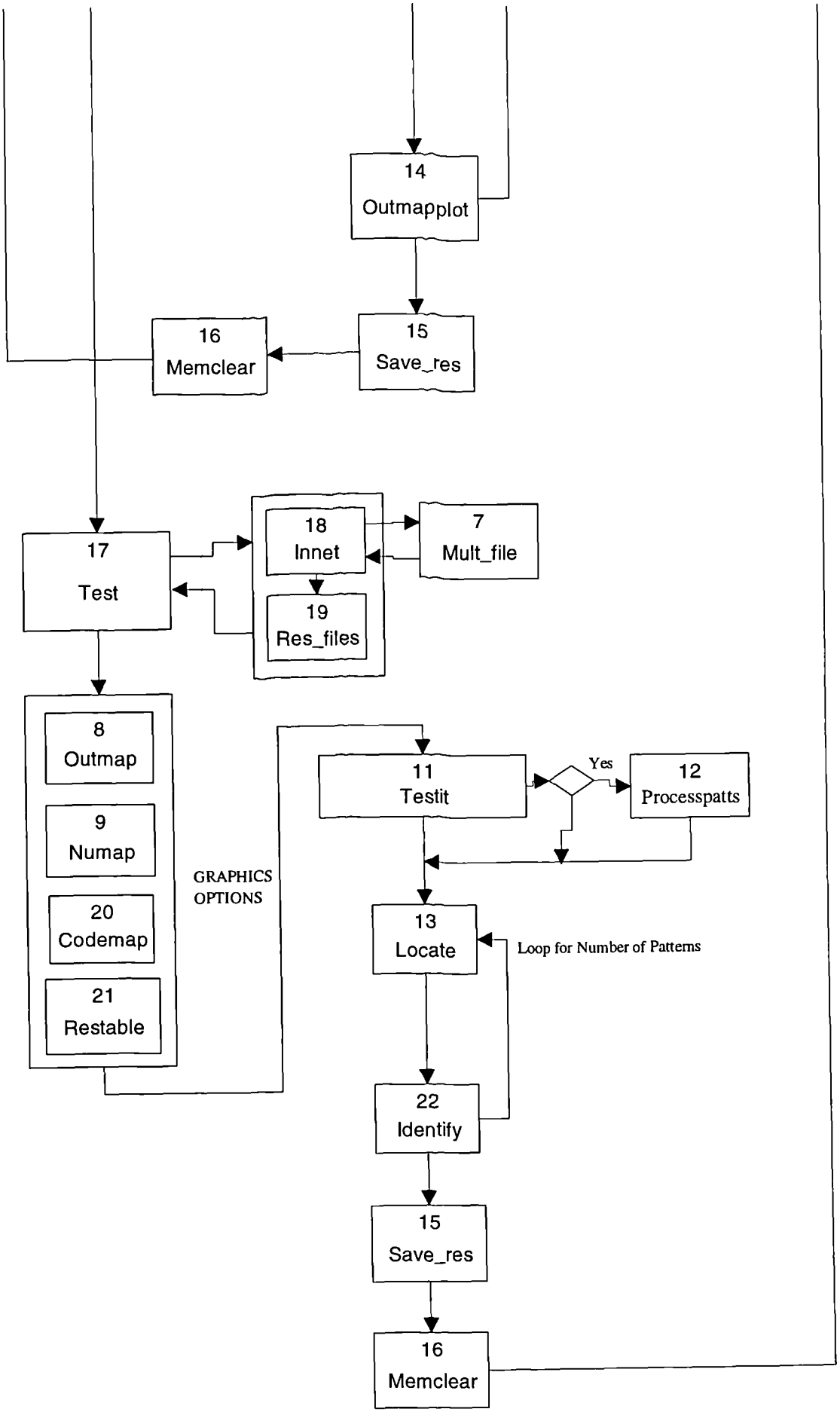

DIMENSIONAL MAP
NETWORK DATA
TEST FILE = 3d_300r.nwd
INPUT/OUTPUT NODES = 3 / 187
PATTERN NUMBER (T=300) = 299
TOTAL PATTERNS = 300
ITERATIONS DONE = Test Run
NEIGHBOURHOOD % = Test Run
LEARNING COEFFICIENT = Test Run
COEFFICIENT DECREMENT = Test Run
WINNING NEURON = 168
ERROR BAR = 0.044575
TESTING IN PROGRESS
Euclidean Error= 0.000605
Iteration Test complete. Enter to return...

APPENDIX 4

Source Code for SOFM Classifier

Flow Chart for Code Book Data Analysis (CDA_1.c)




14
Outmapplot
16
Memclear
15
Save_res
17
Test
18
Innet
7
Mult_file
19
Res_files
8
Outmap
9
Numap
20
Codemap
21
Restable
GRAPHICS
OPTIONS
11
Testit
Yes
12
Processpatts
13
Locate
Loop for Number of Patterns
22
Identify
15
Save_res
16
Memclear

FUNCTIONS KEY

2 void setborder(short color);	Sets colour & format for screen border
3 void cda1(void);	Sets title page & switches to test, train, NW or escape
5 void label_proc(void);	Resets any training parameters and calls training procedure save all necessary files post training
6 void cda2(void);	Takes and sets training data file parameters
7 void mult_file(void);	Multi-test/train file input
8 void outmap(void);	Map of output layer
9 void numap (void);	Sets background text for numerical data with titles
10 void labkey (void);	Label key table - graphics
11 void testit(void);	Full test iteration on data file
12 void processpatts(void);	Normalises input pattern matrix
13 void locate(void);	Computes Euclidean errors and No of winning neuron
14 void outmapplot(void);	Plots online firing frequency of output nodes
15 void save_res(void);	Save all resuluts to file
16 void memclear(void);	Calls all procedures for freeing allocated memory
17 void test(void);	Preliminary test procedure and graphics setup
18 void innet(void);	Loads all pretrained network data
19 void res_files(void);	Set up result files from test
20 void codemap(void);	Colour codes pre-labelled map
21 void restable (void);	Results table graphics
22 void identify(void);	Shows winning neuron on labelled layer

```
/*****************************************************************************
*                                                                            *
*  CDA_1.C (Code-Book Data Analysis).                                        *
*                                                                            *
*                                                                            *
*  Compiled by J.R. McCardle, Neural Applications Group, Brunel University   *
*  Department of Design.                                                     *
*****************************************************************************/

/* Inclusion headers */

#include   <stdio.h>
#include   <stdlib.h>
#include   <math.h>
#include   <conio.h>
#include   <ctype.h>
#include   <string.h>
#include   <graph.h>
#include   <time.h>
#include   <dos.h>
#include   <process.h>
#include   <share.h>
#include   <errno.h>
#include   <malloc.h>

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Program construction prototypes and Typedefs */

typedef float *PTR_FLOAT;
typedef PTR_FLOAT VECTOR;
typedef PTR_FLOAT *MATRIX;

int vectormem(VECTOR *ptr_vector, int Ncolumns);            /* Allocates memory for vectors */
void columnmem(PTR_FLOAT matrix[], int Nrows, int Ncolumns);/* Allocates memory for columns in..*/
                                                            /*.. matrix */
int matrixmem(MATRIX *ptr_matrix, int Nrows, int Ncolumns); /* Allocates memory for variable...*/
                                                            /*.. matrix */
void matrixdel(MATRIX matrix, int Nrows, int Ncolumns);     /* Releases all memory allocated to..*/
                                                            /*... matrices */

void setborder(short color);  /* Sets colour & format for screen border */
void Enterkey(void);          /* Looks for & registers return key press */

void cda1(void);         /* Sets title page & switches to test, train, NW or escape */
void cda2(void);         /* Takes and sets training data file parameters */


void calcout(void);       /* Calculates No of outputs nodes to form output layer */

void outmap(void);        /* Map of output layer */
void numap (void);        /* Sets background text for numerical data with titles */
void frqout (void);       /* Open and save outlayer frequency map */
void restable (void);     /* results table graphics */
void labkey (void);       /* Label key table - graphics */

void processpatts(void);      /* Normalises input pattern matrix */


void label_proc(void);        /* Resets any training parameters and calls training procedure,..*/
                              /*.. save all necessary files post training */

void test(void);              /* Preliminary test procedure and graphics setup */
void testit(void);            /* Full test iteration on data file */
void innet(void);             /* Loads all pretrained network data */
void res_files(void);         /* Set up result files from test */
void save_res(void);          /* Save all resuluts to file */
void mult_file(void);         /* Multi-test/train file input */


void locate(void);        /* Computes Euclidean errors and No of winning neuron */
void limit(void);         /* Locates winner in output layer, defines neighbourhood, adapts...*/
                          /*... synaptic weights */


void memclear(void);          /* Calls all procedures for freeing allocated memory */
void yesno(void);             /* Registers a yes or no answer as TRUE/FALSE */

void identify(void);          /* Shows winning neuron on labelled layer */
void outmapplot(void);        /* Plots online firing frequency of output nodes */
void codemap(void);           /* Colour codes pre-labelled map */
void outmaptidy(void);        /* Redraws firing frequency plot with mesh */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Global defines */

#define INCOLOR         0       /* black */
#define BACKGNDCOLOR    3       /* cyan */
#define AXISCOLOR       1       /* blue */
#define GRAPHCOLOR      4       /* red */
#define TITLECOLOR      14      /* yellow */
#define LABELCOLOR      15      /* white */

#define ALWAYS    1            /* decision flags */
#define TRUE      1
#define FALSE     0

/* Composition of "printat" */
#define printat(x, y, z)        _settextposition(x, y);\
                _outtext(z);
```

```
#define YCENTER 240             /* Center pixel coordinates of graphics */
#define XCENTER 320             /* screen                               */

     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
          /* DYNAMIC ARRAY STORAGE DEFINES */

MATRIX inlayer;
MATRIX outlayer;
MATRIX syn_weights;
MATRIX labelvec;
MATRIX out_label;


VECTOR pattern_no;
VECTOR out_colour;
VECTOR euc_err;

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Global variables */

short   xorig,yorig;    /* Origins for graphics pages */

unsigned int lab;          /* train or test flags */
unsigned int tes;
unsigned int labf;         /* No of label and/or test files */
unsigned int tesf;
unsigned int nodecolour;       /* Change of output node colour */

unsigned int side;              /* Side dimension of output layer */
unsigned int end;               /* End dimension of output layer */
unsigned int  rep;              /* Repeat procedure flag */
unsigned long p;        /* inpatts flag */
unsigned long o;        /* outnodes flag */
unsigned long n;        /* innodes flag */
unsigned int win;               /* winning neuron */
unsigned int neighbside;        /* neighbourhood dimensions of ratio 5:8 */
unsigned int left, right, up, down; /* limits of output layer for neighbourhood */
unsigned int labdim;        /* Dimensions of label vector */

unsigned int norm;      /* normalisation routine */
unsigned int ans;               /* Yes or no flag */
unsigned int tfile_lab;     /* test file is labelled */

unsigned int tesfno;        /* No of present test file */
unsigned int labfno;        /* No of present train file */

unsigned long innodes;          /* Number of input nodes */
unsigned long inpatts;          /* Number of input patterns */
unsigned long outnodes;         /* Number of ouput nodes */
unsigned long e;        /* epochs flag */
unsigned long labnums;      /* Number of separate labels supplied with data */


float big;              /* largest possible % neighbourhood */
float defneigh;         /* Default neibourhood size as a fraction of output layer */
float defval;           /* Default learning coefficient */
float defdec;           /* Default learning coefficient decrement */
float Small_Euc_Dist;   /* Smallest Euclidean distance gives winning neuron */
float bar;      /* error vector total for error bar plot */

FILE *outweights, *inweights;         /* Pointers for I/O files */
FILE *inlabfile, *intesfile, *outresfile;                              /* and output save files */

char drive[_MAX_DRIVE], dir[_MAX_DIR];
char fname[_MAX_FNAME], ext[_MAX_EXT];

char outpathweights[_MAX_PATH], inpathweights[_MAX_PATH];

char testfiles[20][_MAX_PATH];
char resultfiles[20][_MAX_PATH];
char labelfiles[20][_MAX_PATH];
char labnames[10][10];
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*****************************************************************************/

void main(void) {


/* TO ENSURE RUN_TIME FLOATING POINT LIB IS LOADED ONLY ! thanks Microsoft !!!*/

    defneigh = 1.0F;            /* Set default neighbourhood */
    defval = 0.60F;     /* Set default learning coeff */
    defdec = 0.20F;     /* Set default learning coeff decrement */


_setvideomode(_MAXRESMODE);

_setbkcolor(_BLACK);

setborder(11);

  while(ALWAYS) {
    lab = FALSE;       /* reset flags */
    tes = FALSE;

    _setvieworg(0,0);
    _clearscreen(_GCLEARSCREEN);
    cdal();
```

```
    /* run label    procedure */

    if(lab) {
        label_proc();

     }

    /* otherwise do test routine */

    if(tes){
         test();

      }


   } /* end of always loop */

} /* End of main() */

/*************************************************************************/

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void cda1(void) {

int input[3];
int ok;
ok = FALSE;

{
_settextcolor(GRAPHCOLOR);
printat(yorig + 2, xorig + 28, "CODE BOOK VECTOR MAPPING");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 7, xorig + 3, " Pre-trained networks are *.nww files. Output layers are labelled
                               with");
printat(yorig + 8, xorig + 3, " conditioned data files *.nwd with the desired label on the next
                               line");
printat(yorig + 9, xorig + 3, " following the input pattern. All files are comma separated ascii
                               text.");


_settextcolor(3);
printat(yorig + 17, xorig + 24, " (R)ETURN to training & test procedure.");
printat(yorig + 19, xorig + 24, " (L)ABEL a trained network.");
printat(yorig + 21, xorig + 24, " (T)EST a labelled network.");
printat(yorig + 23, xorig + 24, " (E)SCAPE to DOS.");

/* Little red circles */
_setcolor(4);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 260, xorig + 187, yorig + 267);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 292, xorig + 187, yorig + 299);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 324, xorig + 187, yorig + 331);
_ellipse(_GFILLINTERIOR, xorig + 180, yorig + 356, xorig + 187, yorig + 363);

          /*--------------------------------------*/

                  /* Switch modes */

while(input[0] = getch()){

  switch(input[0]) {

    case 'e':
    case 'E': memclear();                          /* Escape to DOS */
          _clearscreen(_GCLEARSCREEN);
          _setvideomode(_DEFAULTMODE);
          _displaycursor(_GCURSORON);
          _settextposition(25,81);
          exit(0);

    case 'r':
    case 'R': memclear();
          _clearscreen(_GCLEARSCREEN);
          execl("sofm_3.exe", NULL);    /* Call SOFM procedure */
          exit(1);

    case 'l':
    case 'L': memclear();
          _clearscreen(_GCLEARSCREEN);  /* continue with label */
          lab = TRUE;
          ok = TRUE;
          break;

    case 't':
    case 'T': memclear();
          _clearscreen(_GCLEARSCREEN);  /* continue with test */
          tes = TRUE;
          ok = TRUE;
          break;
    } /* end of switch */

if (ok == TRUE) break;

} /* end of while loop */

}

}   /* end of cda1() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void label_proc(void) {

char mess[45];

unsigned long c, i, a;

float r;

_clearscreen(_GCLEARSCREEN);
cda2();

/* do all remaining memory allocations */
vectormem(&pattern_no, (short)inpatts);
matrixmem(&outlayer, (short)inpatts, (short)outnodes);
if(rep == 1)
   {
     matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));
     matrixdel(labelvec, (short)inpatts, (short)labdim);
     matrixdel(outlayer, (short)inpatts, (short)outnodes);
     matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
   exit(0);
   }

matrixmem(&out_label, (short)outnodes, (short)labdim);
if(rep == 1)
   {
     matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));
     matrixdel(labelvec, (short)inpatts, (short)labdim);
     matrixdel(out_label, (short)outnodes, (short)labdim);
     matrixdel(outlayer, (short)inpatts, (short)outnodes);
     matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
   exit(0);
   }

vectormem(&out_colour, (short)outnodes);
vectormem(&euc_err, (short)outnodes);


printat(26, 27, "                              ");

_settextcolor(GRAPHCOLOR);
printat(26, 30, "FILE OPTIONS ACCEPTED");
printat(27, 29, "Press ENTER to continue.");
Enterkey();

_clearscreen(_GCLEARSCREEN);
outmap();
numap();
labkey();

_settextcolor(GRAPHCOLOR);
printat(25, 6, "Press ENTER to label layer.");
Enterkey();
printat(25, 6, "                              ");

for(a = 0; a < outnodes; a++){   /* load colour matrix with background colour */
out_colour[a] = 3;
}

nodecolour = 4; /* initiate the node colour */

for(labfno = 0; labfno < labf; labfno++){

     if(labnums > 1 && labfno != 0) { nodecolour += 1;}

    _settextposition(4, 65);
    _settextcolor(TITLECOLOR);
    sprintf(mess,"%s", &labelfiles[labfno][0]);
    _outtext(mess);

inlabfile = fopen( &labelfiles[labfno][0], "rb");
       for(i = 0; i < inpatts; i++) {
         for(c = 0; c < innodes; c++) {
         fscanf(inlabfile, "%f,", &r);
         inlayer[i][c] = r;
           }
         fscanf(inlabfile, "\n");
         for(c = 0; c < labdim; c++){
         fscanf(inlabfile, "%f,", &r);
         labelvec[i][c] = r;
        }
      fscanf(inlabfile, "\n");
       }

fclose(inlabfile);

testit();

}

printf("\a");
_settextcolor(GRAPHCOLOR);
printat(25, 5, "Network Labelled. Press Enter.");
Enterkey();
printat(25, 5, "                              ");


printat(25, 6, "SAVING NETWORK LABEL MATRIX");
printat(26, 6, "        PLEASE WAIT.");


outweights = fopen(outpathweights, "wb");
```

```
/* print file header */
fprintf(outweights, "%ld\n%ld\n%d\n%d\n%ld\n", outnodes, innodes, side, end, labdim);

for(a = 0; a < labdim; a++){
fprintf(outweights, "%s\n", &labnames[a][0]);
}

for(o = 0; o < outnodes; o++) {     /* print data to file comma separated */
    for(i = 0; i < innodes; i++) {
          fprintf(outweights, "%f,", syn_weights[o][i]);
    }
    fprintf(outweights, "\n");
    for(c = 0; c < labdim; c++){
    fprintf(outweights, "%f,", out_label[o][c]);
    }
    fprintf(outweights, "\n");
    fprintf(outweights, "%d,", (int)out_colour[o]);
    fprintf(outweights, "\n");
}
fclose(outweights);

printat(25, 6, "                              ");

printat(25, 6, "LABELLING IS NOW COMPLETE AND");
printat(26, 6, "      ALL FILES SAVED.");
printat(27, 4, "Press ENTER to return to MENU PAGE");

Enterkey();


} /*end of label_proc()*/

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void locate(void)
             /* Locate winning output neuron */
{
char dat[3];

float d;
unsigned long h;
float Euc_Dist; /* Actual Euclidean distance */


      Small_Euc_Dist = 0.0F;

      for(o = 0; o < outnodes; o++) {

          Euc_Dist = 0.0F;

        for(n = 0; n < innodes; n++) {
          d = inlayer[p][n] - syn_weights[o][n];
          Euc_Dist += d * d;
          }

          if(o == 0 || Euc_Dist <= Small_Euc_Dist) {
         Small_Euc_Dist = Euc_Dist;
          win = (short)o;
          }
     }


/* colour & label winning node if smallest euclidean error */

if(lab && !tes){
   if( euc_err[win] == 0 || euc_err[win] > Small_Euc_Dist){
      euc_err[win] = Small_Euc_Dist ;
      out_colour[win] = nodecolour;
      for(h = 0; h < labdim; h++){
     out_label[win][h] = labelvec[p][h];
       }
    }
 }

/* update numerical output */

printat(6, 65, "      ");
_settextposition(6, 65);
_settextcolor(TITLECOLOR);
sprintf(dat, "%d", win + 1);
_outtext(dat);

} /* end of locate */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


void cda2(void) {

int  c, i, b;
unsigned long a, x;

float r;

char mess[45];
char dat[12];

{
_settextcolor(GRAPHCOLOR);
```

```
printat(yorig + 2, xorig + 30, "FILE DECLARATIONS");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 4, xorig + 9, " Please Enter the desired file names & paths at the prompts.");
printat(yorig + 5, xorig + 12, " Extentions are awarded automatically for OUTPUT files.");

_settextcolor(3);
printat(yorig + 7, xorig + 5, " FILE NAME OF PRE-TRAINED NETWORK (*.nww) .............");
printat(yorig + 9, xorig + 5, " NUMBER OF INPUT PATTERN LABEL FILES ..................");
printat(yorig + 11, xorig + 5, " EUCLIDEAN NORMALISATION REQUIRED (Y/N) ..............");
printat(yorig + 13, xorig + 5, " FILE TO STORE LABELLED NETWORK (*.nwl automatic).....");
printat(yorig + 15, xorig + 5, " NUMBER OF INPUT PATTERNS PER LABEL FILE .............");
printat(yorig + 17, xorig + 5, " DIMENSIONALITY OF LABEL VECTOR ......................");
printat(yorig + 19, xorig + 5, " LABEL TITLES (Number expected =     ) ..............");
printat(yorig + 21, xorig + 5, " NUMBER OF DIFFERENT LABELS SUPPLIED ..................");

/* Little red circles */
_setcolor(4);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 100, xorig + 27, yorig + 107);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 132, xorig + 27, yorig + 139);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 164, xorig + 27, yorig + 171);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 196, xorig + 27, yorig + 203);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 228, xorig + 27, yorig + 235);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 260, xorig + 27, yorig + 267);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 292, xorig + 27, yorig + 299);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 324, xorig + 27, yorig + 331);
}

         /*--------------------------------------*/

/* ask for pretrained network */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 7, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 7, xorig + 62);

flushall();   /* clear buffer */

   do{
     _settextposition(yorig + 7, xorig + 62);
     gets(inpathweights);
     _settextcolor(TITLECOLOR);
     _outtext(inpathweights);

     if( (inweights = fopen( inpathweights, "rb")) == NULL ) {   /* error message */
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     sprintf( mess, " Can't open weights file, error number: %d", errno);
     printat(17, 18, mess);
     printat(18, 16, " Drive and/or Directory has to exist to use !");
     printat(20, 20, " Press ENTER to repeat declaration.");

     Enterkey();

     /* Erase error message */
    printat(17, 18, "                                                ");
    printat(18, 16, "                                                ");
    printat(20, 20, "                                                ");
    printat(yorig + 7, xorig + 62, "                    "); /* Erase path */

    }

 } while( inweights == NULL);   /* Loop declaration until valid */


_settextcolor(GRAPHCOLOR);
printat(26, 27, "PROCESSING DATA PLEASE WAIT");

fscanf(inweights, "%ld\n%ld\n%d\n%d\n", &outnodes, &innodes, &side, &end);      /* read header */

matrixmem(&syn_weights, (short)outnodes, ((short)innodes + 1)); /* set matrix */
if(rep == 1)
   {
     matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
   exit(0);
   }



for(i = 0; i < outnodes; i++) {      /* read data */
    for(c = 0; c < innodes; c++) {
    fscanf(inweights, "%f,", &r);
    syn_weights[i][c] = r;
      }
      fscanf(inweights, "\n");
 }

fclose(inweights);

printat(26, 27, "                           ");

printat(yorig + 7, xorig + 60, " ");  /* Erase delta prompt */


/* Ask for number of label files */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 9, xorig + 60);
_outtext("\x10");                    /* ASCII delta to right character */
_settextposition(yorig + 9, xorig + 62);
scanf("%d", &labf);
sprintf(dat,"%d", labf);
```

```
_settextcolor(TITLECOLOR);
_outtext(dat);
printat(yorig + 9, xorig + 60, " ");      /* Erase delta cursor */
}

/* check for normalisation */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 11, xorig + 60);
_outtext("\x10");                         /* ASCII delta to right character */

yesno();

norm = ans;

_settextcolor(TITLECOLOR);
_settextposition(yorig + 11, xorig + 62);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 11, xorig + 60, " ");    /* Erase delta cursor */

}

flushall();

/* Ask for output file name for labelled network */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 13, xorig + 60);
_outtext("\x10");                         /* ASCII delta to right character */

do{
_settextposition(yorig + 13, xorig + 62);
gets(outpathweights);
_settextcolor(TITLECOLOR);
_splitpath( outpathweights, drive, dir, fname, ext);    /* create output path */
strcpy ( ext, "nwl" );
_makepath( outpathweights, drive, dir, fname, ext );
_outtext(outpathweights);

if( (outweights = _fsopen( outpathweights, "wb", SH_DENYWR )) == NULL ) /* error message */

  {
     _settextcolor(GRAPHCOLCR);
     printf("\a"); /* Alarm */
     printat(13, 62, " Can't create file !!");
     Enterkey();

     /* Erase error message */
     printat(13, 62, "                      ");

   }
} while( outweights == NULL );  /* Loop declaration until valid */

fclose(outweights);

printat(yorig + 13, xorig + 60, " ");  /* Erase delta prompt */

}

         /*--------------------------------------*/

/* Ask for number of input patterns per label files */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 15, xorig + 60);
_outtext("\x10");                         /* ASCII delta to right character */
_settextposition(yorig + 15, xorig + 62);
scanf("%ld", &inpatts);
sprintf(dat,"%ld", inpatts);
_settextcolor(TITLECOLOR);
_outtext(dat);
printat(yorig + 15, xorig + 60, " ");    /* Erase delta cursor */
}

matrixmem(&inlayer, (short)inpatts, ((short)innodes + 1));  /* set matrices */
if(rep == 1)
   {
     matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
     matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));

   exit(0);
   }



/* Ask for dimensionality of label vector */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 17, xorig + 60);
_outtext("\x10");                         /* ASCII delta to right character */
_settextposition(yorig + 17, xorig + 62);
scanf("%d", &labdim);
sprintf(dat,"%d", labdim);
_settextcolor(TITLECOLOR);
_outtext(dat);
printat(yorig + 17, xorig + 60, " ");    /* Erase delta cursor */
_settextposition(yorig + 19, xorig + 38);
_outtext(dat);               /* Print expected dimensions */
```

```
}

matrixmem(&labelvec, (short)inpatts, (short)labdim); /* allocate label vector memory */
if(rep == 1)
    {
      matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
      matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));
      matrixdel(labelvec, (short)inpatts, (short)labdim);
    exit(0);
    }



/* Ask for label titles */
flushall();
{
for(a = 0, b = xorig + 60; a < labdim; a++, b += 4) {
_settextposition(yorig + 19, b);
_settextcolor(GRAPHCOLOR);
_outtext("\x10");
_settextposition(yorig + 19, b + 3);
gets(&labnames[a][0]);
sprintf(mess,"%s", &labnames[a][0]);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 19, b, " ");       /* Erase delta cursor */
}

}

/* Ask for No of different labels supplied */
{
_settextcolor(GRAPHCOLOR);
_settextposition
(yorig + 21, xorig + 60);
_outtext("\x10");                          /* ASCII delta to right character */
_settextposition(yorig + 21, xorig + 62);
scanf("%ld", &labnums);
sprintf(dat,"%ld", labnums);
_settextcolor(TITLECOLOR);
_outtext(dat);
printat(yorig + 21, xorig + 60, " ");      /* Erase delta cursor */
}

if(labf > 1) {
 mult_file();    /* set page for multiple file input list */
 }

else{
    _settextcolor(3);
    printat(yorig + 23, xorig + 5, " FILE NAME OF LABEL PATTERNS [DIMS EXPECTED =    ].....");
    _setcolor(4);
    _ellipse(_GFILLINTERIOR, xorig + 20, yorig + 356, xorig + 27, yorig + 363);


    _settextcolor(GRAPHCOLOR);
    _settextposition(yorig + 23, xorig + 61);
    _outtext("\x10");

    sprintf(mess,"%ld", innodes);
    _settextcolor(TITLECOLOR);
    _settextposition(yorig + 23, xorig + 51);
    _outtext(mess);


    flushall();

do{
   _settextposition(yorig + 23, xorig + 62);
   gets(&labelfiles[0][0]);
   _settextcolor(TITLECOLOR);
   sprintf(mess,"%s", &labelfiles[0][0]);
   _outtext(mess);


if( (inlabfile = fopen( &labelfiles[0][0], "rb")) == NULL ) { /* error message */
       _settextcolor(GRAPHCOLOR);
       printf("\a");              /* Alarm */
       printat(23, 62, " Can't find label file !!");
       Enterkey();

       /* Erase error message */
       printat(23, 62, "                          ");

     }

} while( inlabfile == NULL);    /* Loop declaration until valid */

printat(yorig + 23, xorig + 61, " ");        /* Erase delta prompt */

fclose(inlabfile);

} /* end of else */


} /* End of cda2 */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

               /* TEST PROCEDURE */

void test(void) {
```

```
unsigned long i, c, b;

int cmd[3];
float r, d, a;

char mess[45];
char dat[12];

{

innet();     /* fetch test data parameters */

res_files();     /* set up results files */

_settextcolor(GRAPHCOLOR);
printat(22, 30, "FILE OPTIONS ACCEPTED");

Enterkey();

/* draw labelled network graphics */

_clearscreen(_GCLEARSCREEN);
outmap();
numap();
codemap();
restable();

_settextcolor(GRAPHCOLOR);
printat(25, 6, "Press ENTER to test layer.");
Enterkey();
printat(25, 6, "                              ");

for( tesfno = 0; tesfno < tesf; tesfno++ ) {

    _settextposition(4, 65);
    _settextcolor(TITLECOLOR);
    sprintf(mess,"%s", &testfiles[tesfno][0]);
    _outtext(mess);

    intesfile = fopen( &testfiles[tesfno][0], "rb");

     for(i = 0; i < inpatts; i++) {
         for(c = 0; c < innodes; c++) {
         fscanf(intesfile, "%f,", &r);
         inlayer[i][c] = r;
          }
         fscanf(intesfile, "\n");
       if(tfile_lab){for(c = 0; c < labdim; c++){
        fscanf(intesfile, "%f,", &r);
        labelvec[i][c] = r;
         }
        fscanf(intesfile, "\n");}
      }

     fclose(intesfile);

     testit();

    }  /*end of for tesfno loop*/


_settextcolor(GRAPHCOLOR);
printf("\a");  /*alarm */
printat(25, 6,"Iteration Test complete. ");
printat(26, 6,"   Enter to return...");

Enterkey();

}
}  /*end of test()*/

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void res_files(void) {

int a;

for( a = 0; a < tesf; a++){

_splitpath( &testfiles[a][0], drive, dir, fname, ext);  /* create output path */
strcpy ( ext, "res" );
_makepath( &resultfiles[a][0], drive, dir, fname, ext );


if( (outresfile = _fsopen( &resultfiles[a][0], "wb", SH_DENYWR )) == NULL ) { /* error message */
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     printat(13, 62, " Can't create file !!");
     Enterkey();

     /* Erase error message */
     printat(13, 62, "                      ");


} while( outresfile == NULL );  /* Loop declaration until valid */

fclose(outresfile);

}

} /* end of res_files() */
```

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void save_res(void) {

int a, b, c;

outresfile = fopen(&resultfiles[tesfno][0], "a+b");

for(a = 0; a < labdim; a++) {   /* print data to file comma separated */
     if(tfile_lab){fprintf(outresfile, "%f,", labelvec[p][a]);}
     fprintf(outresfile, "%f,", out_label[win][a]);
}
fprintf(outresfile, "\n");

fclose(outresfile);  /* close results file */

} /* end of save_res() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void testit(void) {
int b, g;
char dat[12];
unsigned long j;


{

if(norm) {      /* perform normalisation if required */
         processpatts();
      }


for( p = 0; p < inpatts; p++ ) { /* draw individual input patterns */

    locate();    /*  find closest neuron */

if(lab && !tes){outmapplot();}  /* change colour of firing neuron */

if(tes && !lab){identify();}   /* flash winning neuron */

/* print pattern No */

printat(5, 65, "            ");
_settextcolor(TITLECOLOR);
 _settextposition(5, 65);
sprintf(dat, "%d", p + 1);
_outtext(dat);

if(tes && !lab){
printat((13 + tesfno), 55, "      ");
_settextcolor(TITLECOLOR);
 _settextposition((13 + tesfno), 55);
sprintf(dat, "%d", p + 1);
_outtext(dat);
}

if(lab && !tes){
       _settextcolor(TITLECOLOR);
       for(j = 0, b = 13, g = 59; j < labdim; j++, g += 7){
      _settextposition(b + labfno, g);
      printat(b + labfno, g, "      ");
      _settextposition(b + labfno, g);
      sprintf(dat, "%.1f", labelvec[p][j]);
      _outtext(dat);
    }
}

if(tes && !lab){
       _settextcolor(TITLECOLOR);
       for(j = 0, b = 13, g = 59; j < labdim; j++, g += 7){
      _settextposition(b + tesfno, g);
      printat(b + tesfno, g, "      ");
      _settextposition(b + tesfno, g);
      sprintf(dat, "%.1f", out_label[win][j]);
      _outtext(dat);
    }

save_res(); /* save result to file */

}

 printat(7,65, "           ");         /* update error bar output */
_settextposition(7, 65);
_settextcolor(TITLECOLOR);
sprintf(dat, "%.4f", Small_Euc_Dist);
_outtext(dat);

} /* end of pattern iteration test */

}

}  /* end testit() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


void innet(void) {    /* load pre-trained network data */

unsigned long  c, i, a;
```

```
unsigned int g;
float r;
char mess[45];


_settextcolor(GRAPHCOLOR);
printat(yorig + 2, xorig + 30, "FILE DECLARATIONS");
printat(yorig + 30, xorig + 24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(yorig + 4, xorig, " N.B. ENSURE THE TEST DATA AND NETWORK MATRIX ARE OF THE SAME
                          DIMENSIONALITY !");

_settextcolor(3);

printat(yorig + 7, xorig + 5, " FILE NAME FOR LABELLED NETWORK ( ?.nwl ).............");
printat(yorig + 9, xorig + 5, " NORMALISATION REQUIRED ON TEST PATTERNS ? (Y/N) ......");
printat(yorig + 11, xorig + 5, " NUMBER OF INPUT TEST PATTERNS PER FILE.............");
printat(yorig + 13, xorig + 5, " ARE THE TEST FILES LABELLED ? (Y/N) ..................");
printat(yorig + 15, xorig + 5, " NUMBER OF DATA FILES TO TEST .......................");


_settextcolor(GRAPHCOLOR);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 100, xorig + 27, yorig + 107);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 132, xorig + 27, yorig + 139);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 164, xorig + 27, yorig + 171);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 196, xorig + 27, yorig + 203);
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 228, xorig + 27, yorig + 235);

/* ask for labelled network */
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 7, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 7, xorig + 62);

flushall();   /* clear buffer */

   do{
     _settextposition(yorig + 7, xorig + 62);
     gets(inpathweights);
     _settextcolor(TITLECOLOR);
     _outtext(inpathweights);

     if( (inweights = fopen( inpathweights, "rb")) == NULL ) {   /* error message */
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     printat(7, 62, " Can't open network file !!");
     Enterkey();

     /* Erase error message */
    printat(7, 62, "                          ");

    }

 } while( inweights == NULL);   /* Loop declaration until valid */


_settextcolor(GRAPHCOLOR);
printat(22, 27, "PROCESSING DATA PLEASE WAIT");

/* read header */
fscanf(inweights, "%ld\n%ld\n%d\n%d\n%ld\n", &outnodes, &innodes, &side, &end, &labdim);

for(a = 0; a < labdim; a++){
fscanf(inweights, "%s\n", &labnames[a]);
}

matrixmem(&out_label, (short)outnodes, (short)labdim);
if(rep == 1)
   {
     matrixdel(out_label, (short)outnodes, (short)labdim);
   exit(0);
   }

matrixmem(&syn_weights, (short)outnodes, ((short)innodes + 1)); /* set matrix */
if(rep == 1)
   {
     matrixdel(out_label, (short)outnodes, (short)labdim);
     matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
   Enterkey();
   exit(0);
   }

vectormem(&out_colour, (short)outnodes);
vectormem(&euc_err, (short)outnodes);

for(i = 0; i < outnodes; i++) {  /* read data */
    for(c = 0; c < innodes; c++) {
    fscanf(inweights, "%f,", &r);
    syn_weights[i][c] = r;
      }
      fscanf(inweights, "\n");
       for(c = 0; c < labdim; c++){
      fscanf(inweights, "%f,", &r);
      out_label[i][c] = r;
    }
      fscanf(inweights, "\n");
      fscanf(inweights, "%d,\n", &g);
      out_colour[i] = g;
 }

fclose(inweights);

printat(22, 27, "                          ");
```

```
printat(yorig + 7, xorig + 60, " ");  /* Erase delta prompt */


/* check for normalisation */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 9, xorig + 60);
_outtext("\x10");                        /* ASCII delta to right character */

yesno();

norm = ans;

_settextcolor(TITLECOLOR);
_settextposition(yorig + 9, xorig + 62);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 9, xorig + 60, " ");    /* Erase delta cursor */

}

/* Ask for number of input patterns */

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 11, xorig + 60);
_outtext("\x10");
_settextposition(yorig + 11, xorig + 62);
scanf("%ld", &inpatts);
sprintf(mess,"%ld", inpatts);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 11, xorig + 60, " ");     /* Erase delta cursor */

matrixmem(&labelvec, (short)inpatts, (short)labdim); /* allocate label vector memory */
if(rep == 1)
   {
     matrixdel(out_label, (short)outnodes, (short)labdim);
     matrixdel(syn weights, (short)outnodes, ((short)innodes + 1));
     matrixdel(labelvec, (short)inpatts, (short)labdim);
   exit(0);
   }

vectormem(&pattern_no, (short)inpatts);

matrixmem(&inlayer, (short)inpatts, ((short)innodes + 1)); /* set matrices */
if(rep == 1)
   {
     matrixdel(out_label, (short)outnodes, (short)labdim);
     matrixdel(labelvec, (short)inpatts, (short)labdim);
     matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
     matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));
   Enterkey();
   exit(0);
   }

matrixmem(&outlayer, (short)inpatts, (short)outnodes);
if(rep == 1)
   {
     memclear();
   Enterkey();
   exit(0);
   }

/* check for labelled test files */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 13, xorig + 60);
_outtext("\x10");                        /* ASCII delta to right character */

yesno();

tfile_lab = ans;

_settextcolor(TITLECOLOR);
_settextposition(yorig + 13, xorig + 62);
   if(ans) {
    _outtext("YES");
    }
   else {
    _outtext("NO");
    }

printat(yorig + 13, xorig + 60, " ");   /* Erase delta cursor */

}

/* Ask for number of data files */
{
_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 15, xorig + 60);
_outtext("\x10");                        /* ASCII delta to right character */
_settextposition(yorig + 15, xorig + 62);
scanf("%d", &tesf);
sprintf(mess,"%d", tesf);
_settextcolor(TITLECOLOR);
_outtext(mess);
printat(yorig + 15, xorig + 60, " ");    /* Erase delta cursor */
}
```

```
if(tesf == 1) {

/* Ask for drive and file declaration */
/* and load input pattern matrix */

flushall();  /* clear all buffers */
_ellipse(_GFILLINTERIOR, xorig + 20, yorig + 260, xorig + 27, yorig + 267);
_settextcolor(3);
printat(yorig + 17, xorig + 5, " FILE NAME OF TEST PATTERNS [DIMS EXPECTED =    ].....");

sprintf(mess,"%ld", innodes);
_settextcolor(TITLECOLOR);
_settextposition(yorig + 17, xorig + 50);
_outtext(mess);

_settextcolor(GRAPHCOLOR);
_settextposition(yorig + 17, xorig + 60);
_outtext("\x10");                        /* ASCII delta to right character */

do {

_settextposition(yorig + 17, xorig + 62);
gets(&testfiles[0][0]);
_settextcolor(TITLECOLOR);
sprintf(mess,"%s", &testfiles[0][0]);
_outtext(mess);


if( (intesfile = fopen( &testfiles[0][0], "rb")) == NULL ) /* error message */
  {
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     printat(17, 62, " Can't open test file !!");
     Enterkey();

     /* Erase error message */
     printat(17, 62, "                        ");

  }

} while( intesfile == NULL);     /* Loop declaration until valid */

printat(yorig + 17, xorig + 60, " ");        /* Erase delta prompt */

fclose(intesfile);


}

if(tesf > 1) {
     mult_file();
     }

} /* end of innet() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void mult_file(void) {

unsigned int a, yg, yt;
char q[2];
char mess[45];


{
 /* draw file input box and title */
_setcolor(8);
_rectangle(_GFILLINTERIOR, XCENTER - 150, YCENTER - 200, XCENTER + 170, YCENTER + 200);
_setcolor(0);
_rectangle(_GFILLINTERIOR, XCENTER - 160, YCENTER - 210, XCENTER + 160, YCENTER + 190);
_setcolor(3);
_rectangle(_GBORDER, XCENTER - 160, YCENTER - 210, XCENTER + 160, YCENTER + 190);

_settextcolor(GRAPHCOLOR);
printat(4, 31, "INPUT FILE LIST");
_settextcolor(3);
printat(5, 23, " PATTERN DIMENSION EXPECTED =    ");
sprintf(mess,"%ld", innodes);
_settextcolor(TITLECOLOR);
_settextposition(5, 53);
_outtext(mess);
}

 /* for No of tesf or labf number slots */
if(tes) {
    for(a = 1, yt = 7; a <= tesf; a++, yt += 1) {
       _settextcolor(GRAPHCOLOR);
       _settextposition(yt, 25);
       sprintf(q, "%d.", a);
       _outtext(q);
      }
  }

  if(lab) {
    for(a = 1, yt = 7; a <= labf; a++, yt += 1) {
       _settextcolor(GRAPHCOLOR);
       _settextposition(yt, 25);
       sprintf(q, "%d.", a);
       _outtext(q);
      }
  }
```

```
  /* take in file names and paths and label as strings */

flushall();

if(tes) {
    for(tesfno = 0, yt = 7; tesfno < tesf; tesfno++, yt += 1) {
      do{
    _settextcolor(GRAPHCOLOR);
    _settextposition(yt, 28);
    _outtext("\x10");          /* ASCII delta to right character */
    _settextposition(yt, 30);
    gets(&testfiles[tesfno][0]);

       if( (intesfile = fopen( &testfiles[tesfno][0], "rb")) == NULL ) /* error message */
       {
       _settextcolor(GRAPHCOLOR);
       printf("\a");              /* Alarm */
       sprintf(mess, "I can't find it !");
       _outtext(mess);
       Enterkey();
       printat(yt, 30, "                    ");
    }

    } while (intesfile == NULL);


    _settextcolor(TITLECOLOR);
    sprintf(mess,"%s", &testfiles[tesfno][0]);
    _outtext(mess);
    printat(yt, 28, " ");
    fclose(intesfile);


     } /* end of for loop */

 } /* end of if tes */


if(lab) {
    for(labfno = 0, yt = 7; labfno < labf; labfno++, yt += 1) {
      do{
    _settextcolor(GRAPHCOLOR);
    _settextposition(yt, 28);
    _outtext("\x10");          /* ASCII delta to right character */
    _settextposition(yt, 30);
    gets(&labelfiles[labfno][0]);

       if( (inlabfile = fopen( &labelfiles[labfno][0], "rb")) == NULL ) /* error message */
       {
       _settextcolor(GRAPHCOLOR);
       printf("\a");              /* Alarm */
       sprintf(mess, "I can't find it !");
       _outtext(mess);
       Enterkey();
       printat(yt, 30, "                    ");
    }

    } while (inlabfile == NULL);


    _settextcolor(TITLECOLOR);
    sprintf(mess,"%s", &labelfiles[labfno][0]);
    _outtext(mess);
    printat(yt, 28, " ");
    fclose(inlabfile);


     } /* end of for loop */

 } /* end of if lab */

} /* end of mult_file() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


/* ROUTINES FOR MEMORY ALLOCATION */

int vectormem(VECTOR *ptr_vector, int Ncolumns)
{
{    *ptr_vector = (VECTOR) calloc(Ncolumns, sizeof(float));

    if( *ptr_vector == NULL)
    {                                          /* sets memory allocation for vectors */
     _settextcolor(GRAPHCOLOR);
   printf("\a");
   printat(27, 22, "MEMORY TOO SMALL FOR VECTOR ALLOCATION");
   printat(28, 18, "Press ENTER and try smaller number of patterns.");
   rep = 1;
   Enterkey();
    }

}
}  /* end of vectormem */
        /* - - - - - - - - - - - - - - - - */

int matrixmem(MATRIX *ptr_matrix, int Nrows, int Ncolumns)
{
{   *ptr_matrix = (MATRIX) calloc(Nrows, sizeof(float));

   if(*ptr_matrix == NULL)
   {                                                /* sets memory allocation for matrices */
```

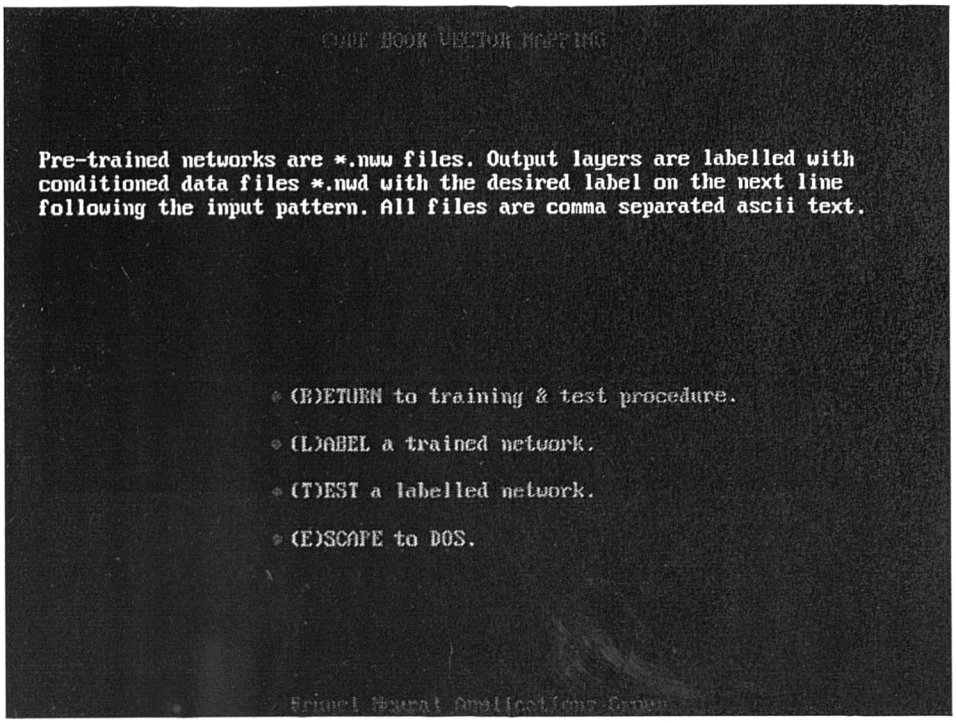
```
   _settextcolor(GRAPHCOLOR);
   printf("\a");
   printat(27, 22, "MEMORY TOO SMALL FOR MATRIX ALLOCATION");
   printat(28, 18, "Press ENTER and try smaller nodal combinations.");
   rep = 1;
   Enterkey();
   }

columnmem(*ptr_matrix, Nrows, Ncolumns);

}
}   /* end of matrixmem */

         /* - - - - - - - - - - - - - - - - */

void columnmem(PTR_FLOAT matrix[], int Nrows, int Ncolumns)
{
int i;
      for(i = 0; i < Nrows; i++) {                          /* sets up column matrix */
       if(rep != 1) {
      vectormem(&matrix[i], Ncolumns);
       }
}
}    /* end of columnmem */

        /* - - - - - - - - - - - - - - - - */

void matrixdel(MATRIX matrix, int Nrows, int Ncolumns)
{
int i, a;
      for(i = 0; i < Nrows; i++){               /* deletes all matrices */
     for(a = 0; a < Ncolumns; a++){
      free(matrix);
     }
    free(matrix[i]);
       }
}    /* end of matrixdel */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void processpatts(void) {

/************************************************
 *   Compute Euclidean Norm for pattern matrix.  *
 *  of form :- P_new = P_old / [E(P_old)^2] ^1/2 *
 ************************************************/
{
unsigned long  c, i;

double  d = 0.0;
double  a = 0.0;

for(i = 0; i < inpatts; i++) {

    for(c = 0; c < innodes; c++) {
     a = inlayer[i][c];
     d += (a * a);
     }

 d = sqrt(d);

    for(c = 0; c < innodes; c++) {
     inlayer[i][c] /= d;
     }
 d = 0;
}

}
}  /* End of pattern normalisation routine */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void memclear(void)
{
     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
             /* FREE ALL MEMORY ALLOCTIONS */

matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));
matrixdel(outlayer, (short)inpatts, (short)outnodes);
matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));
matrixdel(out_label, (short)outnodes, (short)labdim);
matrixdel(labelvec, (short)inpatts, (short)labdim);

free(pattern_no);
free(out_colour);
free(euc_err);

}  /* end of memclear() */

     /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*************************************************************************/
             /* GRAPHICS ROUTINES */
/*************************************************************************/


void numap(void) {

_settextcolor(GRAPHCOLOR);          /* title */
```

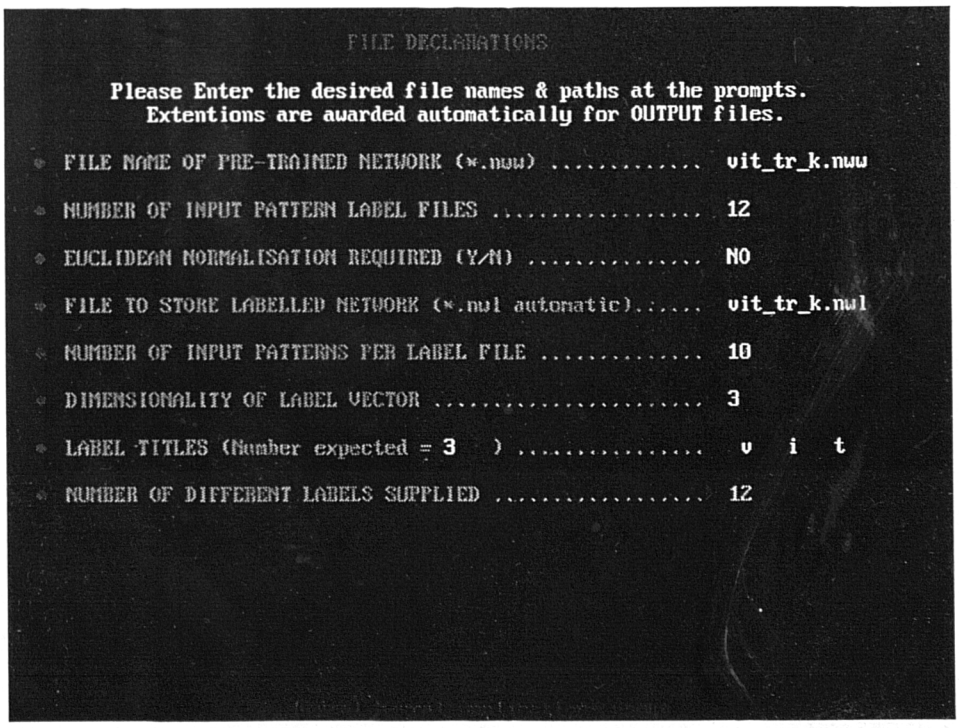
```
printat(2, 55, "DETAILS");

_setcolor(3);
_rectangle(_GBORDER, 315, 40, 630, 124);

_settextcolor(3);
if(lab && !tes){
printat(4, 44, "LABEL FILE        =");
}

if(tes && !lab){
printat(4, 44, "TEST FILE         =");
}

printat(5, 44, "PATTERN NUMBER    =");
printat(6, 44, "WINNING NEURON    =");
printat(7, 44, "ERROR BAR         =");

} /* end of numap() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void outmap(void) {

unsigned int x, y, a, b, c;
unsigned int x1, y1;
unsigned int tempx, tempy;
unsigned int tempX, tempY;
char mess[25];

{


_settextcolor(GRAPHCOLOR);                    /* title */
printat(2, 15, "OUTPUT LAYER");
_settextcolor(TITLECOLOR);
_settextposition(3, 17);
_outtext(inpathweights);



if(tes){
    _settextposition(4, 65);
    sprintf(mess, "%s", &testfiles[tesfno][0]);
    _outtext(mess);
}

_setcolor(3);              /* draw bkgnd rectangle*/
_rectangle(_GFILLINTERIOR, 20, 50, 295, 325);

tempX = 270 / end;
tempY = 270 / end;
tempx = 270 / side; /* available pixels to allocate outlayer nodes */
tempy = 270 / side;

_setcolor(0);         /* draw black base grid */

for(b = 0, x1 = 25, y1 = 55; b <= end; b++, y1 += tempY) {
    _moveto(x1, y1);                  /* draw horizantal lines */
    _lineto(x1 + tempx * side , y1);
    }


for(a = 0, x1 = 25, y1 = 55; a <= side; a++, x1 += tempx) {
    _moveto(x1, y1);                    /* draw vertical lines */
    _lineto(x1, y1 + tempY * end);
    }

}

} /* end of outmap() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void outmapplot(void) {

int winend, winside;
int x, y, a, b, c, z;
int x1, y1;
int tempx, tempy;
int tempX, tempY;
int node_size = 1;

{

x1 = 25; /* starting pixel */
y1 = 55;

tempX = 270 / end;
tempY = 270 / end;
tempx = 270 / side; /* available pixels to allocate outlayer nodes */
tempy = 270 / side;

if(win < side) {          /* locate active neuron within output slab */
    winside = win;
    winend = 0;
    }
else {
    winside = win % side;
    winend = win / side;
    }
```

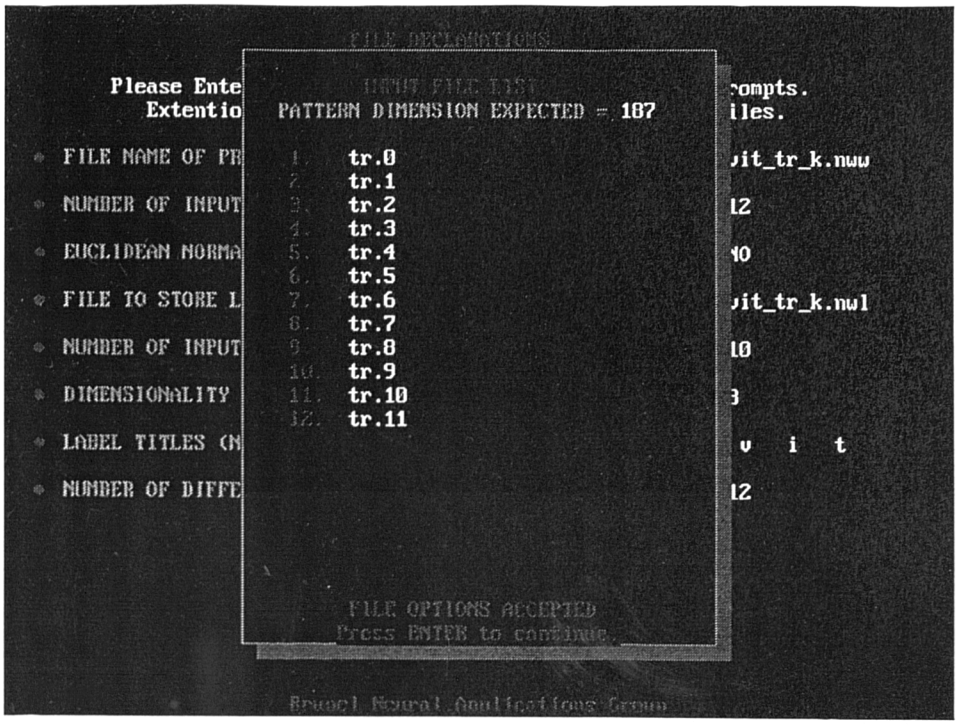
```
/*translate position in slab to pixel coordinates */
y1 += winend * tempY;
x = x1;
y = y1;
x += winside * tempx;

if(labnums > 1 && labfno != 0) { node_size = labfno + 1;}


_setcolor(nodecolour);    /* change winning neuron to appropriate colour */
_rectangle(_GFILLINTERIOR, x + node_size, y + node_size, x + (tempx - node_size), y +
                           (tempY - node_size));

}

} /* end of outmapplot() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/*************************************************************************
*                  GRAPHICS ROUTINES                                    *
*************************************************************************/
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* draw colour code and code vector key */

void labkey (void) {

char mess[10];
unsigned int a, b, c;


_setcolor(3);
_rectangle(_GBORDER, 315, 154, 630, 450);

_settextcolor(GRAPHCOLOR);           /* title */
printat(9, 56, "LEGEND");

for (a = 0, b = 60; a < labdim; a++, b += 7) {
     _settextcolor(3);
     _settextposition(11, b);
     sprintf(mess,"%s", &labnames[a][0]);
     _outtext(mess);
     }

for (a = 0, b = 13; a < labf; a++, b += 1) {
     _settextcolor(TITLECOLOR);
     _settextposition(b, 45);
     sprintf(mess,"%s", &labelfiles[a][0]);
     _outtext(mess);
     }

for (a = 0, b = 4, c = 196; a < labf; a++, c += 16) {
     if(labnums > 1 && a != 0) (b += 1;)
     _setcolor(b);
     _rectangle(_GFILLINTERIOR, 320, c, 327, c + 7);
     }

} /* end of labkey */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* 3. For test only - draw test file list, pattern no & code book vector result */

void restable (void) {

char mess[10];
unsigned int a, b, c;


_setcolor(3);
_rectangle(_GBORDER, 315, 154, 630, 450);

_settextcolor(GRAPHCOLOR);           /* title */
printat(9, 56, "RESULT");

for (a = 0, b = 60; a < labdim; a++, b += 7) {
     _settextcolor(3);
     _settextposition(11, b);
     sprintf(mess,"%s", &labnames[a][0]);
     _outtext(mess);
     }

_settextcolor(GRAPHCOLOR);
printat(11, 42, "   File   Pat No");


for (a = 0, b = 13; a < tesf; a++, b += 1) {
     _settextcolor(TITLECOLOR);
     _settextposition(b, 42);
     sprintf(mess,"%s", &testfiles[a][0]);
     _outtext(mess);
     }


} /* end of restable */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void codemap(void) {

int x, y, a, b, c, z;
int x1, y1;
int tempx, tempy;
```

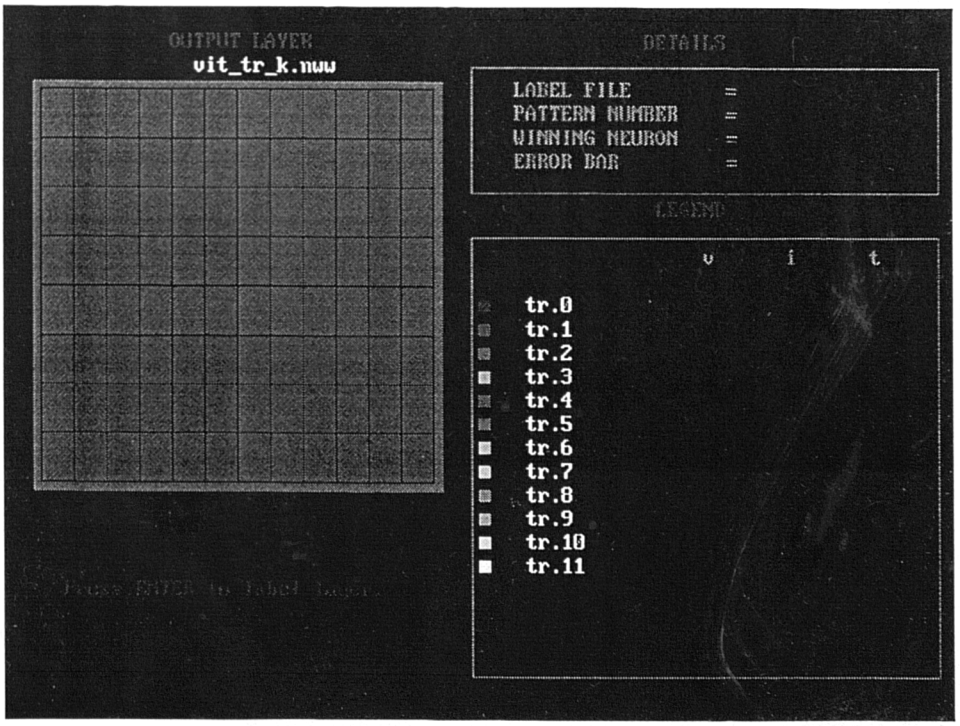
```
int tempX, tempY;

{
tempX = 270 / end;
tempY = 270 / end;
tempx = 270 / side; /* available pixels to allocate outlayer nodes */
tempy = 270 / side;

for(a = 0, c = 0, x = 25, y = 55; a < end; a++, x = 25, y = 55){
  y += a * tempY;
    for(b = 0; b < side; b++, c++, x = 25){
    x += b * tempx;
    nodecolour = (int)out_colour[c];
       _setcolor(nodecolour);      /* change winning neuron to appropriate colour */
       _rectangle(_GFILLINTERIOR, x + 1, y + 1, x + (tempx - 1), y + (tempY - 1));
    }

}


}

} /* end of codemap() */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void identify(void) {

int winend, winside;
int x, y, a, b, c, z;
int x1, y1;
int tempx, tempy;
int tempX, tempY;
int node_size = 1;

{

x1 = 25; /* starting pixel */
y1 = 55;

tempX = 270 / end;
tempY = 270 / end;
tempx = 270 / side; /* available pixels to allocate outlayer nodes */
tempy = 270 / side;

if(win < side) {          /* locate active neuron within output slab */
    winside = win;
    winend = 0;
    }
else {
    winside = win % side;
    winend = win / side;
    }

/*translate position in slab to pixel coordinates */
y1 += winend * tempY;
x = x1;
y = y1;
x += winside * tempx;

for(a = 0; a < 3; a++){        /* flash 3 times */
  for(b = 0; b < 500; b++){
_setcolor(0);     /* change winning neuron to black */
_rectangle(_GFILLINTERIOR, x + 4, y + 4, x + (tempx - 4), y + (tempY - 4));
}

for(b = 0; b < 500; b++){
nodecolour = (int)out_colour[win];
_setcolor(nodecolour);
_rectangle(_GFILLINTERIOR, x + 4, y + 4, x + (tempx - 4), y + (tempY - 4));
}

}
}
} /* end of identify() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*     Wait for an enter key press     */

void Enterkey(void)
 {

int  input[3];
int  ok;

ok = FALSE;

   while(input[0] = getch()) {     /*  Do nothing until user hits enter    */
   switch(input[0]) {
    case '\x0d' : ok = TRUE;
          break;
           }
 if (ok == TRUE) break;

}
}  /*   End Enterkey() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
```

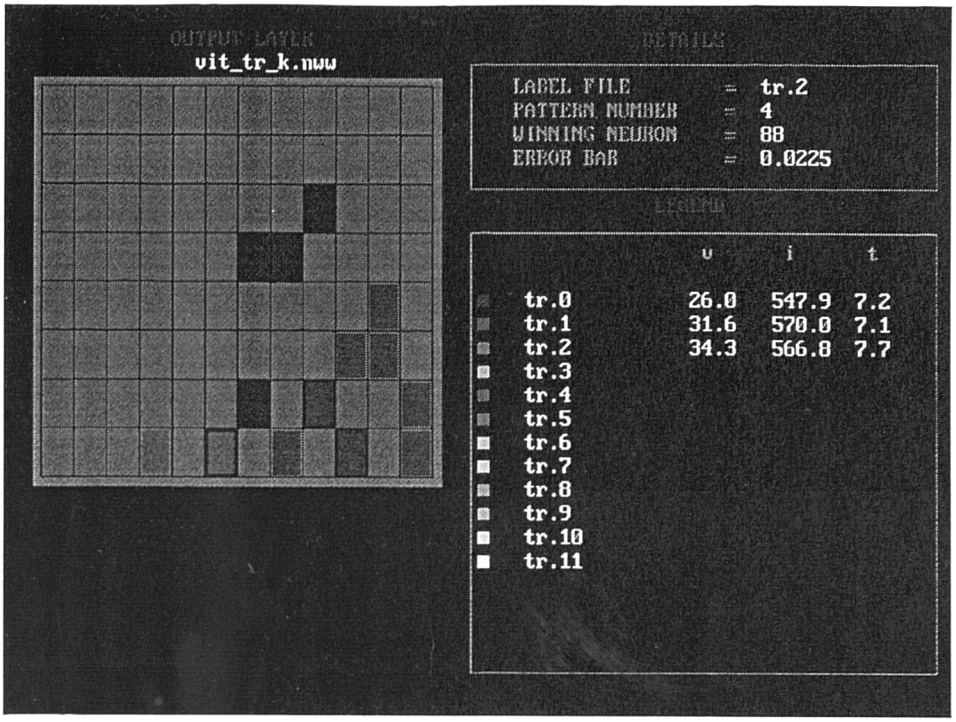
```
 /* Yes or No switch */

void yesno(void) {

int ok;
int input[1];

ok = FALSE;
    while(input[0] = getch())
       {

        switch(input[0]) {
           case 'y':
           case 'Y':ans = TRUE;
                ok = TRUE;
                break;

           case 'n':
           case 'N':ans = FALSE;
                ok = TRUE;
                break;
        }
     if( ok == TRUE) break;
    }

} /* end of yesno */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Set PC overscan register  */
void setborder(short color)
{
    union REGS regs;

    regs.x.ax = 0x01001;    /* Overscan register */
    regs.h.bh = color;
    int86(0x10, &regs, &regs);
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*********************************- FIN -***********************************/
```

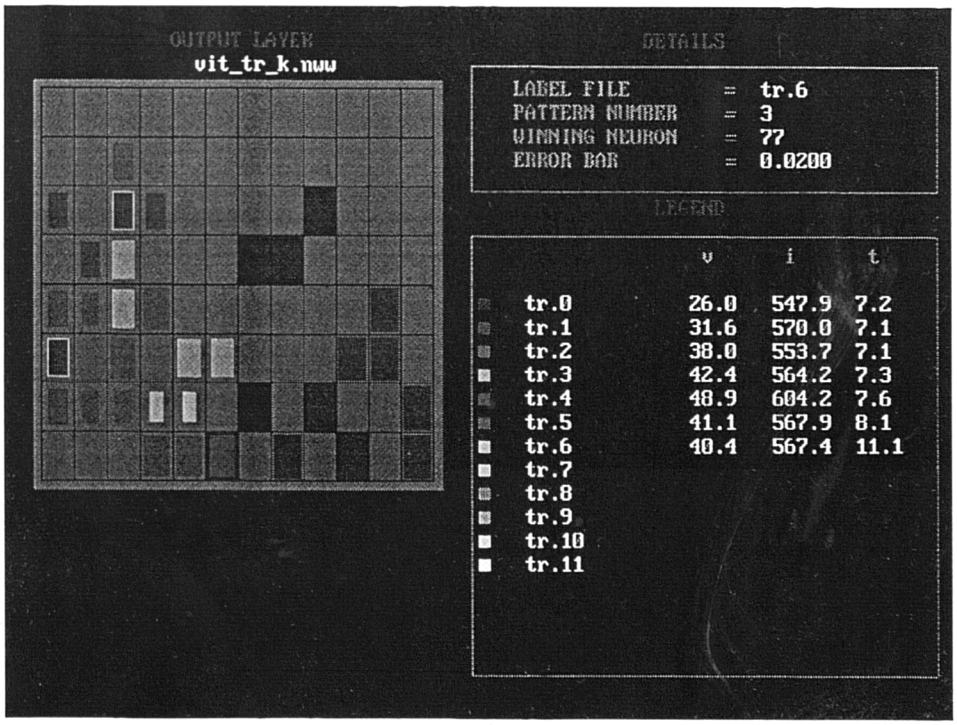

CODE BOOK VECTOR MAPPING
Pre-trained networks are *.nww files. Output layers are labelled with conditioned data files *.nwd with the desired label on the next line following the input pattern. All files are comma separated ascii text.
(R)ETURN to training & test procedure.
(L)ABEL a trained network.
(T)EST a labelled network.
(E)SCAPE to DOS.

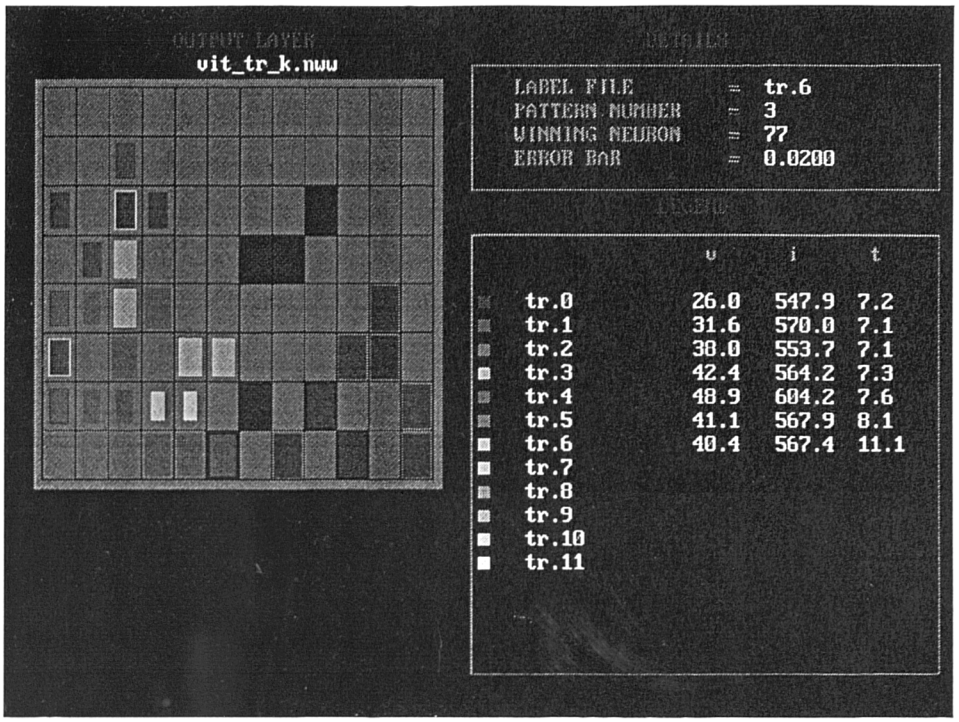

FILE DECLARATIONS
Please Enter the desired file names & paths at the prompts.
Extentions are awarded automatically for OUTPUT files.
FILE NAME OF PRE-TRAINED NETWORK (*.nww) ..... vit_tr_k.nww
NUMBER OF INPUT PATTERN LABEL FILES 12
EUCLIDEAN NORMALISATION REQUIRED (Y/N) ..... NO
FILE TO STORE LABELLED NETWORK (*.nwl automatic)..... vit_tr_k.nwl
NUMBER OF INPUT PATTERNS PER LABEL FILE 10
DIMENSIONALITY OF LABEL VECTOR 3
LABEL TITLES (Number expected = 3) ..... v i t
NUMBER OF DIFFERENT LABELS SUPPLIED 12

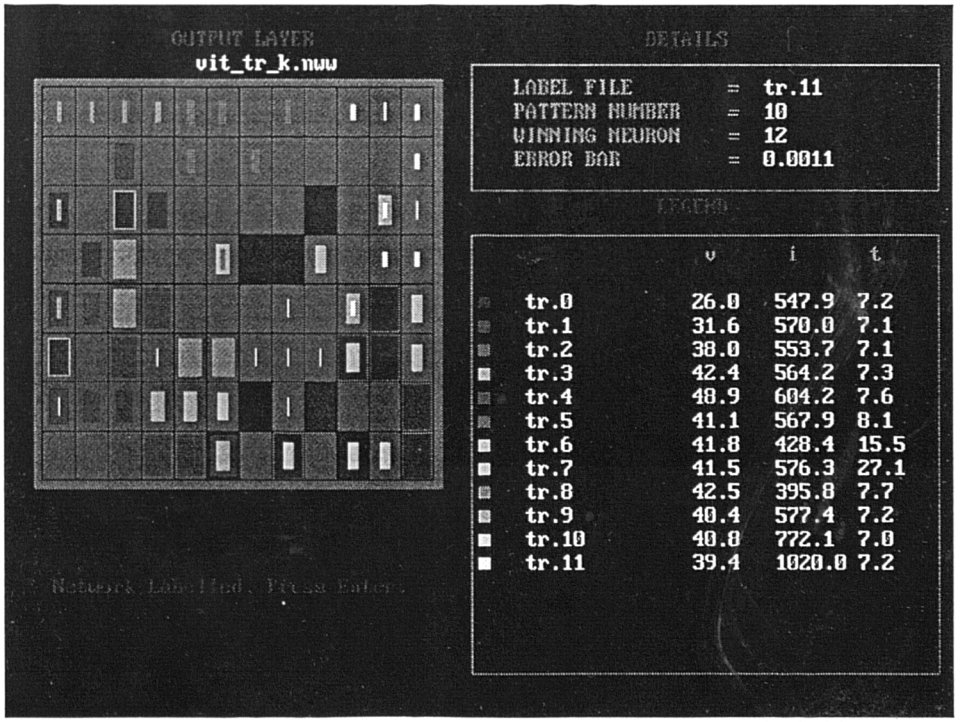

FILE DECLARATIONS
Please Ente
Extentio
compts.
iles.
FILE NAME OF PR
vit_tr_k.nww
NUMBER OF INPUT
12
EUCLIDEAN NORMA
10
FILE TO STORE L
vit_tr_k.nwl
NUMBER OF INPUT
10
DIMENSIONALITY
3
LABEL TITLES (N
v i t
NUMBER OF DIFFE
12
INPUT FILE LIST
PATTERN DIMENSION EXPECTED = 187
1. tr.0
2. tr.1
3. tr.2
4. tr.3
5. tr.4
6. tr.5
7. tr.6
8. tr.7
9. tr.8
10. tr.9
11. tr.10
12. tr.11
FILE OPTIONS ACCEPTED
Press ENTER to continue
Bristol Neural Applications Group

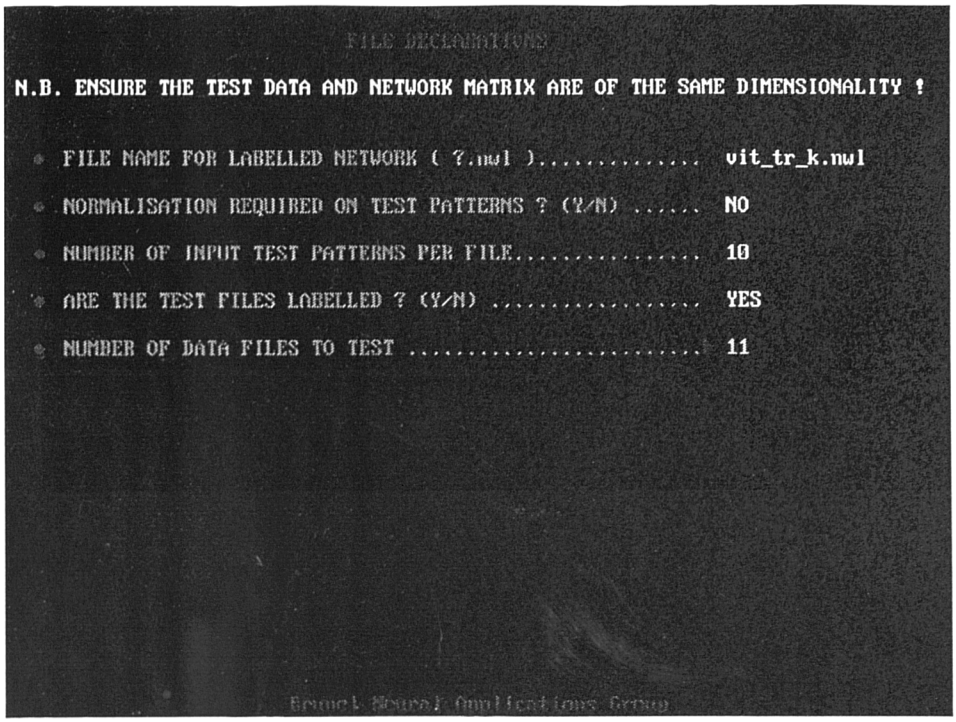

OUTPUT LAYER
vit_tr_k.nww
DETAILS
LABEL FILE =
PATTERN NUMBER =
WINNING NEURON =
ERROR BAR =
LEGEND
v i t
tr.0
tr.1
tr.2
tr.3
tr.4
tr.5
tr.6
tr.7
tr.8
tr.9
tr.10
tr.11

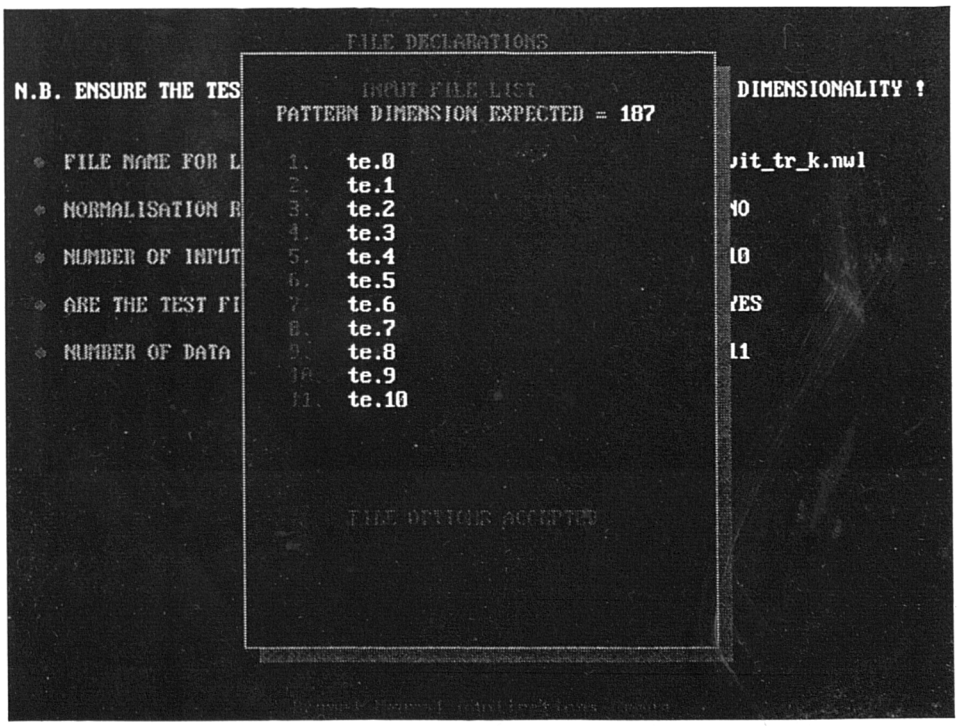

OUTPUT LAYER
vit_tr_k.nww
DETAILS
LABEL FILE = tr.2
PATTERN NUMBER = 4
WINNING NEURON = 88
ERROR BAR = 0.0225
LEGEND
v i t
tr.0 26.0 547.9 7.2
tr.1 31.6 570.0 7.1
tr.2 34.3 566.8 7.7
tr.3
tr.4
tr.5
tr.6
tr.7
tr.8
tr.9
tr.10
tr.11

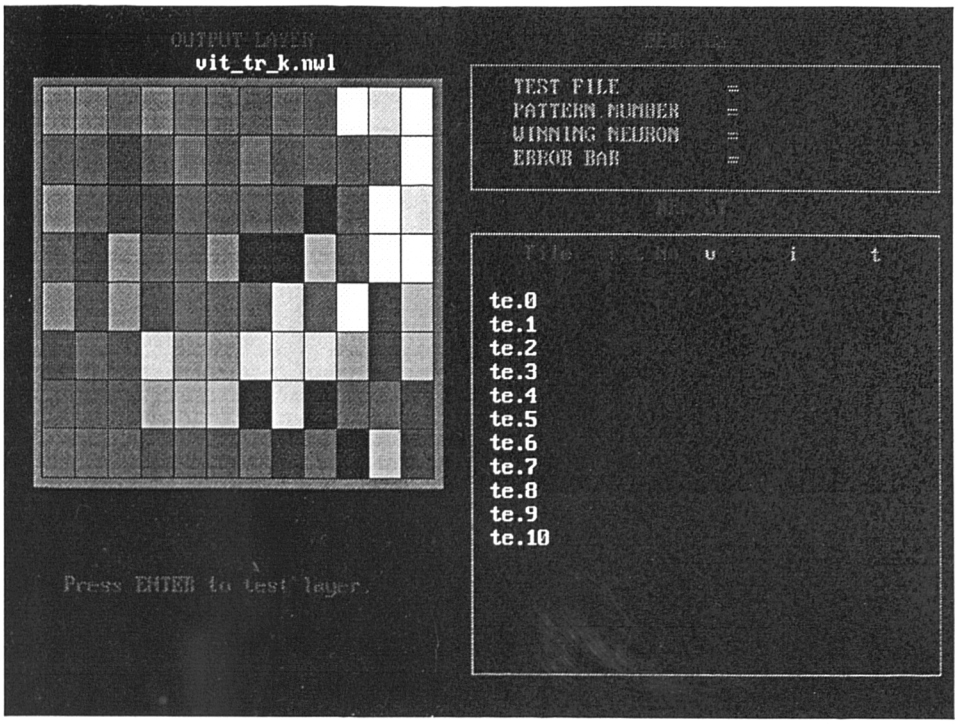

OUTPUT LAYER
vit_tr_k.nww
DETAILS
LABEL FILE = tr.6
PATTERN NUMBER = 3
WINNING NEURON = 77
ERROR BAR = 0.0200
LEGEND
v i t
tr.0 26.0 547.9 7.2
tr.1 31.6 570.0 7.1
tr.2 38.0 553.7 7.1
tr.3 42.4 564.2 7.3
tr.4 48.9 604.2 7.6
tr.5 41.1 567.9 8.1
tr.6 40.4 567.4 11.1
tr.7
tr.8
tr.9
tr.10
tr.11

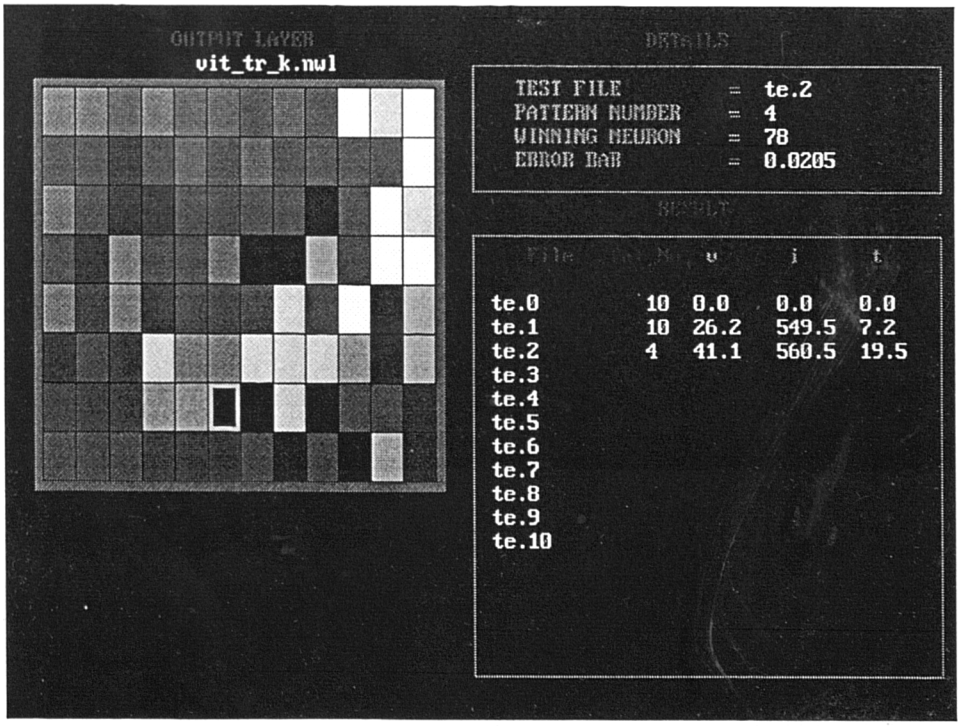

OUTPUT LAYER
vit_tr_k.nwu
DETAILS
LABEL FILE = tr.6
PATTERN NUMBER = 3
WINNING NEURON = 77
ERROR BAR = 0.0200
LEGEND
v i t
tr.0 26.0 547.9 7.2
tr.1 31.6 570.0 7.1
tr.2 38.0 553.7 7.1
tr.3 42.4 564.2 7.3
tr.4 48.9 604.2 7.6
tr.5 41.1 567.9 8.1
tr.6 40.4 567.4 11.1
tr.7
tr.8
tr.9
tr.10
tr.11

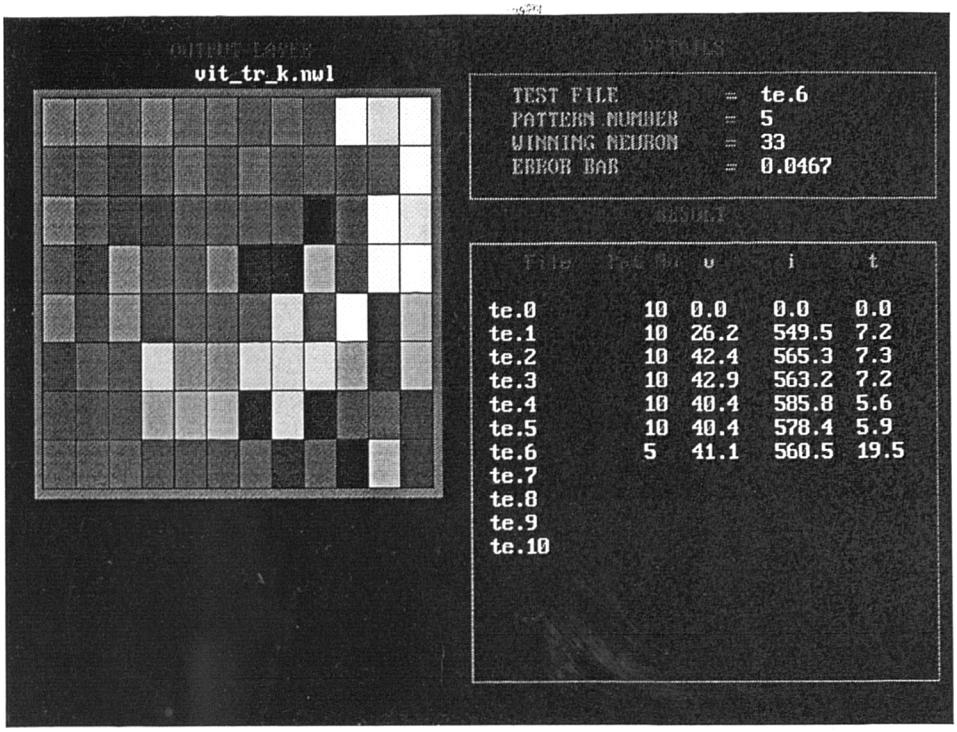

OUTPUT LAYER
vit_tr_k.nwu
DETAILS
LABEL FILE = tr.11
PATTERN NUMBER = 10
WINNING NEURON = 12
ERROR BAR = 0.0011
LEGEND
v i t
tr.0 26.0 547.9 7.2
tr.1 31.6 570.0 7.1
tr.2 38.0 553.7 7.1
tr.3 42.4 564.2 7.3
tr.4 48.9 604.2 7.6
tr.5 41.1 567.9 8.1
tr.6 41.8 428.4 15.5
tr.7 41.5 576.3 27.1
tr.8 42.5 395.8 7.7
tr.9 40.4 577.4 7.2
tr.10 40.8 772.1 7.0
tr.11 39.4 1020.0 7.2
Network Labelled. Press Enter.

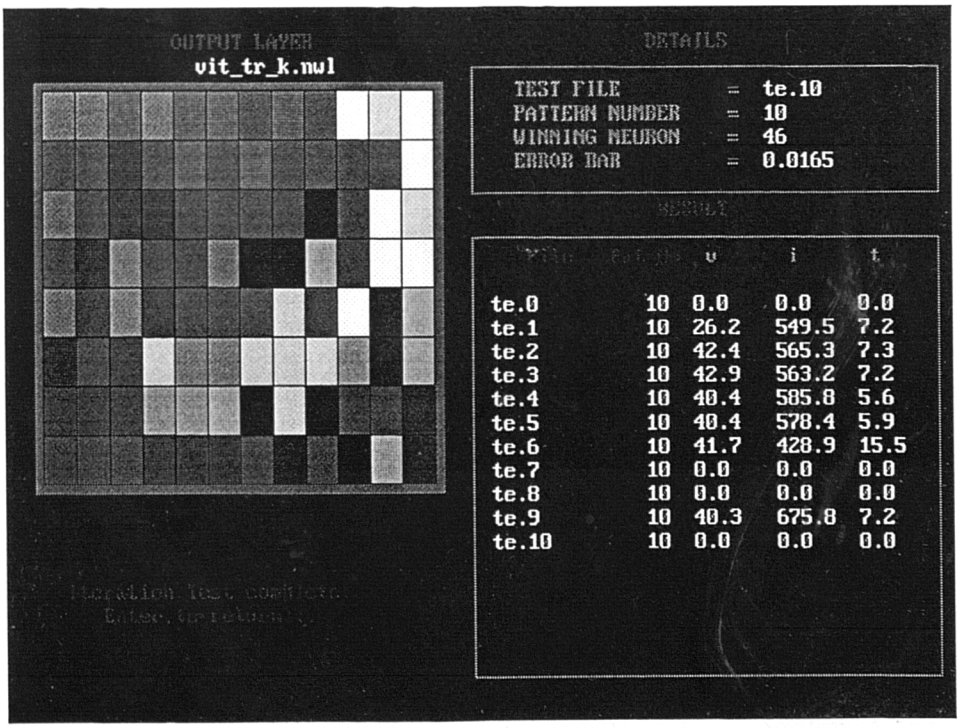

FILE DECLARATIONS
N.B. ENSURE THE TEST DATA AND NETWORK MATRIX ARE OF THE SAME DIMENSIONALITY !
FILE NAME FOR LABELLED NETWORK (?.nwl) ..... vit_tr_k.nwl
NORMALISATION REQUIRED ON TEST PATTERNS ? (Y/N) ..... NO
NUMBER OF INPUT TEST PATTERNS PER FILE..... 10
ARE THE TEST FILES LABELLED ? (Y/N) ..... YES
NUMBER OF DATA FILES TO TEST 11
Brunel Neural Applications Group


FILE DECLARATIONS
N.B. ENSURE THE TES
DIMENSIONALITY !
FILE NAME FOR L
vit_tr_k.nwl
NORMALISATION R
NO
NUMBER OF INPUT
10
ARE THE TEST FI
YES
NUMBER OF DATA
11
INPUT FILE LIST
PATTERN DIMENSION EXPECTED = 187
1. te.0
2. te.1
3. te.2
4. te.3
5. te.4
6. te.5
7. te.6
8. te.7
9. te.8
10. te.9
11. te.10
FILE OPTIONS ACCEPTED


OUTPUT LAYER
vit_tr_k.nwl
TEST FILE =
PATTERN NUMBER =
WINNING NEURON =
ERROR BAR =
File N u i t
te.0
te.1
te.2
te.3
te.4
te.5
te.6
te.7
te.8
te.9
te.10
Press ENTER to test layer.


OUTPUT LAYER
vit_tr_k.nwl
DETAILS
TEST FILE = te.2
PATTERN NUMBER = 4
WINNING NEURON = 78
ERROR BAR = 0.0205
RESULT
File N u i t
te.0 10 0.0 0.0 0.0
te.1 10 26.2 549.5 7.2
te.2 4 41.1 560.5 19.5
te.3
te.4
te.5
te.6
te.7
te.8
te.9
te.10

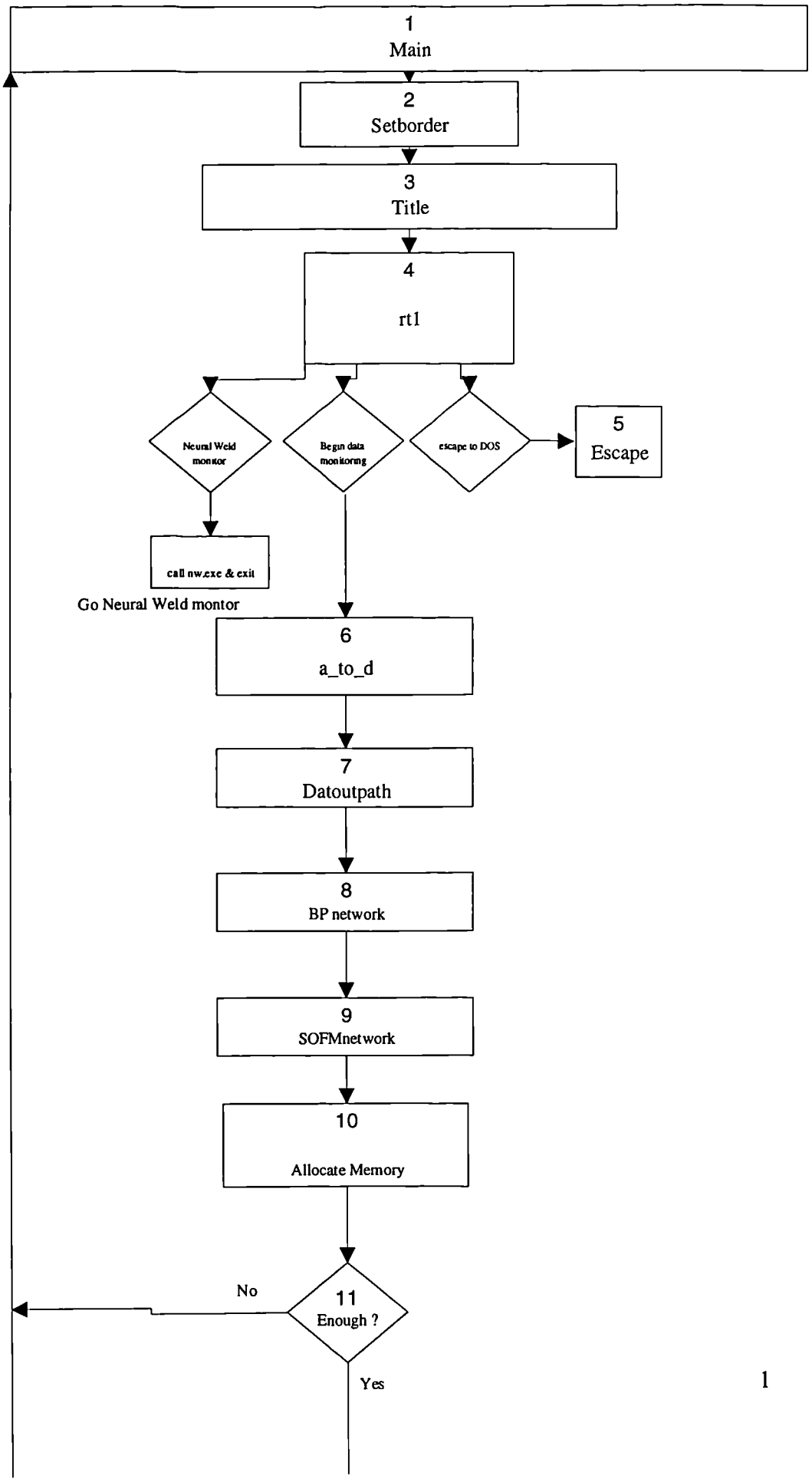

OUTPUT LAYER
vit_tr_k.nwl
DETAILS
TEST FILE = te.6
PATTERN NUMBER = 5
WINNING NEURON = 33
ERROR BAR = 0.0467
RESULT
File Pat. Nr. v i t
te.0 10 0.0 0.0 0.0
te.1 10 26.2 549.5 7.2
te.2 10 42.4 565.3 7.3
te.3 10 42.9 563.2 7.2
te.4 10 40.4 585.8 5.6
te.5 10 40.4 578.4 5.9
te.6 5 41.1 560.5 19.5
te.7
te.8
te.9
te.10

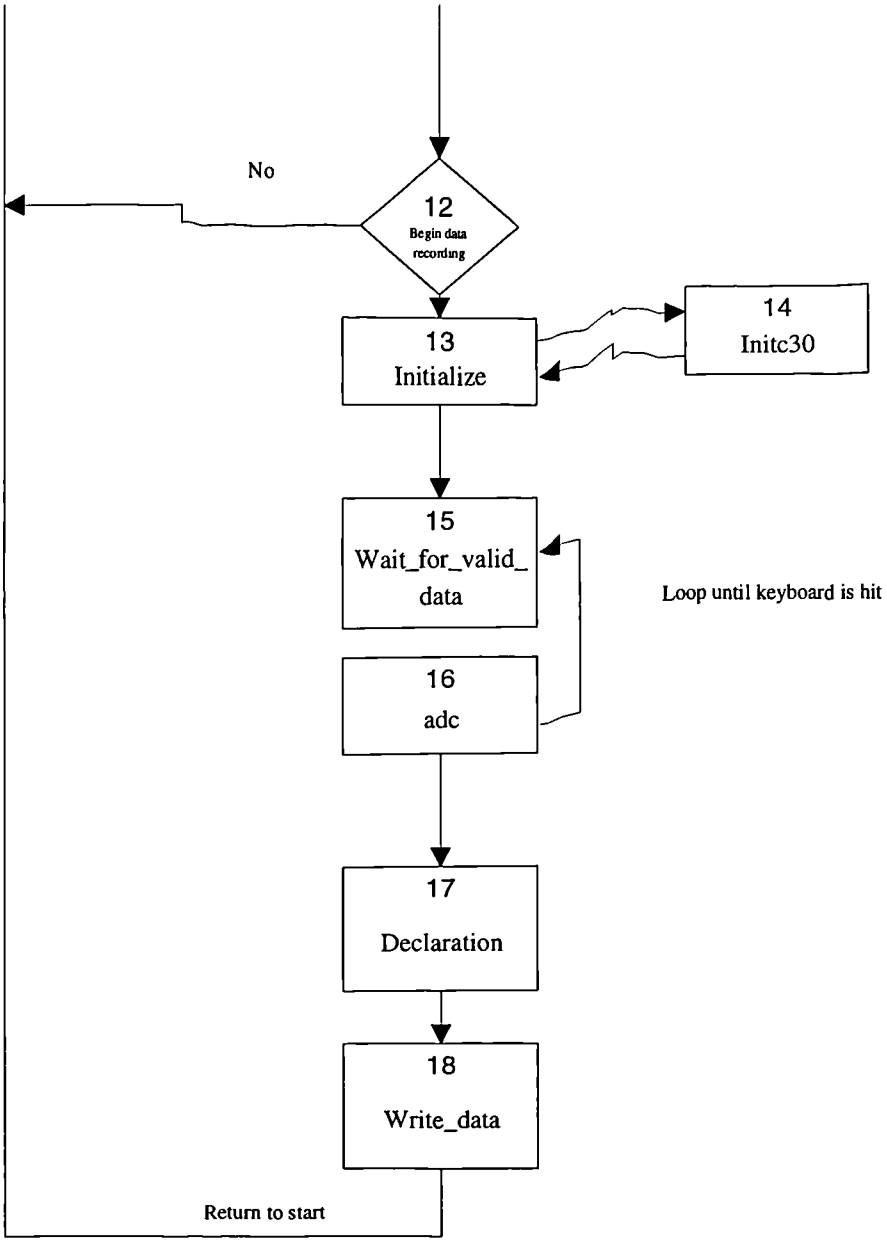

OUTPUT LAYER
vit_tr_k.nwl
DETAILS
TEST FILE = te.10
PATTERN NUMBER = 10
WINNING NEURON = 46
ERROR BAR = 0.0165
RESULT
File Pat. Nr. v i t
te.0 10 0.0 0.0 0.0
te.1 10 26.2 549.5 7.2
te.2 10 42.4 565.3 7.3
te.3 10 42.9 563.2 7.2
te.4 10 40.4 585.8 5.6
te.5 10 40.4 578.4 5.9
te.6 10 41.7 428.9 15.5
te.7 10 0.0 0.0 0.0
te.8 10 0.0 0.0 0.0
te.9 10 40.3 675.8 7.2
te.10 10 0.0 0.0 0.0

APPENDIX 5

Source Code for PC Based Weld Monitor

Flow Chart for Weld Monitoring Software (RTMON.c)





FUNCTIONS KEY

2. void setborder(short color); Sets screen graphics boarder
3. void title(void); Draws title page
4. void rtl(void); Gives information
5. void escape(void); Escapes program to DOS
6. void a_to_d(void); Takes information from user interface to initiate adc. process parameter logging
7. void datoutpath(void); Sets up output path and file name for recorded data
8. Void BPnetwork(void) Loads trained BP network
9. void SOFMnetwork(void) Loads weights from trained sofm network
10. void allocate memory(void) Check memory space and allocate.
13. void initialize(void); Initializes global variables , screen parameters and TMS320C30 board
14. void initc30(void); Initiates TMS320C30 board and downloads fft program
15. void wait_for_valid_data(void); Holds program untill valid data from TMS320C30 board is.. present
16. void adc(void); Reads adc board scales data and loads in memory
18. void plot(void); Reads TMS320C30 board
17. void declaration(void); Accepts or rejects acquired data
18. void write_data(void); Writes recorded data to output file

```
/*************************************************************************
 * This file RTMON.C is the real time acoustic monitor for the NEURAL-WELD  *
 * system. It accepts a pretrained kohonen network generated by SOFM.EXE    *
 * and a pretrained BP network generated bt Neuralworks P2. On line FFTs    *
 * are gathered by the TMS320C30 via the fft.out assembler file, generating *
 * a 128 point (6th order) FFT. The output of the BP is directed to the     *
 * screen as a measure (absolute) of the parameter on which it was trained  *
 *                                                                          *
 * Compiled by J.R. McCardle, Neural Applications Group, Brunel University  *
 *                                                                          *
 *************************************************************************/

#include <graph.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <dos.h>
#include <process.h>
#include <errno.h>
#include <share.h>
#include <math.h>
#include <tms30.h>          /* LSI C30  Interface Library. */

#include"nn_run.h"             /* back prop run time headers */
#include"bp_frq3.h"

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Board parameters:*/

#define BOARDADR        0x290     /* Default I/O Base address of board */

#define BOARD_BUFF_ADR  0x3e010   /* Data output buffer on C30 board */
#define VALID_DATA_FLAG 0x3e00a   /* Flag TRUE when valid data on C30 board */
#define TIMERVAL    0x3e00b   /* Loc of value put into SPOX timer */

#define ALPHA       0x3e00d   /* Location of filter (integration) */
                /* coefficient in C30's memory */
#define SCALE       0x3e00e   /* Location of screen scale factor    */
#define OFFSET      0x3e00f   /* Location of screen offset factor   */


#define BASEOFFSET  58.0      /* Starting offset, in dB, for plot */
                /* of spectral data */

/* Use intrinsic versions of outp and inp */
#pragma intrinsic( outp, inp )

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* BYTRONIC MPIBM6 I/O Board parameters:*/

#define BASEADDRESS 512             /* Base address for i/o board in trigem pc */
#define DATA    BASEADDRESS + 16    /* write address of adc */
#define BUSY    BASEADDRESS + 20    /* busy line address of adc */

#define VOLTS_CHAN  10              /* channels on single ended mode */
#define AMPS_CHAN   11
#define SPEED_CHAN  12

/* Use intrinsic versions of outp and inp */
#pragma intrinsic( outp, inp )

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*  Pretty colour bits:*/

#define BACKGNDCOLOR    0   /* black */
#define AXISCOLOR   3       /* cyan */
#define GRAPHCOLOR  4       /* red */
#define TITLECOLOR  14      /* yellow */
#define LABELCOLOR  15      /* white */
#define printat(x, y, z)    _settextposition(x, y);\
              _outtext(z);
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Standard defines:*/

#define TRUE        1
#define FALSE       0

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Screen graphics and labeling parameters:*/

#define DATALENGTH  64  /* Length of spectral data array.   */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Function Prototypes:*/

typedef float *PTR_FLOAT;
typedef PTR_FLOAT VECTOR;
typedef PTR_FLOAT *MATRIX;

int vectormem(VECTOR *ptr_vector, int Ncolumns);            /* Allocates memory for vectors */
void columnmem(PTR_FLOAT matrix[], int Nrows, int Ncolumns);/* Allocates memory for columns in..*/
                                                            /* matrix */
int matrixmem(MATRIX *ptr_matrix, int Nrows, int Ncolumns); /* Allocates memory for variable matrix */
void matrixdel(MATRIX matrix, int Nrows, int Ncolumns);     /* Releases all memory allocated to..*/
void memclear(void);          /* Calls all procedures for freeing allocated memory */
```

```
                                        /* matrices */


void     initialize(void);
void     initc30(void);
void     wait_for_valid_data(void);


void     rt1(void);
void     listen(void);
void     fftgen(void);
void     escape(void);
void     Enterkey(void);
void     write_data(void);          /* Writes recorded data to output file */
void     declaration(void);         /* Accepts or rejects acquired data */
void     adc(void);                 /* Reads adc board scales data and loads in memory */
void     setborder(short color);

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  */

/*  Global Variables: most variables are kept global for speed */

GLOBALDEF NET *mptr;

SREAL iv[187];            /* Input Vector (Activation Map) for BP network) */
SREAL rv[4];              /* Output Vector of BP network (Volts, Amps, mm/s, kJ/mm) */

float    alpha;           /* Sq. magnitude filter coeff */
float    scale;           /* Used by C30 to scale screen values  */
float    offset;          /* Used by C30 to offset screen values */
float    syn_weights[187][64];   /* set sofm dimension */
float    fftout[50][DATALENGTH]; /* Size of I/O data array */
float    voltage;
float    current;
float    speed;
float    energy;

float    volts;           /* Actual float scale values of process parameters */
float    amps;
float    tspeed;


int volts_byte;      /* 8 bit descriptors for process parameters */
int amps_byte;
int speed_byte;

int      side;       /* Side dimension of output layer  */
int      end;        /* End dimension of output layer  */
int      win;

long     timerval;        /* Holds hex val put into sample timer */
long     intdata[DATALENGTH];     /* Data arrays for FFT and recorded data */
long     innodes;         /* Number of input nodes */
long     inpatts;         /* Number of input patterns */
long     outnodes;        /* Number of ouput nodes */

FILE     *infile;    /* Pointers for recorded and I/O data */
FILE     *outfile;        /* Pointer for recorded data */
FILE     *headfile;       /* Pointer to data header file */
/************************************************************************/

void main(void) {

_setvideomode(_MAXRESMODE);
_setbkcolor(_BLACK);
setborder(11);


rt1();   /* title explanation */

BP_Init(&mptr, (NET*)0,1); /* initiate back prop network */

if((LoadNet ("frq3_te.nnd",0)) != NN_OK) {  /* open ascii network file */
    printf("\n error opening *.nnd file");
    Enterkey();
    }

listen();  /* press enter to begin */

initialize();  /* initialise tms320c30 board */

wait_for_valid_data();  /* wailt for valid fft generation */

while(TRUE) {             /* Loop until exit with escape() */
               /* gives constant FFT output whilst idling */
fftgen();

BP_Recall(mptr, iv, rv);  /* run backprop network and retrieve result */

printf("\n volts = %6.2f", rv[0]); /* output BP to screen */
printf("\n current = %6.2f", rv[1]); /* output BP to screen */
printf("\n speed = %6.2f", rv[2]); /* output BP to screen */
printf("\n energy = %6.2f", rv[3]); /* output BP to screen */

write_data();

if ( kbhit() ){declaration();}      /* If keyboard is hit, look for command from user.*/


    escape();
}
```

```
} /* end of main */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void rt1(void) {

char inpathweights[_MAX_PATH];
char drive[_MAX_DRIVE], dir[_MAX_DIR];
char fname[_MAX_FNAME], ext[_MAX_EXT];
char mess[45];
int  c;
int  i;

float r;


_settextcolor(GRAPHCOLOR);
printat(2,25, " REAL TIME ACOUSTIC MONITOR");
printat(30,24," Brunel Neural Applications Group");

_settextcolor(TITLECOLOR);
printat(5, 0, " The Real Time Acoustic Monitor is a Neural pattern associator which ");
printat(6, 0, " compares acquired FFT frames with a pre-trained network.");
printat(7, 0, " The output is a direct measurement of the parameter on which it is");
printat(8, 0, " trained.");
printat(10, 0, " Enter the files for the pre-trained network at the prompt....");

_settextcolor(AXISCOLOR);
printat(13, 5, " FILE NAME FOR PRE-TRAINED SOFM MATRIX ( ?.nww ).......");
_settextcolor(GRAPHCOLOR);
_ellipse(_GFILLINTERIOR,20,196,27,203);

/* ask for pre-trained weight matrix */
_settextcolor(GRAPHCOLOR);
_settextposition(13,60);
_outtext("\x10");
_settextposition( 13, 62);

flushall();   /* clear buffer */

   do{
     _settextposition(13,62);
     gets(inpathweights);
     _settextcolor(TITLECOLOR);
     _outtext(inpathweights);

     if( (infile = fopen( inpathweights, "rb")) == NULL ) {  /* error message */
     _settextcolor(GRAPHCOLOR);
     printf("\a"); /* Alarm */
     sprintf( mess, " Can't open weights file, error number: %d", errno);
     printat(17, 18, mess);
     printat(18, 16, " Drive and/or Directory has to exist to use !");
     printat(20, 20, " Press ENTER to repeat declaration.");

     Enterkey();

     /* Erase error message */
    printat(17, 18, "                                                  ");
    printat(18, 16, "                                                  ");
    printat(20, 20, "                                                  ");
    printat(13,62, "                    "); /* Erase path */

    }

 } while( infile == NULL);  /* Loop declaration until valid */


_settextcolor(GRAPHCOLOR);
printat(22, 27, "PROCESSING DATA PLEASE WAIT");

fscanf(infile, "%ld\n%ld\n%d\n%d\n", &outnodes, &innodes, &side, &end); /* read header */

matrixmem(&syn_weights, (short)outnodes, ((short)innodes + 1));/* Allocate memory for matrix */
if (rep == 1) {
          _settextcolor(TITLECOLOR);
          printat(29, 27, "SORRY!! NOT ENOUGH MEMORY");
          matrixdel(syn_weights, (short)outnodes, ((short)innodes +1));
          Enterkey();
          fclose(infile);
          escape();

          }

for(i = 0; i < outnodes; i++) {      /* read data */
    for(c = 0; c < innodes; c++) {
    fscanf(infile, "%f,", &r);
    syn_weights[i][c] = r;
      }
       fscanf(infile, "\n\r");
 }

fclose(infile);

printat(22, 27, "                          ");

printat(13,60, " ");  /* Erase delta prompt */

}  /* end of title() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void listen(void) {

printat(18, 16, " Press Enter to begin listen procedure.....");
```

```
Enterkey();

} /* end of listen */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void fftgen(void) {

float d;
float Euc_Dist;
float Small_Euc_Dist;

long    *intdataptr, temp;
int i, c, o, n;

for(o = 0; o < outnodes; o++) {     /* set Yin components to zero */
      iv[o] = 0.0;
     }

for(c = 0; c < 50; c++) {

RdBlkInt(BOARD_BUFF_ADR, DUAL, DATALENGTH, intdata);  /* Get data from board */

intdataptr = intdata;


for (i = 0; i < DATALENGTH; i++) {
      temp = (*intdataptr++);
      fftout[c][i] = (float)temp;
      }

              /* locate winning neuron */

Small_Euc_Dist = 0.0;

for(o = 0; o < outnodes; o++) {

    Euc_Dist = 0.0;

    for(n = 0; n < DATALENGTH; n++) {
    d = fftout[c][n] - syn_weights[o][n];
    Euc_Dist += d * d;
     }

    if(o == 0 || Euc_Dist <= Small_Euc_Dist) {
       Small_Euc_Dist = Euc_Dist;
       win = o;
     }
}

/* form iv[96], frequency map input vector for BP network */

iv[win] += 2.0;   /* increment and scale frq map */

} /* end of 50 fft frame grab */


} /* end of fftgen() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void declaration(void){

int ok = FALSE;
char    dat[12];
int input[3];


_settextcolor(4);
printat(18, 30, " ACCEPTANCE DECLARATION ");
_settextcolor(3);
printat(20, 11, "  If you are NOT happy with the recording you can repeat it.");
printat(21, 11, "  Re-typing the same file name WILL overwrite the existing ");
printat(22, 11, "                                file.");

_settextcolor(AXISCOLOR);
printat(26,26, " ( )CCEPT data file/s and return to start.");
printat(28,26, " ( )EPEAT or record another file.");
_settextcolor(GRAPHCOLOR);
printat(26,28, "A");
printat(28,28, "R");


while(input[0] = getch())
{

    switch(input[0]) {
    case 'a':
    case 'A':
          fclose(outfile);
          ok = TRUE;
          accept = TRUE;     /* Set accept recording flag */
          break;

    case 'r':
    case 'R':
          ok = TRUE;
          accept = FALSE;
          fclose(outfile);
```

```
        )
    if (ok == TRUE) break;
}


} /* end of declaration() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void write_data(void){

float v;
int i;


/* transfer data to named file */

for(i = 0; i < 4; i++) {
    fprintf(outfile, "%f,", rv[i]);
    }

fprintf(outfile, "%.2f,%.2f,%.2f,%.2f", voltage, current, speed, energy);


} /* end of write_data */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void adc (void){

float scale_volts = 0.465F;  /* 0-50v scale range */
float scale_amps = 12.0F;    /* scale range offset */
float scale_speed = 0.4F;    /* 0-28mm\s scale range over 2v max range*/


outp(DATA, VOLTS_CHAN);

while((inp(BUSY) & 1) == 1);

volts_byte = inp(DATA);
volts =  ((float)volts_byte * scale_volts) + 0.6;
if(volts < 2){volts = 0.0;}
_settextposition(14, 13);
printf("       ");
_settextposition(14, 13);
printf("\n %.2f V", volts);

voltage = volts;

          /* - - - - - - - - - - - - - - - - - - - */

outp(DATA, AMPS_CHAN);

while((inp(BUSY) & 1) == 1);

amps_byte = inp(DATA);
amps =  ((float)amps_byte - scale_amps);
amps /= 0.038F;                          /* calibrated linear scale factor y=mx+c (m=0.038) */
_settextposition(14, 37);
printf("         ");
_settextposition(14, 37);
printf("\n %.2f A", amps);

current = amps;

        /* - - - - - - - - - - - - - - - - - - - */

outp(DATA, SPEED_CHAN);

while((inp(BUSY) & 1) == 1);

speed_byte = inp(DATA);
speed = ((float)speed_byte * scale_speed) + 3.5;
if(speed < 4.2){speed = 0.0;}
_settextposition(14, 62);
printf("           ");
_settextposition(14, 62);
printf("\n %.2f mm\\s", speed);

speed = tspeed;

/* calculate energy */

energy = (voltage*current)/ (1000*speed);
printf("\n %.2f kJ\\mm", energy);

} /* end of adc */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void datoutpath (void) {

char  outpath[_MAX_PATH];
char drive[_MAX_DRIVE], dir[_MAX_DIR];
char fname[_MAX_FNAME], ext[_MAX_EXT];
char mess[45];

do {

/* Ask for drive and file declaration */
_settextcolor(4);
```

```
printat(18, 26, " DRIVE AND FILE DECLARATION ");
_settextcolor(3);
printat(20, 28, " Give path to store data.");
printat(21, 23, " An extention of .NWD is automatic.");

printat(22, 18, " Enter Output drive, directory and file name. ");
printat(23, 17, " Default is the given name in home directory :-");

_settextposition(25, 32);  /* Display cursor to prompt file output path */
_settextcolor(GRAPHCOLOR);
_outtext("\x10");

flushall();
_settextposition(25, 34);
gets( outpath );

_splitpath( outpath, drive, dir, fname, ext);   /* create output path */
strcpy ( ext, "nwd" );
_makepath( outpath, drive, dir, fname, ext );

if( (outfile = _fsopen( outpath, "wb", SH_DENYWR )) == NULL ){  /* error message */
    printf("\a");
    sprintf( mess, " Unable to open,(err: %d)", errno);
    _outtext(mess);
    Enterkey();
    printat(25,34, "                          "); /*erase error message */
}

}while( outfile == NULL );  /* Loop declaration until valid */

_settextcolor(TITLECOLOR);
_outtext(outpath);
printat(25,32, " "); /* erase delta cursor */

}  /* End of Datoutpath */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Initialize global variables, screen parameters, and the C30 board.   */

void initialize(void)
{

initc30();          /* Initialize the C30 board by down-*/
              /* loading and starting FFT.OUT .*/
timerval = 417;          /* Timer value (20KHz sampling).*/
PutInt(TIMERVAL,DUAL,timerval); /* Enter timer value to C30 board. */
              /* (freq. range is 1/2 sample rate.)     */
alpha = (float)0.9;        /* Default sq mag filter coefficient.   */
scale = (float)23.0;        /* The screen displays 2.3db per pixel*/

offset = -((scale / 10.0) * (BASEOFFSET + 10 * log10( 1.0/(1.0-alpha) ) ))*2.30;

PutFloat(ALPHA, DUAL, alpha);          /* Update value on C30 board */

PutFloat(OFFSET, DUAL, offset);        /* Update value on C30 board */

} /* End of c30 initialisation */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void initc30(void)
{
    SelectBoard(BOARDADR);      /* Set I/O base address: */

    Hold();             /* Hold processor */
    LoadObjectFile("fft.out");  /* Download C30 program */
    Reset();                    /* Reset & run C30 code */

    *(((int *)VALID_DATA_FLAG)) = 0;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void wait_for_valid_data(void)
{
    while (GetInt(VALID_DATA_FLAG, DUAL) == 0);
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


/* ROUTINES FOR MEMORY ALLOCATION */

int vectormem(VECTOR *ptr_vector, int Ncolumns)
{
{    *ptr_vector = (VECTOR) calloc(Ncolumns, sizeof(float));

    if( *ptr_vector == NULL)
    {                                          /* sets memory allocation for vectors */
     _settextcolor(GRAPHCOLOR);
   printf("\a");
   printat(27, 22, "MEMORY TOO SMALL FOR VECTOR ALLOCATION");
   printat(28, 18, "Press ENTER and try smaller number of patterns.");
   rep = 1;
   Enterkey();
    }

}
}  /* end of vectormem */
        /* - - - - - - - - - - - - - - - - */
```

```
int matrixmem(MATRIX *ptr_matrix, int Nrows, int Ncolumns)
{
{   *ptr_matrix = (MATRIX) calloc(Nrows, sizeof(float));

   if(*ptr_matrix == NULL)
   {                                          /* sets memory allocation for matrices */
   _settextcolor(GRAPHCOLOR);
   printf("\a");
   printat(27, 22, "MEMORY TOO SMALL FOR MATRIX ALLOCATION");
   printat(28, 18, "Press ENTER and try smaller nodal combinations.");
   rep = 1;
   Enterkey();
   }

columnmem(*ptr_matrix, Nrows, Ncolumns);

}
}   /* end of matrixmem */

         /* - - - - - - - - - - - - - - - - */

void columnmem(PTR_FLOAT matrix[], int Nrows, int Ncolumns)
{
int i;
      for(i = 0; i < Nrows; i++) {             /* sets up column matrix */
       if(rep != 1) {
      vectormem(&matrix[i], Ncolumns);
       }
}
}    /* end of columnmem */

        /* - - - - - - - - - - - - - - - - */

void matrixdel(MATRIX matrix, int Nrows, int Ncolumns)
{
int i, a;
      for(i = 0; i < Nrows; i++){               /* deletes all matrices */
     for(a = 0; a < Ncolumns; a++){
      free(matrix);
     }
    free(matrix[i]);
      }
}    /* end of matrixdel */


/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

void memclear(void)
{

             /* FREE ALL MEMORY ALLOCATIONS */

matrixdel(inlayer, (short)inpatts, ((short)innodes + 1));
matrixdel(outlayer, (short)inpatts, (short)outnodes);
matrixdel(syn_weights, (short)outnodes, ((short)innodes + 1));

}  /* end of memclear() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/* Escape program to DOS */

void escape(void)
{
int esc[1];
esc[0] = getch();

switch(esc[0])
    {
            /* Escape Key */
    case '\x1b':    _setvideomode(_DEFAULTMODE);
            _displaycursor(_GCURSORON);
            _settextposition(25,81);
            exit(0);
            break;


    default:    break;

    }  /* End of switch */

}       /* End of escape(). */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

/*     Wait for an enter key press     */

void Enterkey(void)
 {

char  input[3];
int  ok;

ok = 0;

   while(input[0] = getch()) {     /*  Do nothing until user hits enter    */
   switch(input[0]) {
    case '\x0d' : ok = 1;
```

```
          break;
            }
 if (ok == 1) break;

}
}  /*   End Enterkey() */

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  */

/*  Set PC overscan register */

void setborder(short colour)
{
    union REGS regs;

    regs.x.ax = 0x01001;     /* Overscan register */
    regs.h.bh = colour;
    int86(0x10, &regs, &regs);
}

/********************************** THE END ***************************/
```

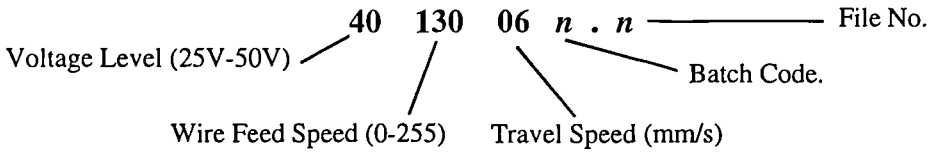
APPENDIX 6

Typical 1st Level FFT Training Data

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

KEY TO FILE NAMES



7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

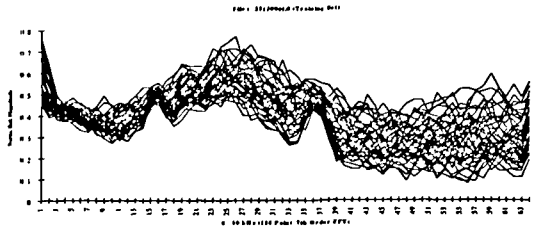


Figure. 1

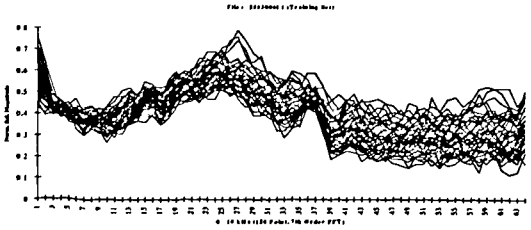


Figure. 2

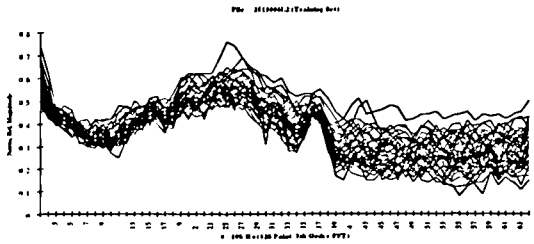


Figure. 3

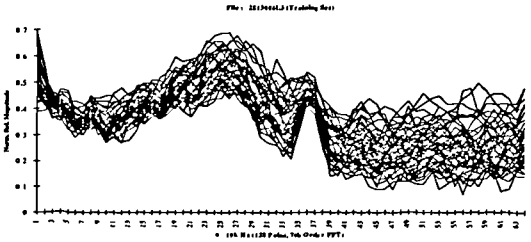


Figure. 4

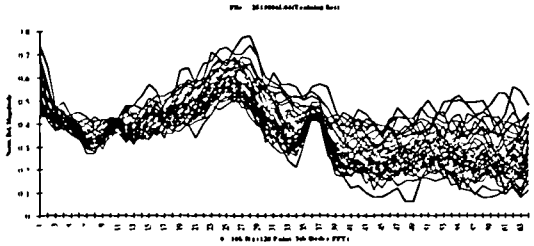


Figure. 5

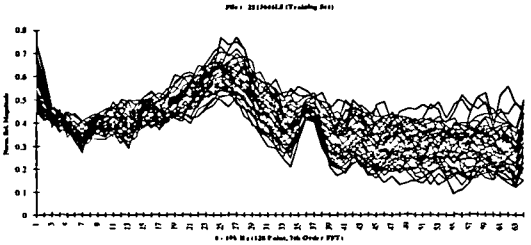


Figure. 6

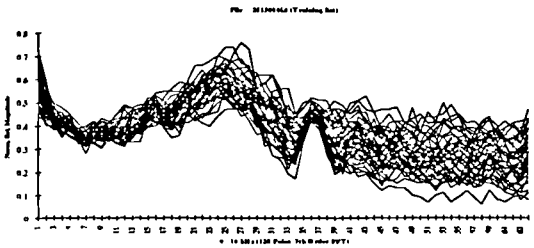


Figure. 7

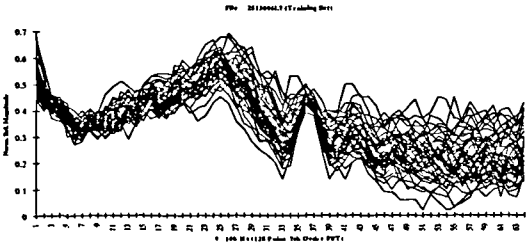


Figure. 8

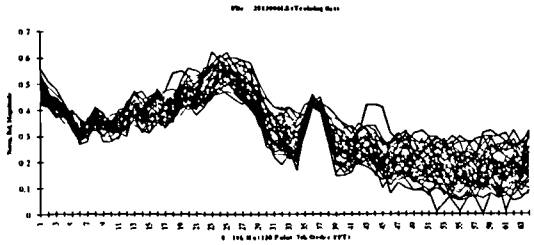


Figure. 9

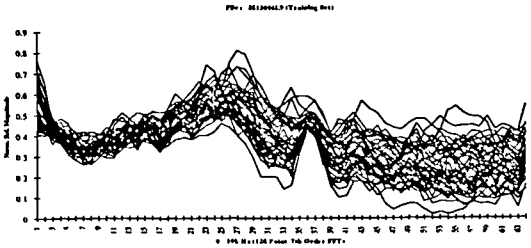


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

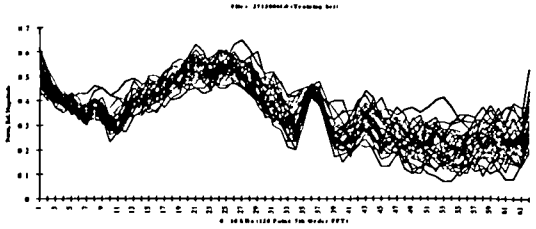


Figure. 1

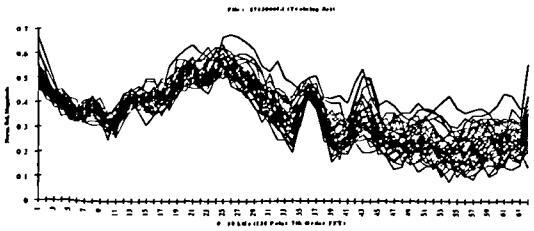


Figure. 2

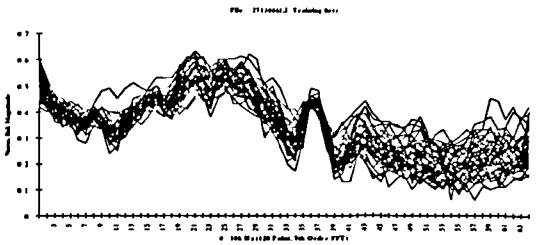


Figure. 3

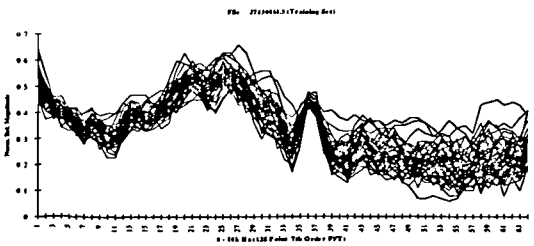


Figure. 4

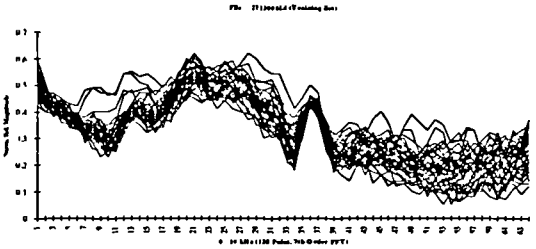


Figure. 5

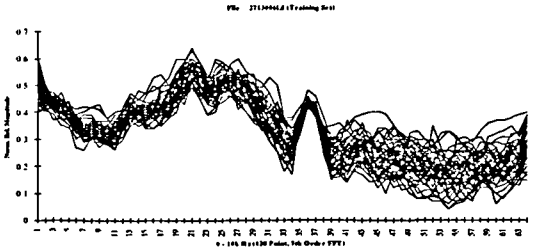


Figure. 6

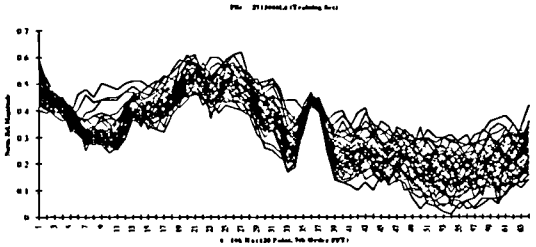


Figure. 7

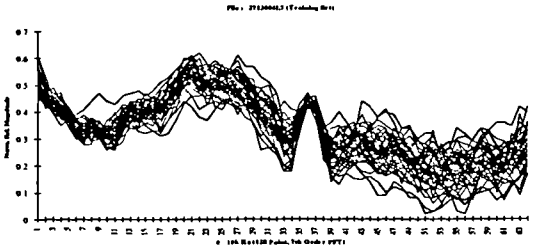


Figure. 8

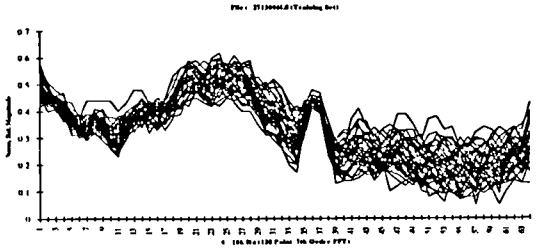


Figure. 9

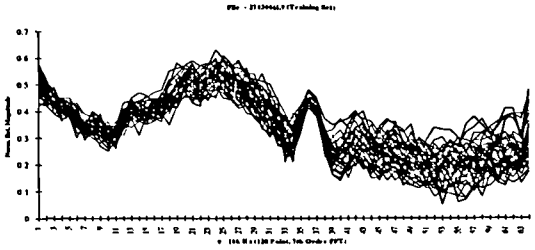


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

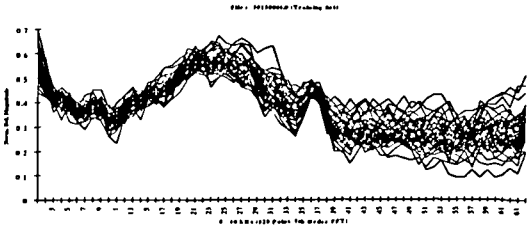


Figure. 1

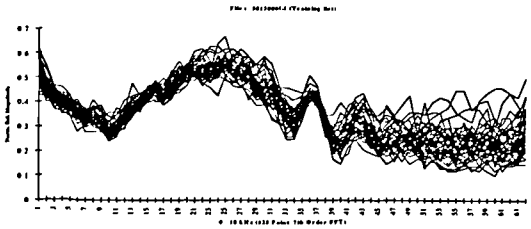


Figure. 2

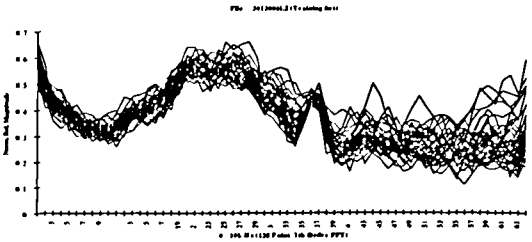


Figure. 3

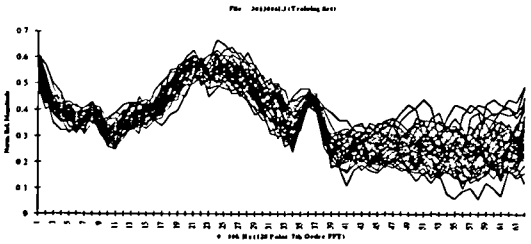


Figure. 4

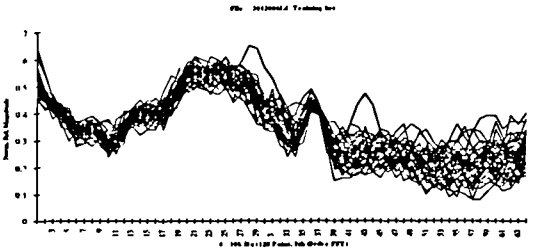


Figure. 5

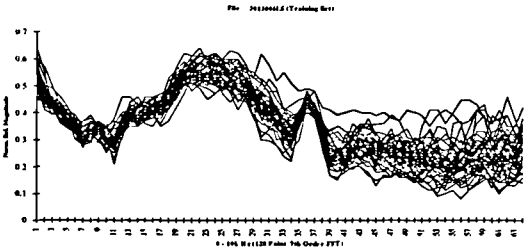


Figure. 6

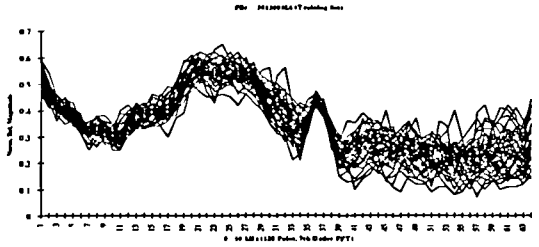


Figure. 7

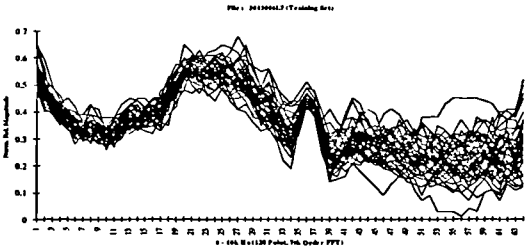


Figure. 8

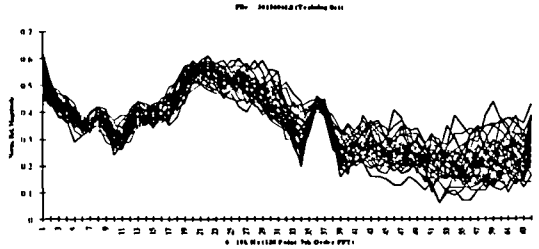


Figure. 9

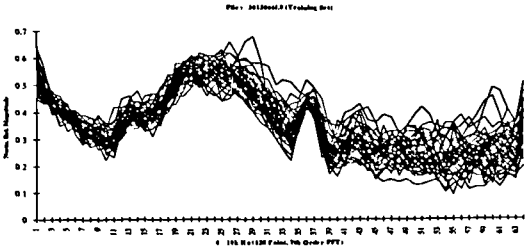


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

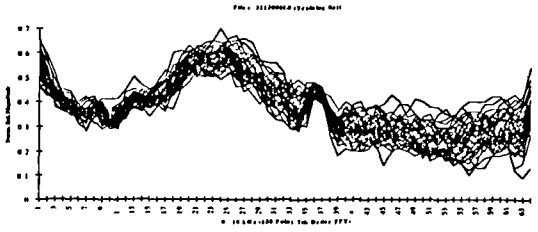


Figure. 1

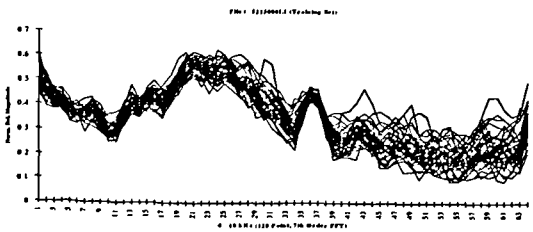


Figure. 2

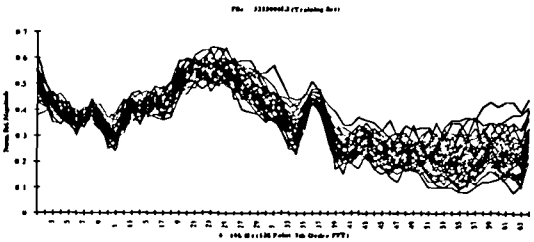


Figure. 3

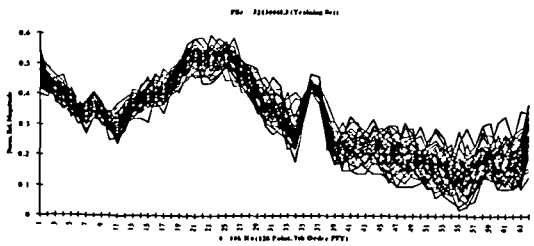


Figure. 4

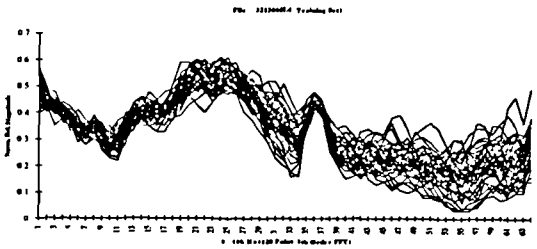


Figure. 5

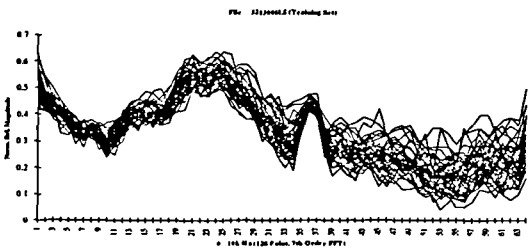


Figure. 6

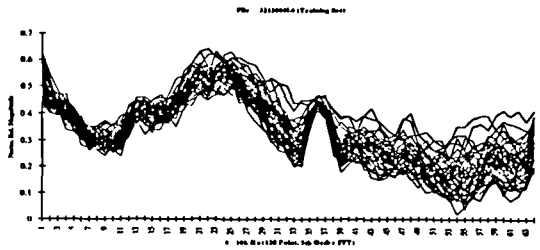


Figure. 7

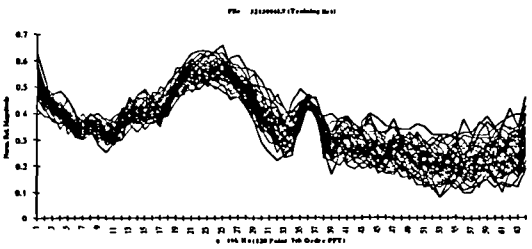


Figure. 8

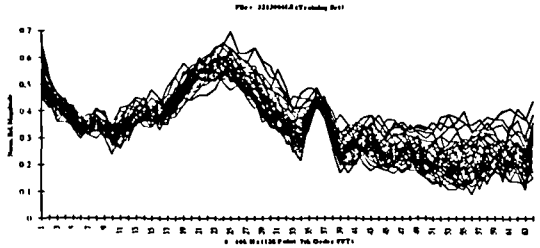


Figure. 9

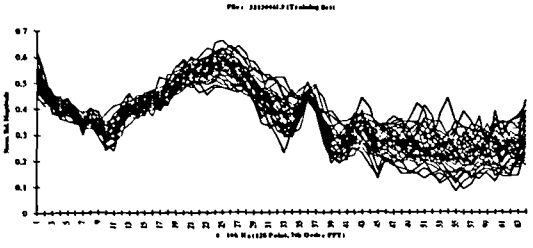


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

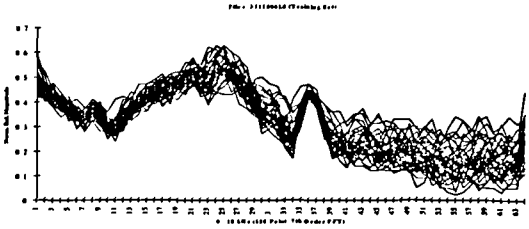


Figure. 1

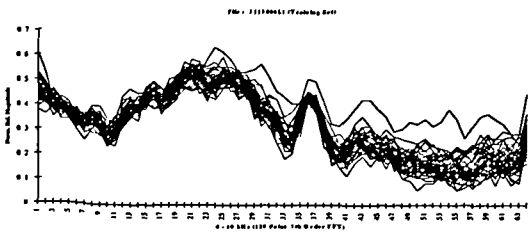


Figure. 2

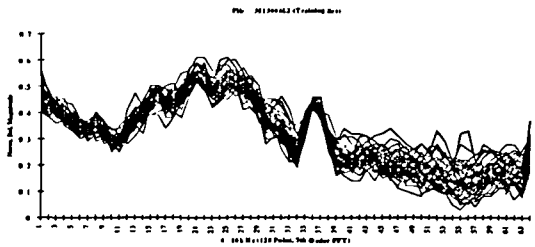


Figure. 3

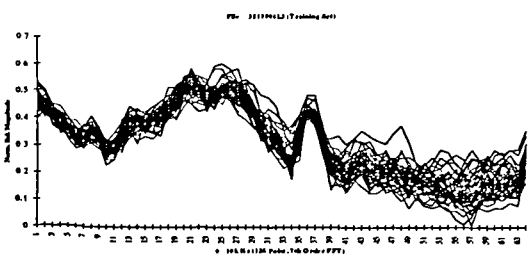


Figure. 4

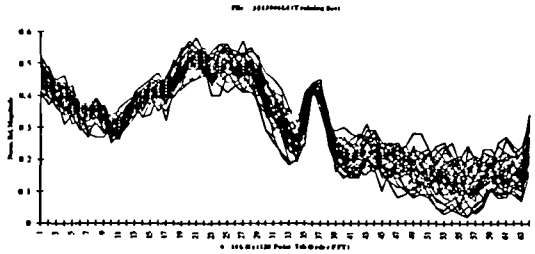


Figure. 5

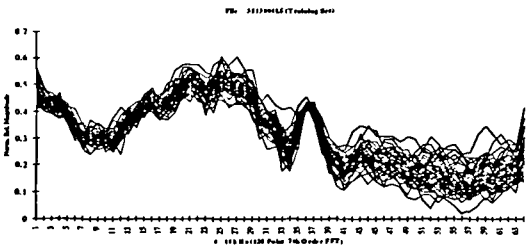


Figure. 6

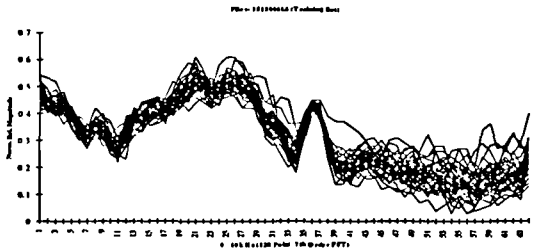


Figure. 7

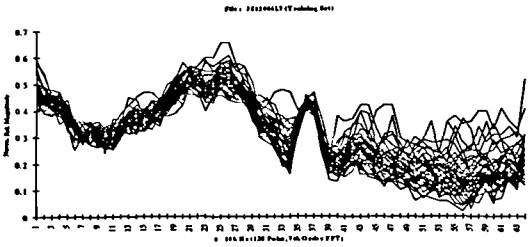


Figure. 8

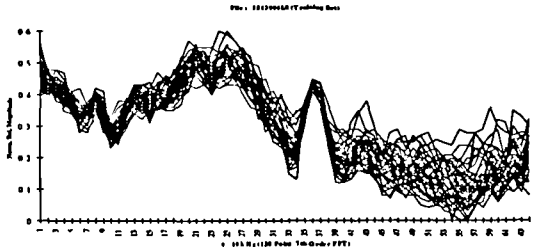


Figure. 9

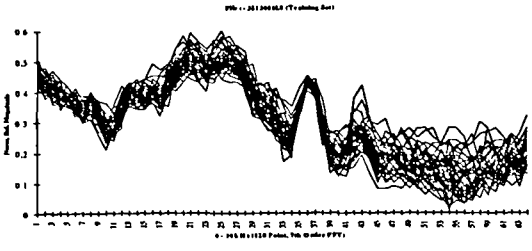


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

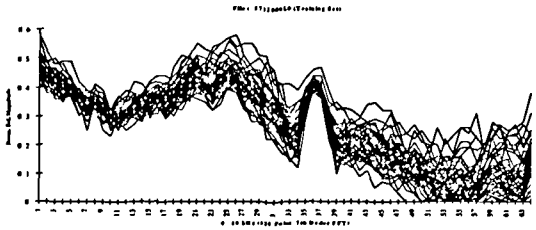


Figure. 1

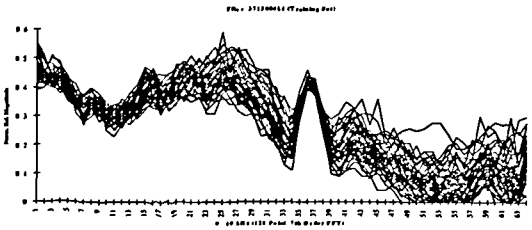


Figure. 2

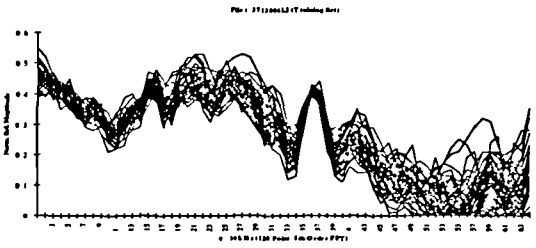


Figure. 3

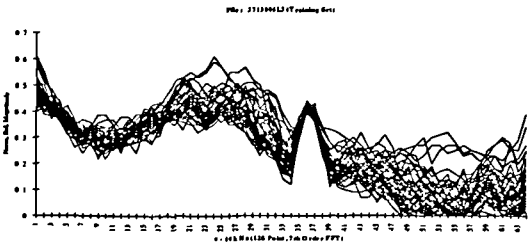


Figure. 4

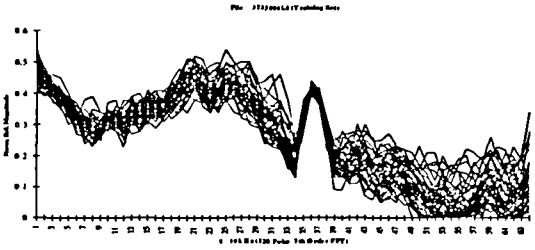


Figure. 5

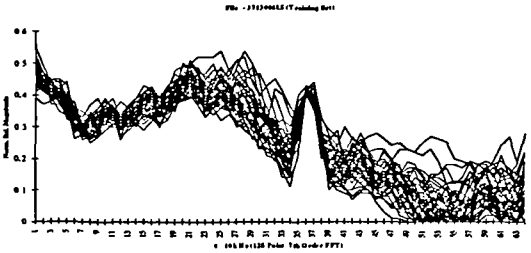


Figure. 6

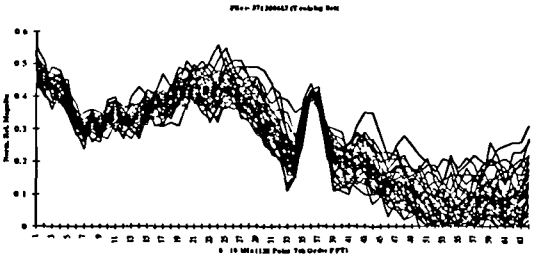


Figure. 7

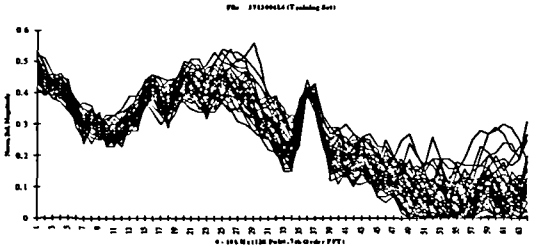


Figure. 8

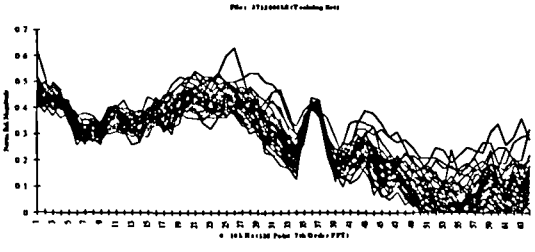


Figure. 9

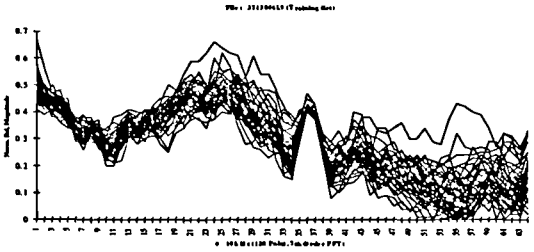


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

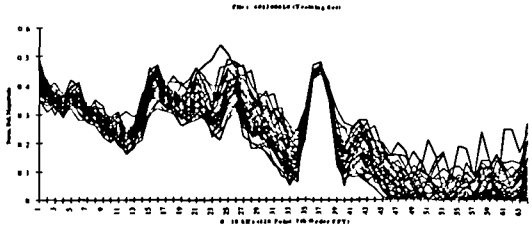


Figure. 1

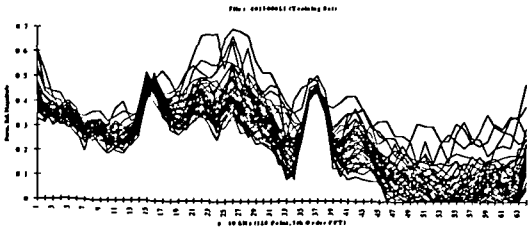


Figure. 2

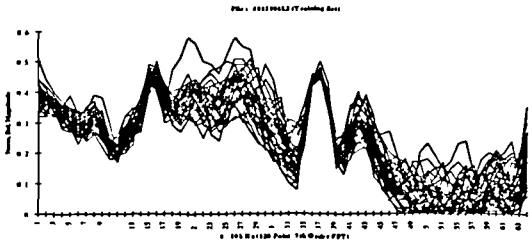


Figure. 3

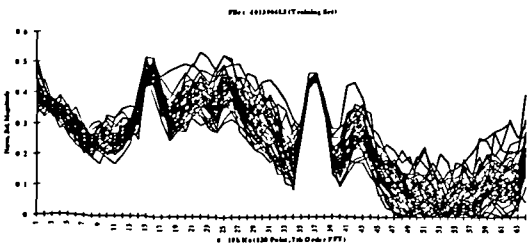


Figure. 4

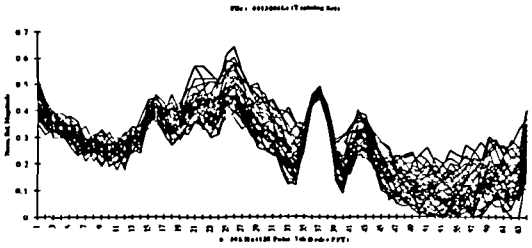


Figure. 5

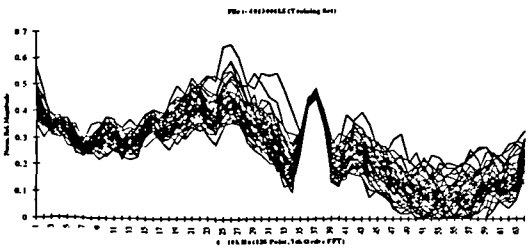


Figure. 6

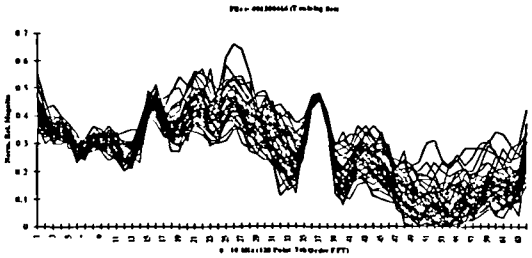


Figure. 7

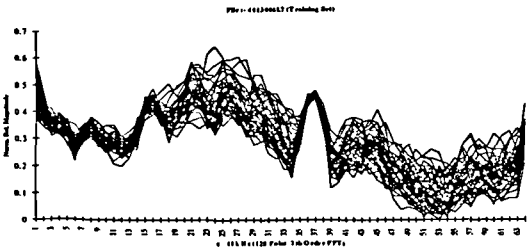


Figure. 8

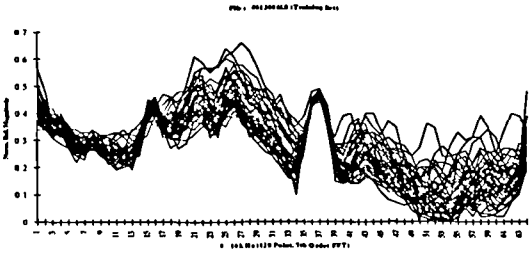


Figure. 9

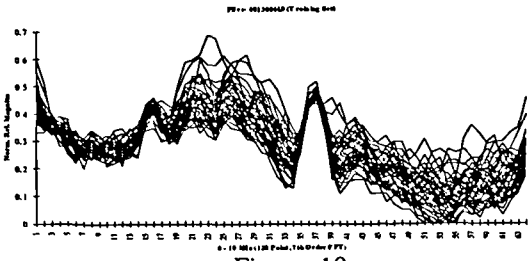


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

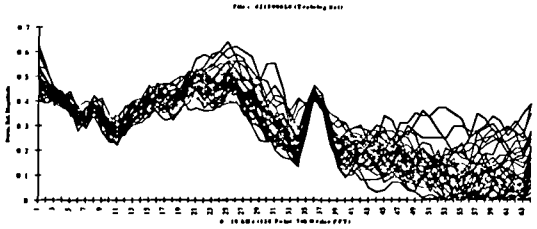


Figure. 1

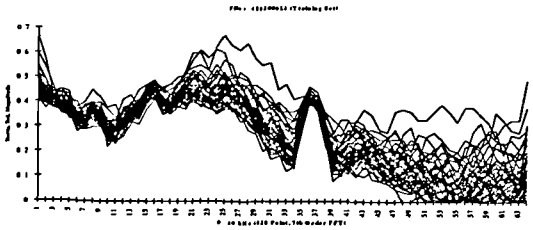


Figure. 2

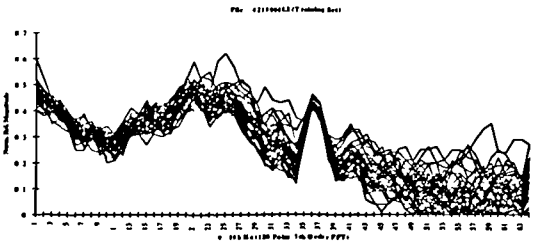


Figure. 3

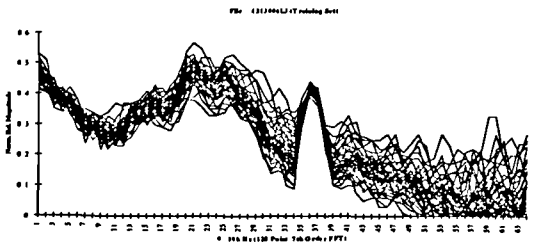


Figure. 4

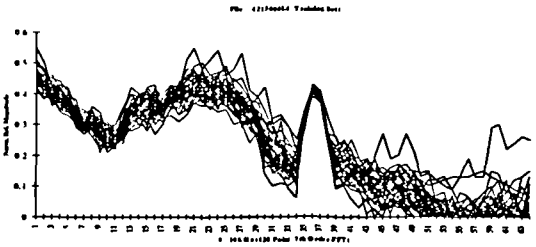


Figure. 5

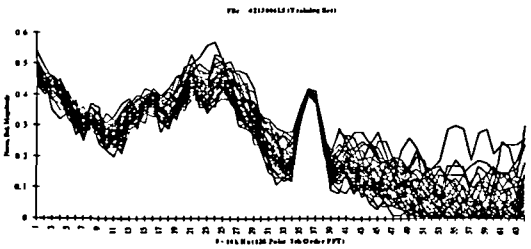


Figure. 6

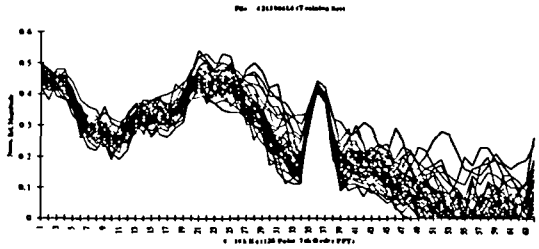


Figure. 7

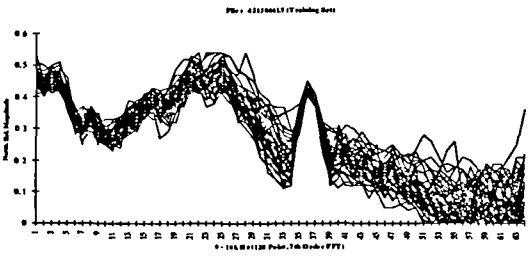


Figure. 8

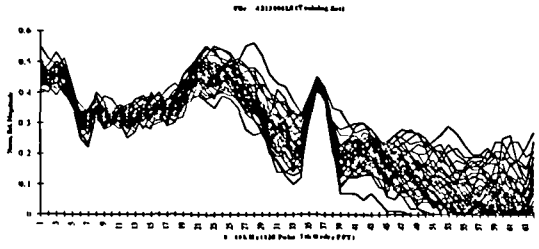


Figure. 9

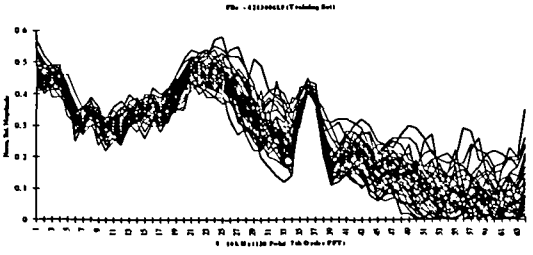


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

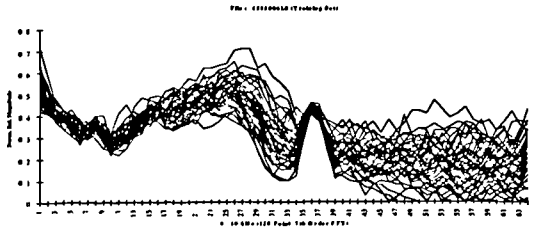


Figure. 1

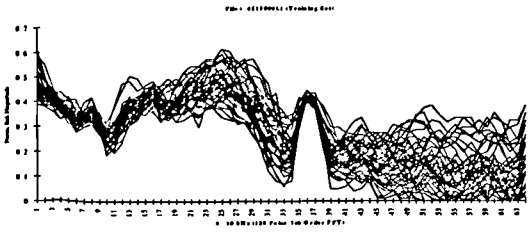


Figure. 2

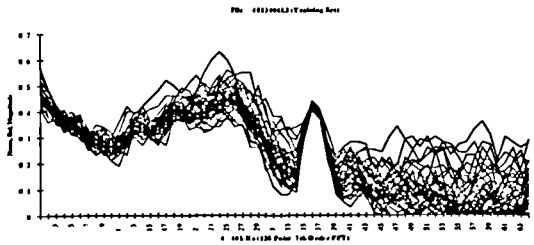


Figure. 3

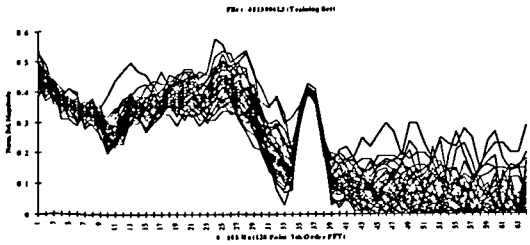


Figure. 4

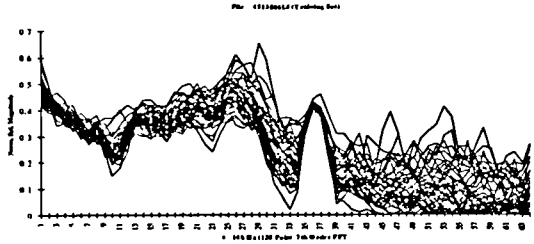


Figure. 5

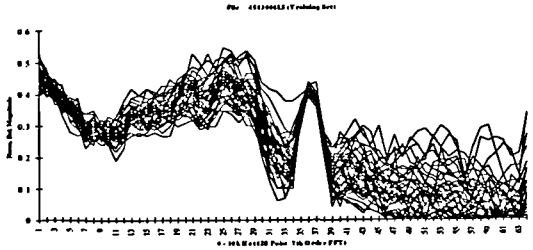


Figure. 6

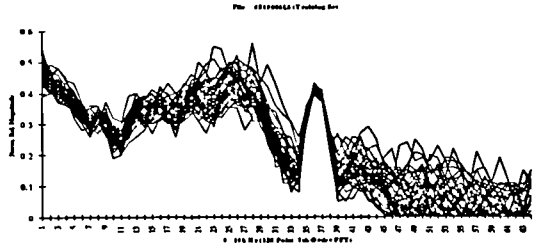


Figure. 7

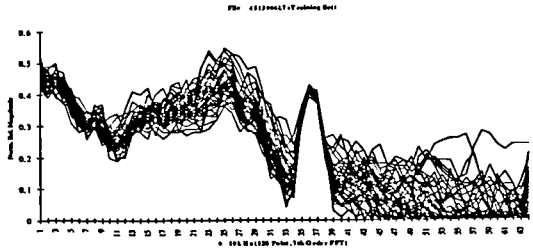


Figure. 8

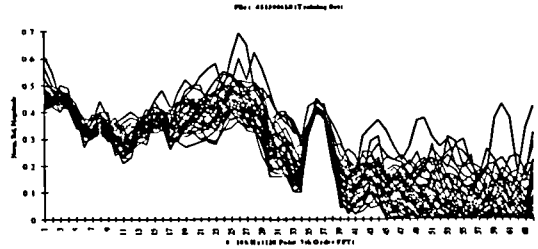


Figure. 9

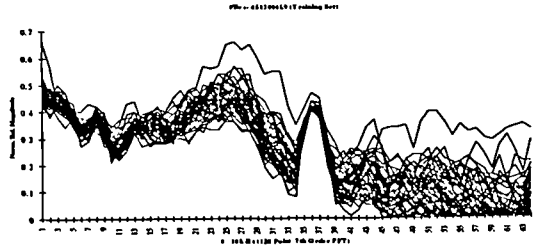


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

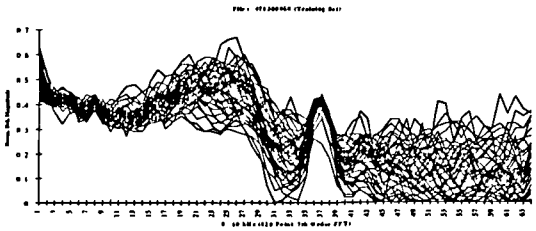


Figure. 1

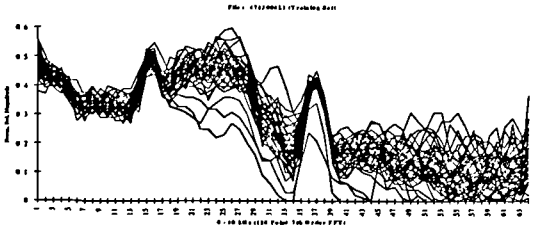


Figure. 2

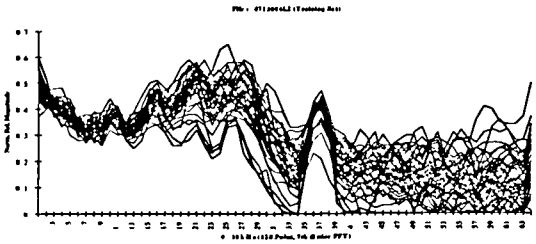


Figure. 3

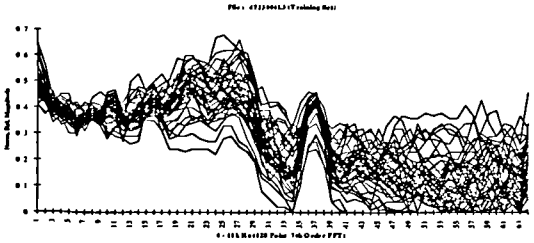


Figure. 4

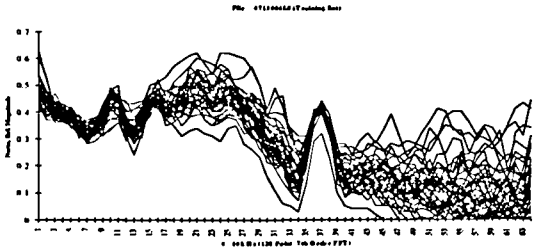


Figure. 5

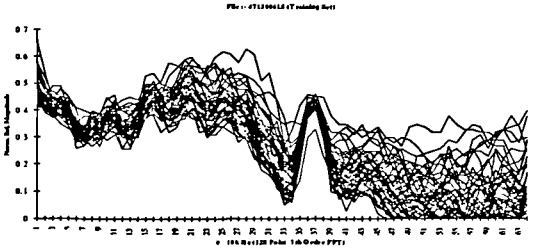


Figure. 6

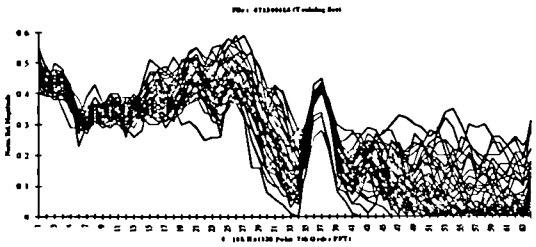


Figure. 7

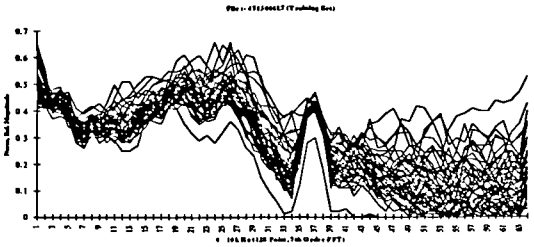


Figure. 8

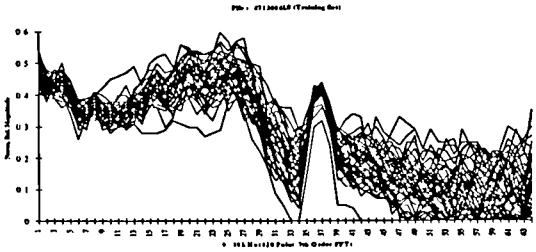


Figure. 9

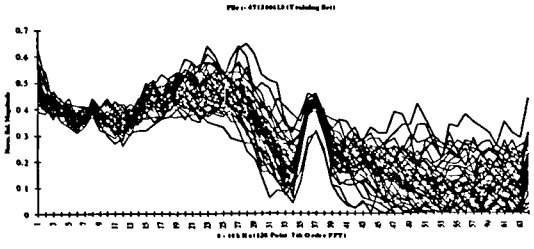


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

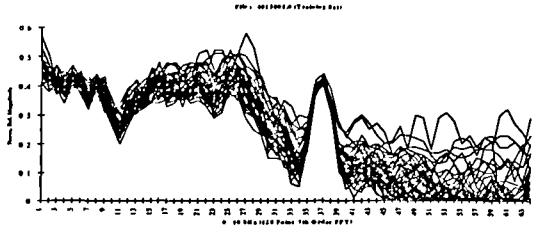


Figure. 1

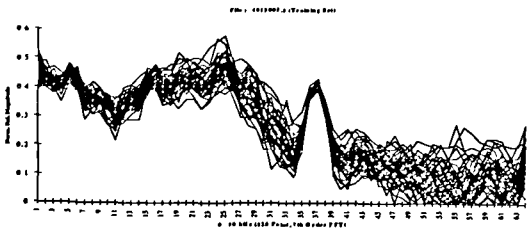


Figure. 2

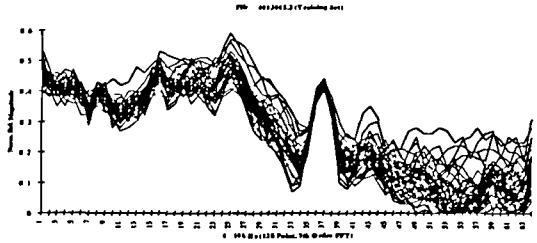


Figure. 3

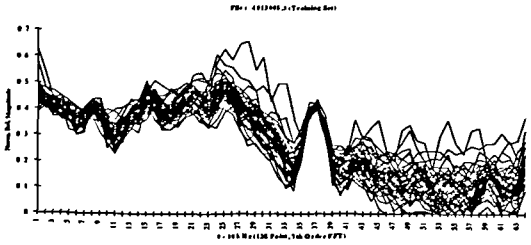


Figure. 4

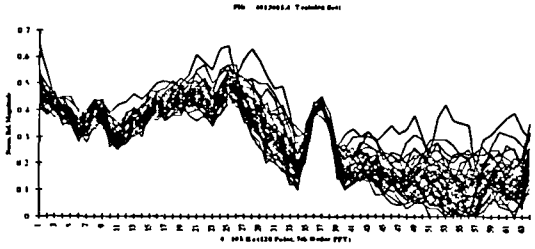


Figure. 5

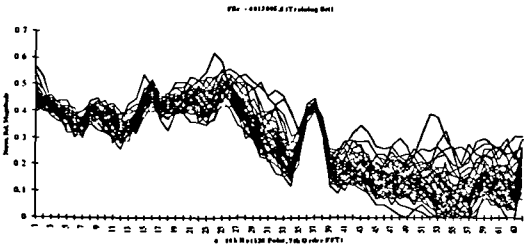


Figure. 6

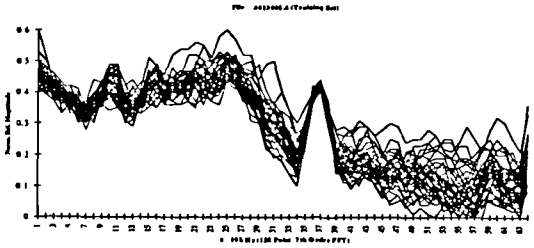


Figure. 7

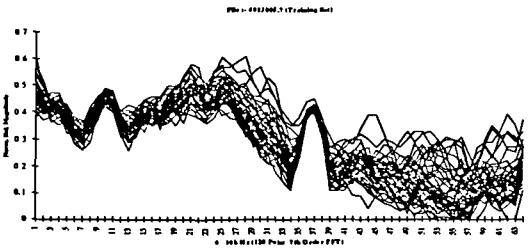


Figure. 8

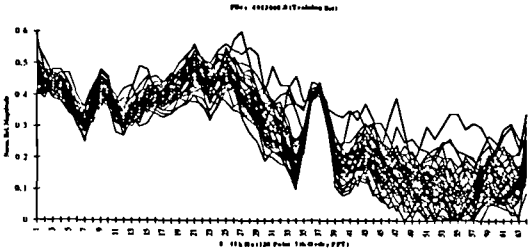


Figure. 9

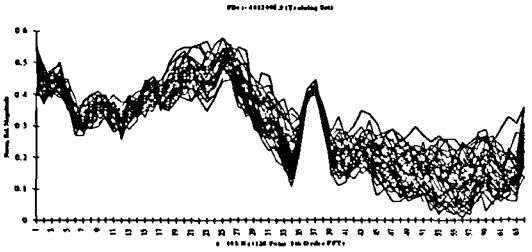


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

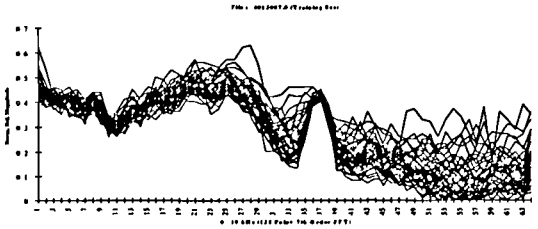


Figure. 1

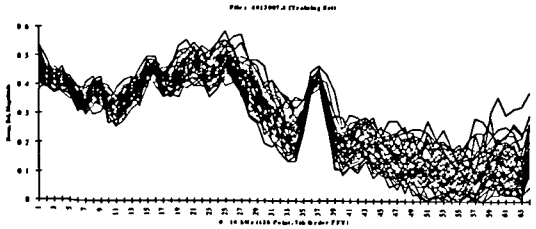


Figure. 2

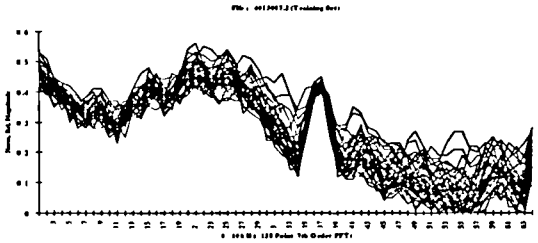


Figure. 3

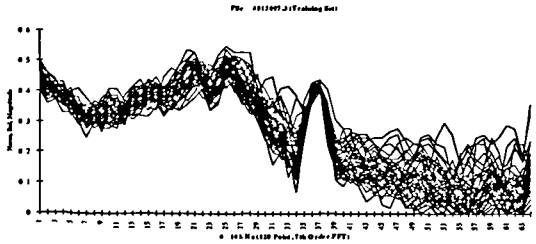


Figure. 4

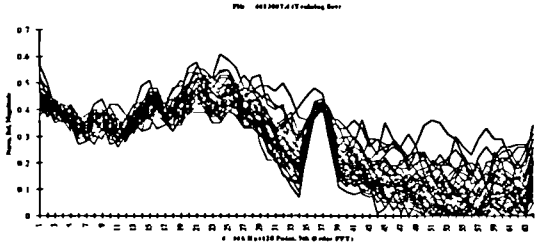


Figure. 5

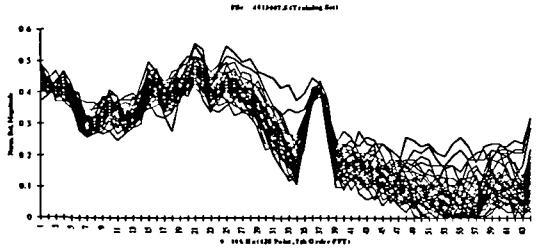


Figure. 6

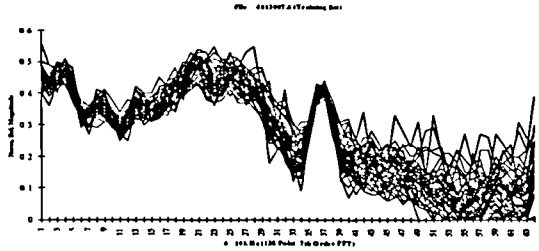


Figure. 7

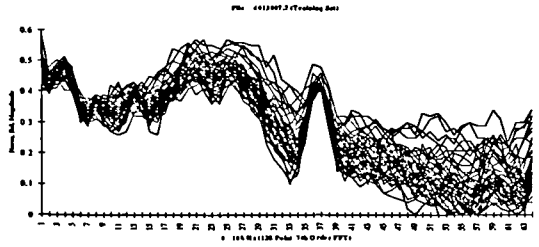


Figure. 8

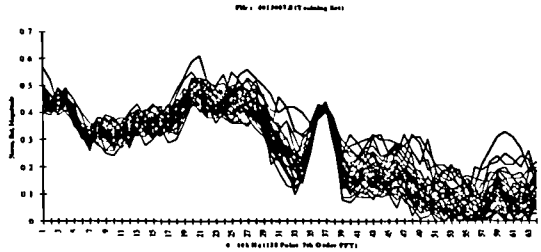


Figure. 9

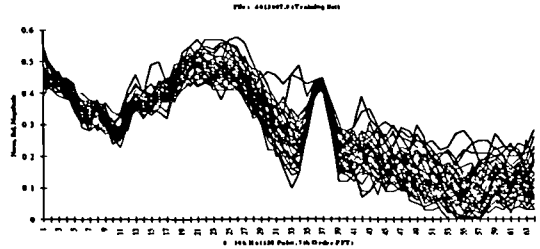


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

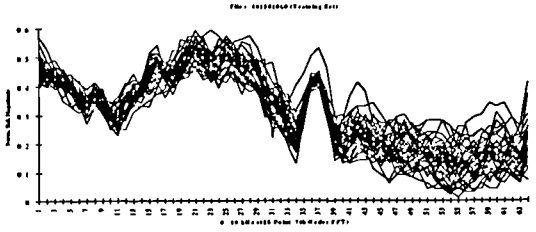


Figure. 1

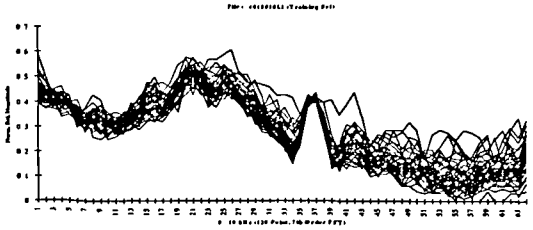


Figure. 2

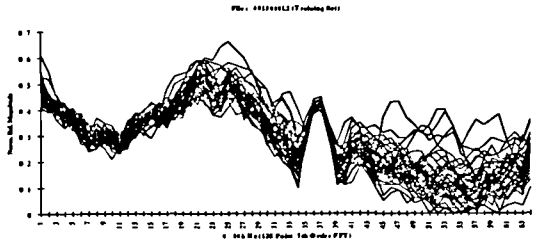


Figure. 3

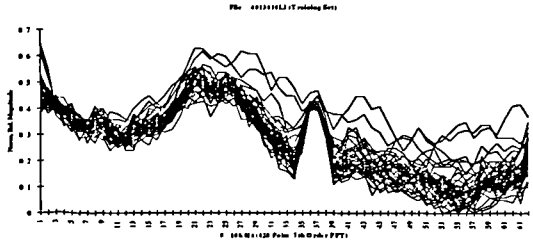


Figure. 4

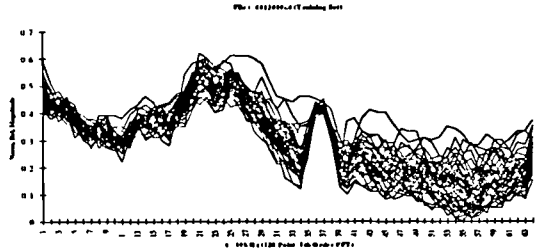


Figure. 5

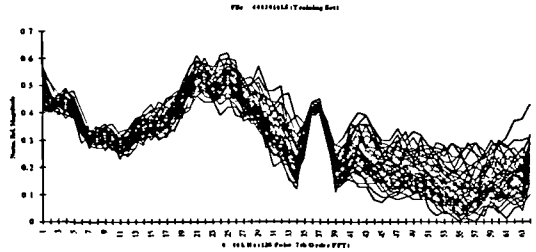


Figure. 6

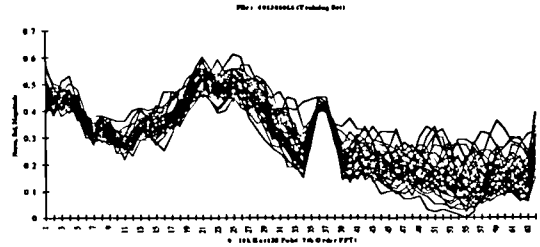


Figure. 7

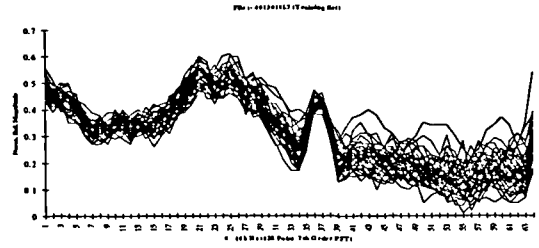


Figure. 8

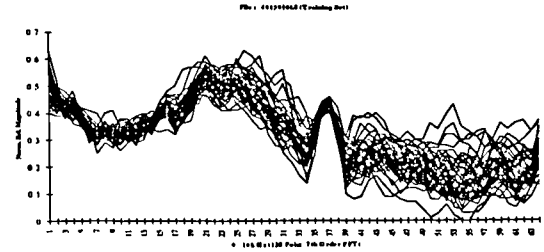


Figure. 9

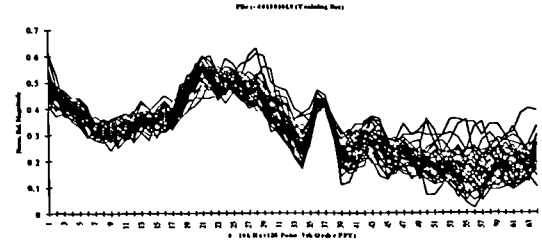


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

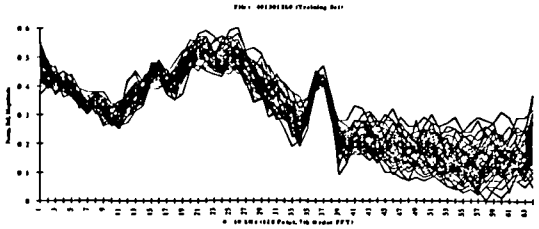


Figure. 1

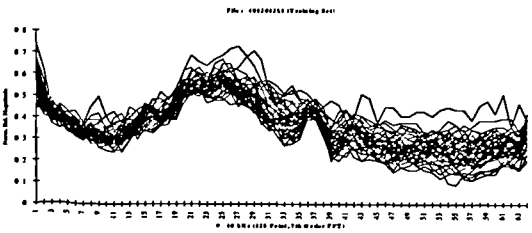


Figure. 2

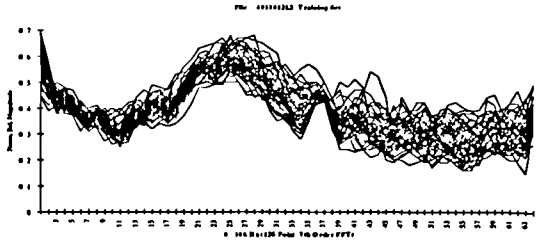


Figure. 3

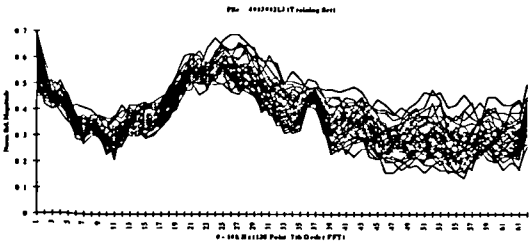


Figure. 4

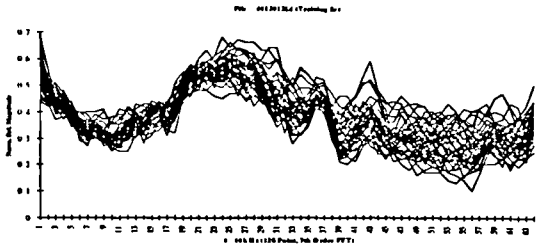


Figure. 5

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

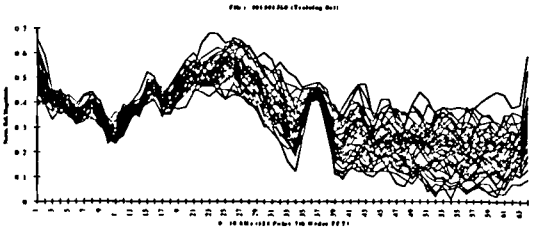


Figure. 1

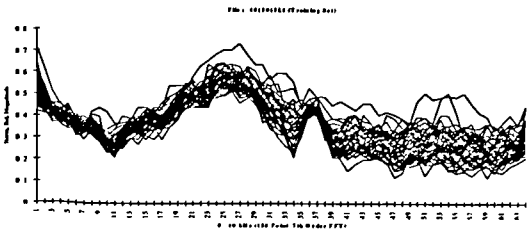


Figure. 2

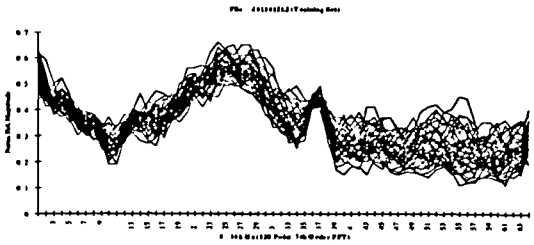


Figure. 3

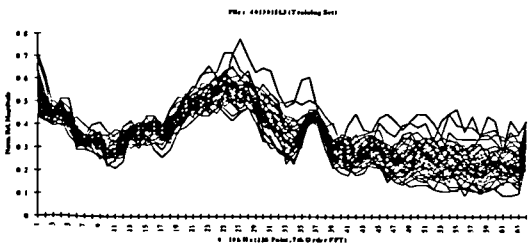


Figure. 4

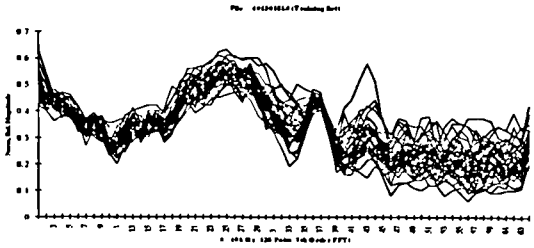


Figure. 5

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

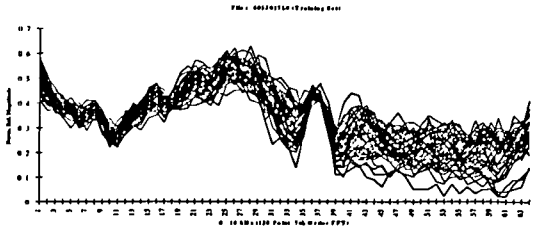


Figure. 1

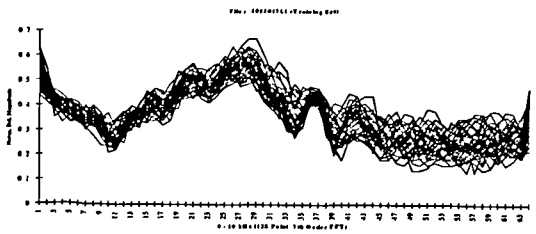


Figure. 2

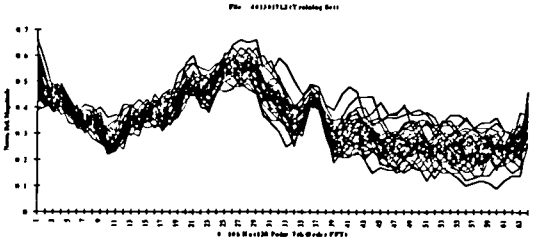


Figure. 3

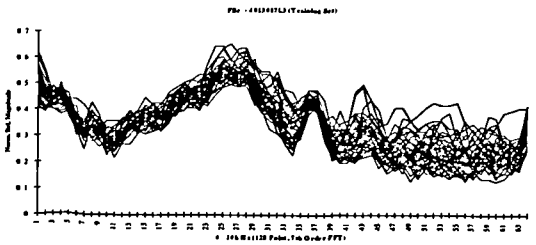


Figure. 4

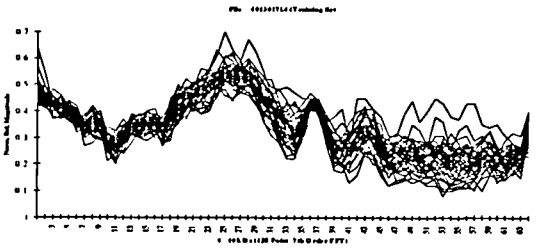


Figure. 5

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

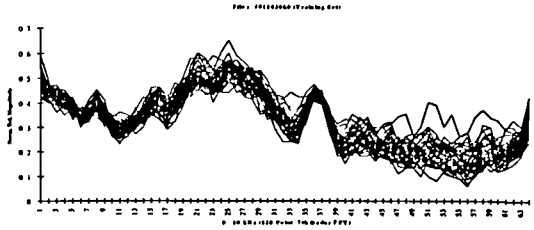


Figure. 1

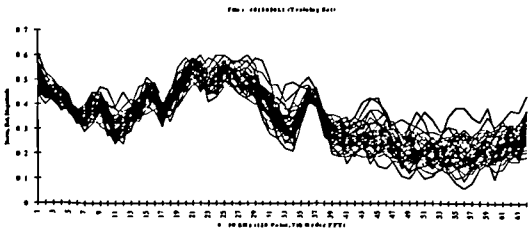


Figure. 2

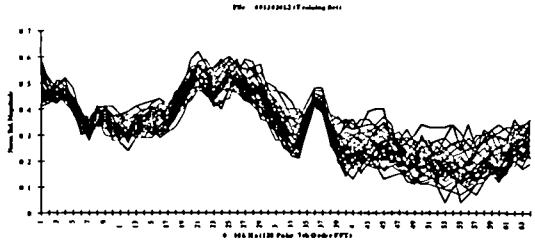


Figure. 3

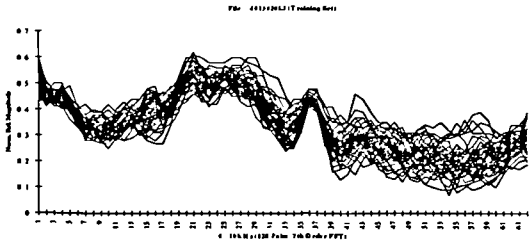


Figure. 4

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

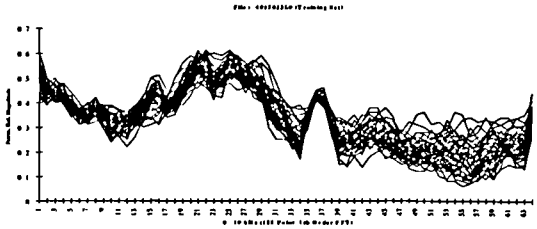


Figure. 1

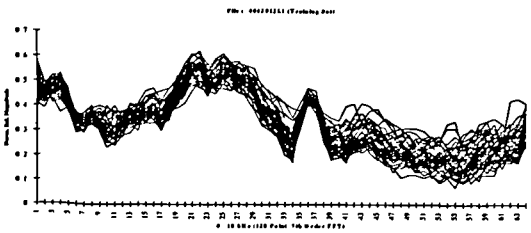


Figure. 2

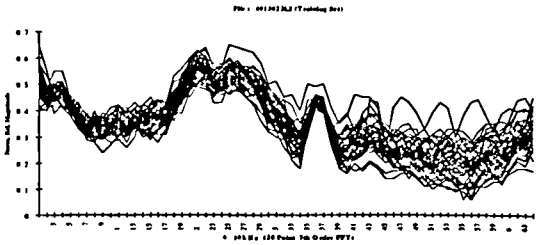


Figure. 3

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

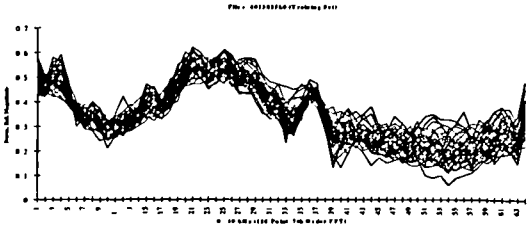


Figure. 1

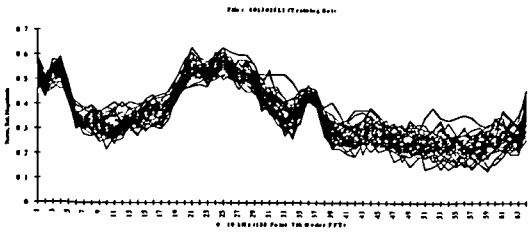


Figure. 2

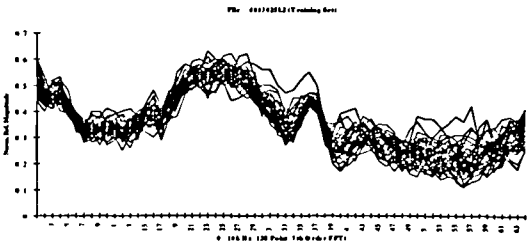


Figure. 3

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

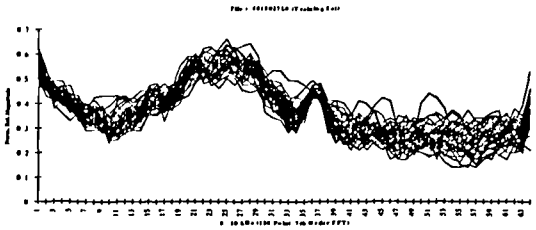


Figure. 1

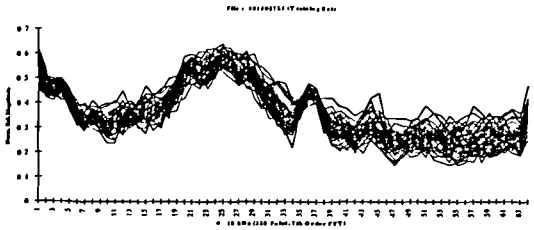


Figure. 2

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

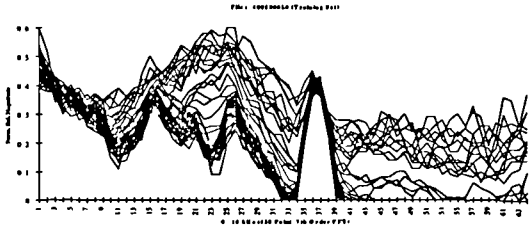


Figure. 1

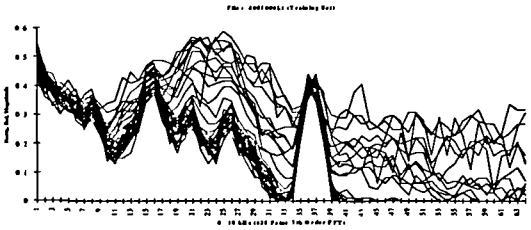


Figure. 2

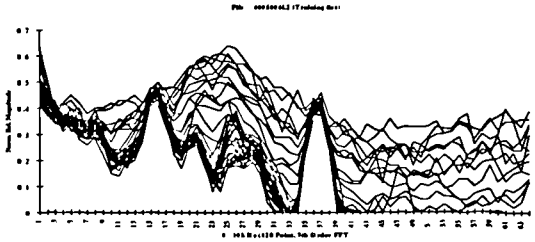


Figure. 3

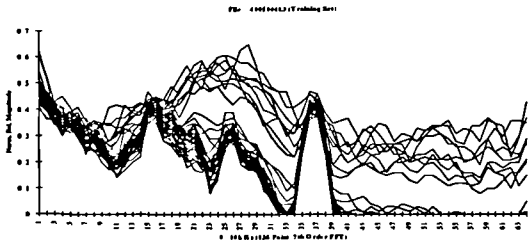


Figure. 4

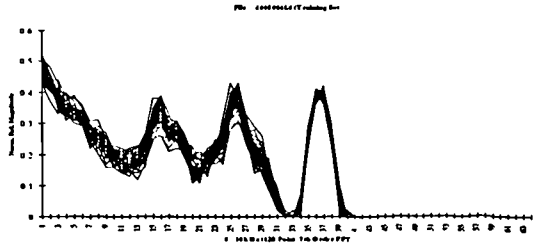


Figure. 5

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

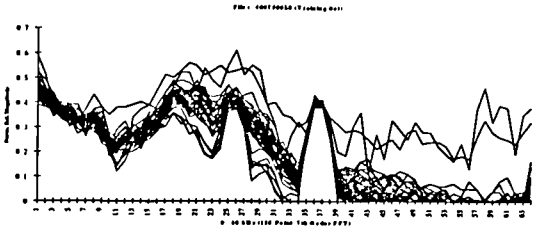


Figure. 1

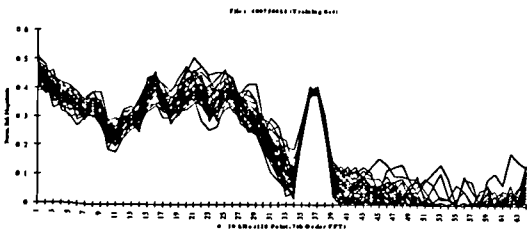


Figure. 2

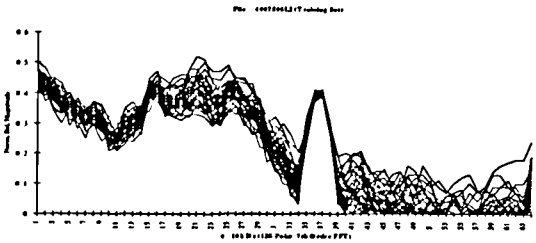


Figure. 3

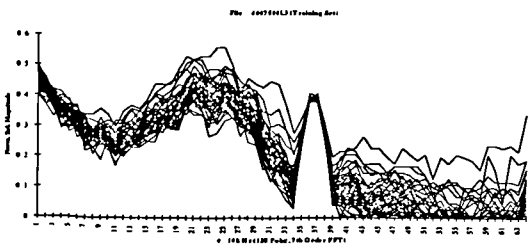


Figure. 4

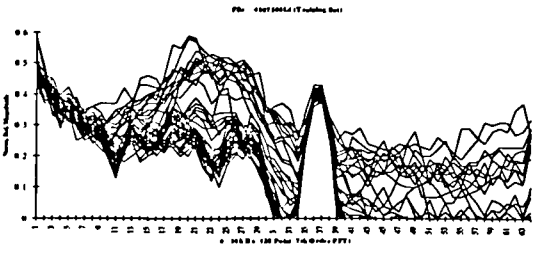


Figure. 5

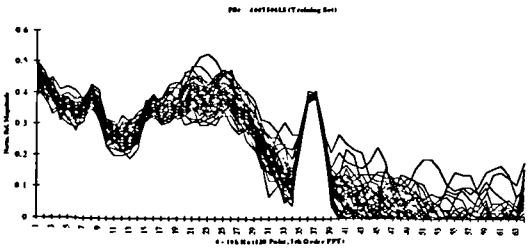


Figure. 6

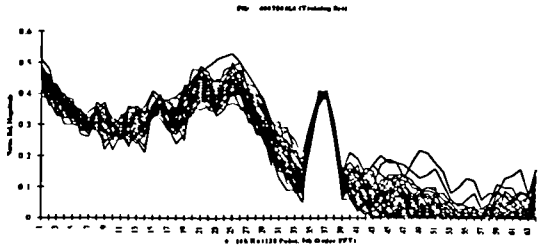


Figure. 7

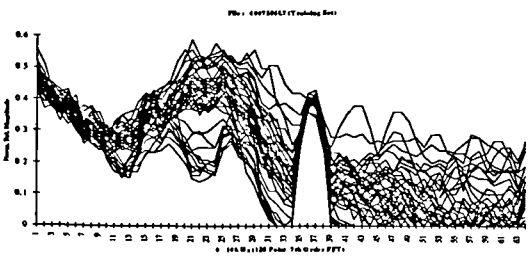


Figure. 8

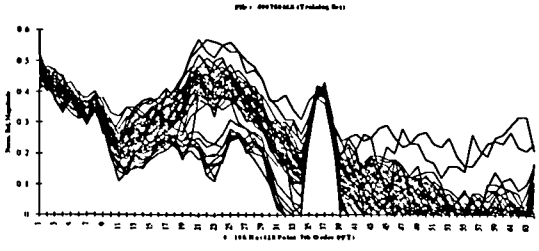


Figure. 9

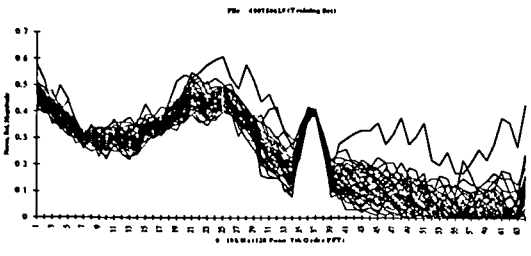


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

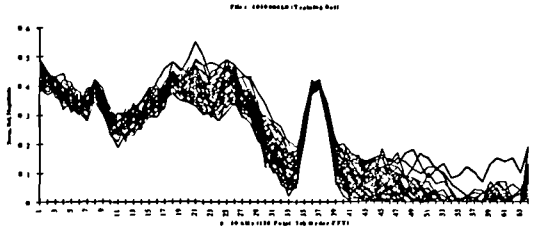


Figure. 1

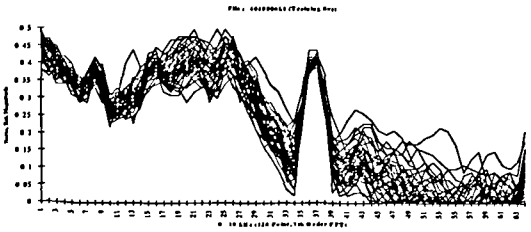


Figure. 2

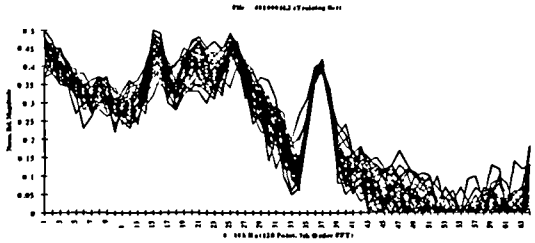


Figure. 3

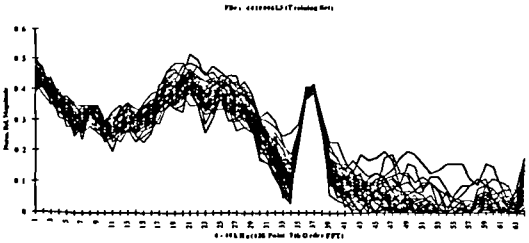


Figure. 4

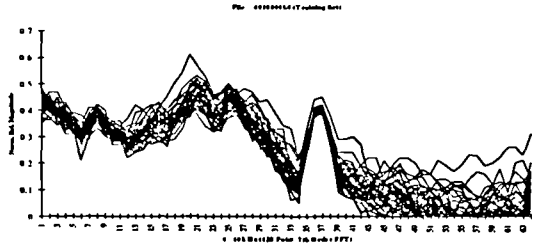


Figure. 5

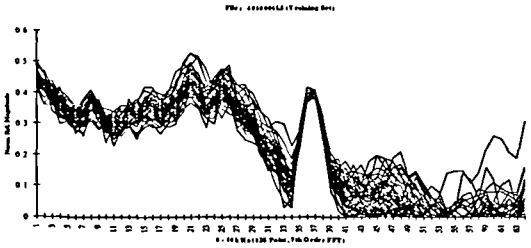


Figure. 6

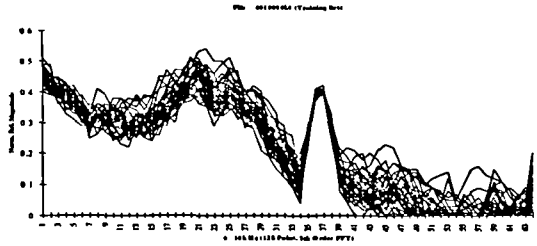


Figure. 7

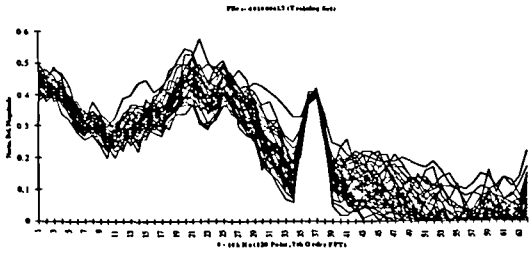


Figure. 8

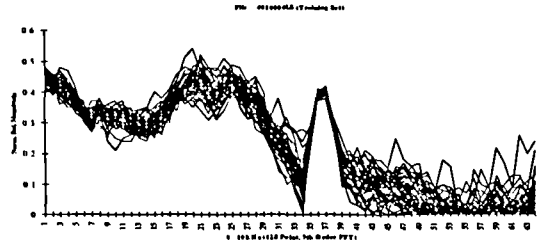


Figure. 9

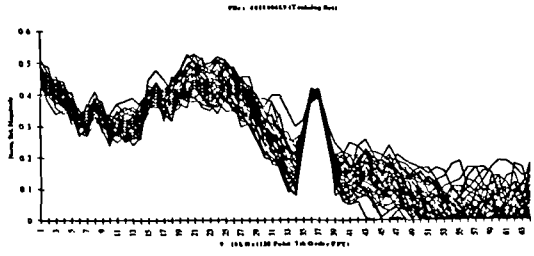


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

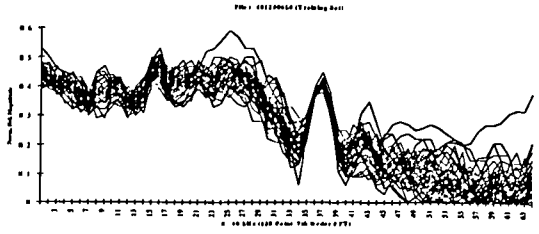


Figure. 1

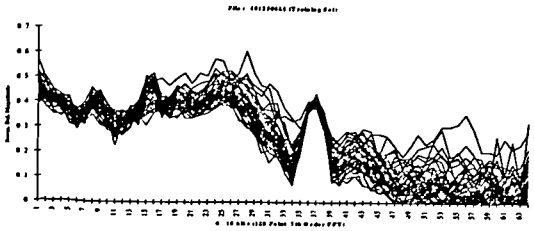


Figure. 2

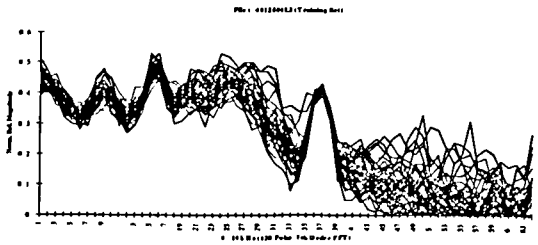


Figure. 3

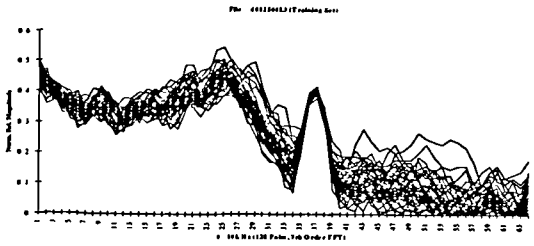


Figure. 4

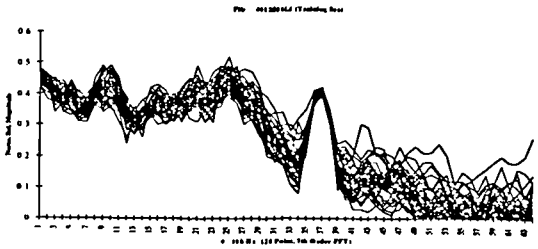


Figure. 5

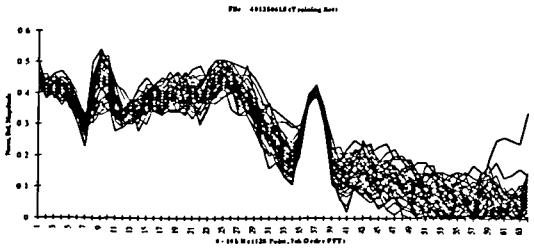


Figure. 6

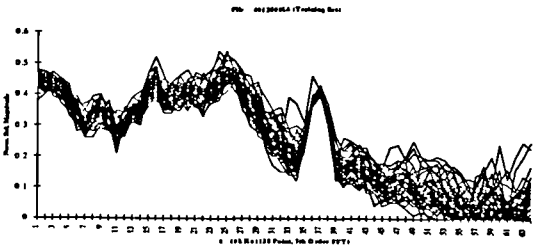


Figure. 7

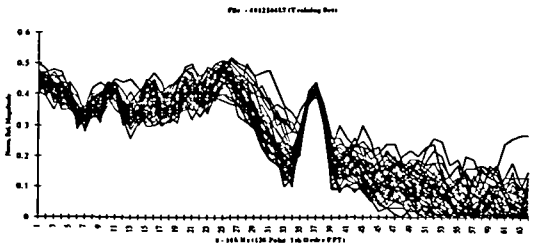


Figure. 8

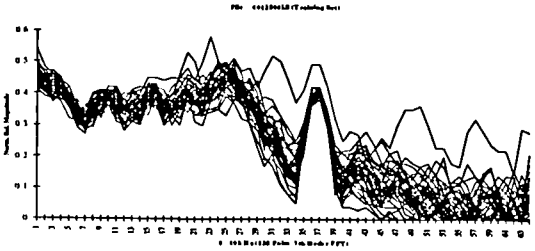


Figure. 9

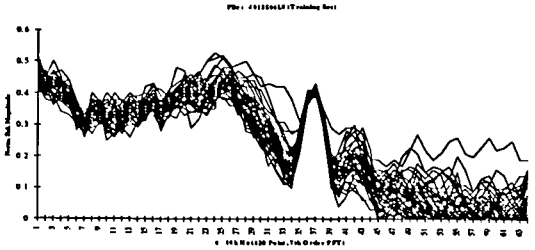


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

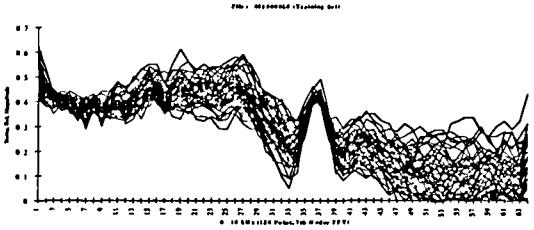


Figure. 1

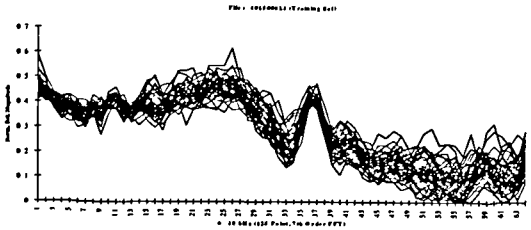


Figure. 2

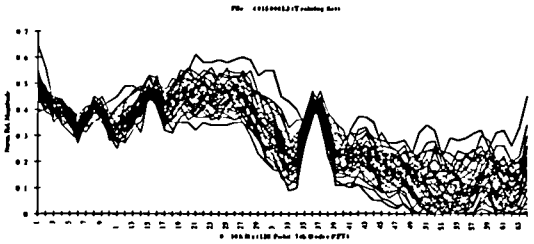


Figure. 3

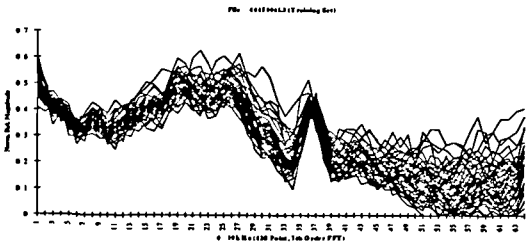


Figure. 4

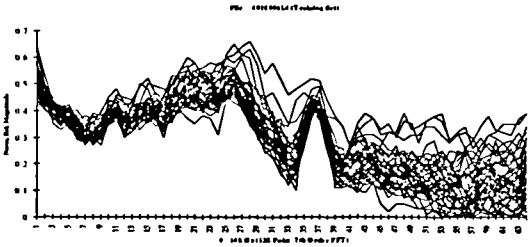


Figure. 5

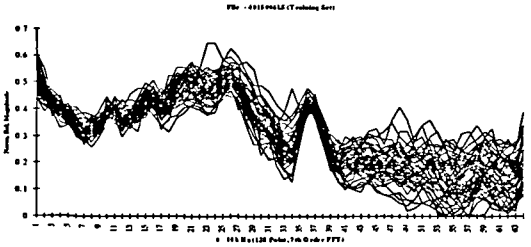


Figure. 6

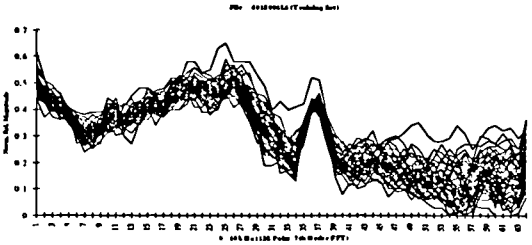


Figure. 7

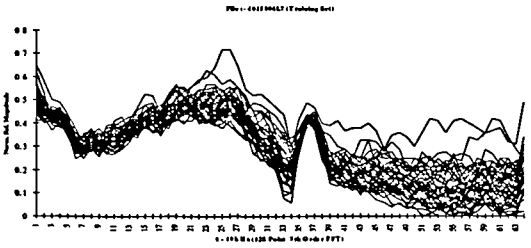


Figure. 8

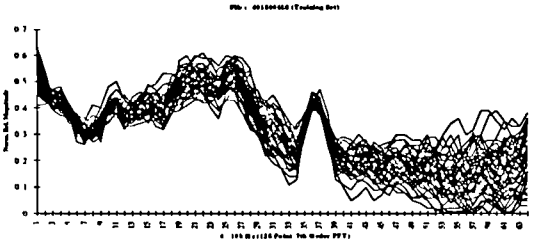


Figure. 9

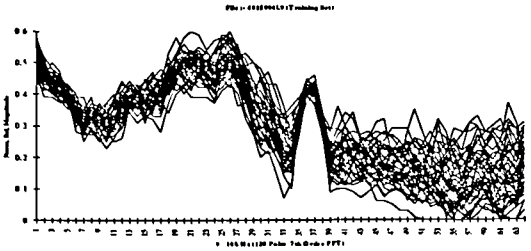


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

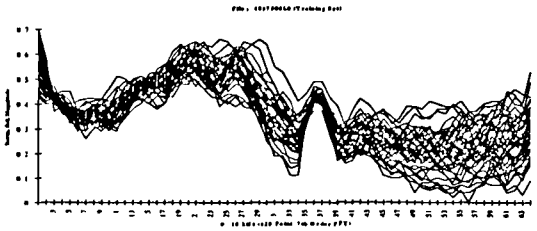


Figure. 1

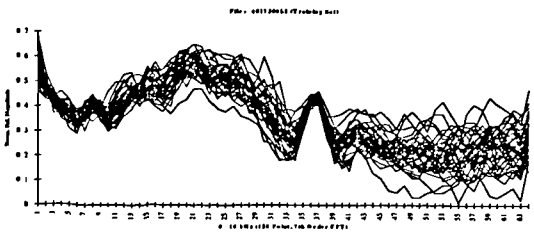


Figure. 2

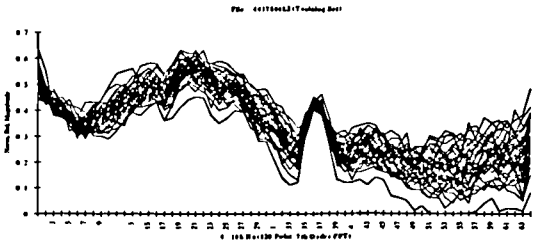


Figure. 3

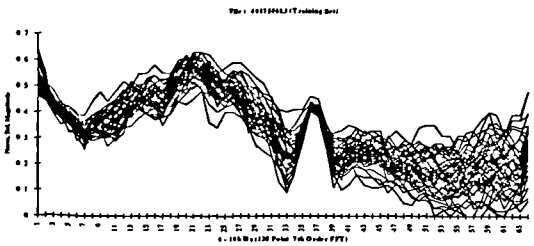


Figure. 4

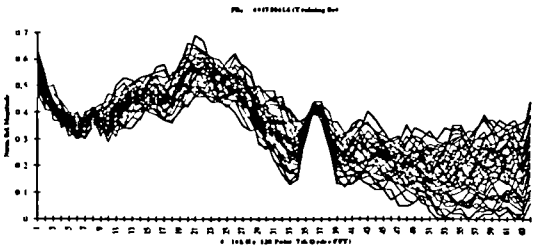


Figure. 5

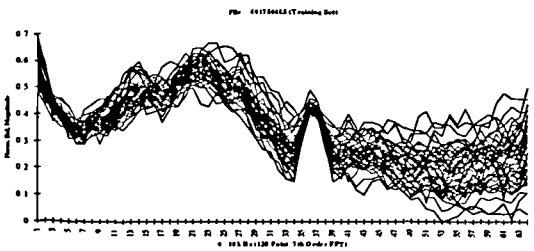


Figure. 6

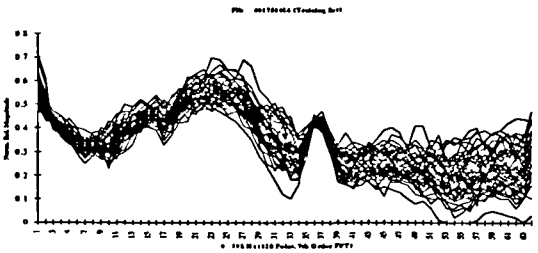


Figure. 7

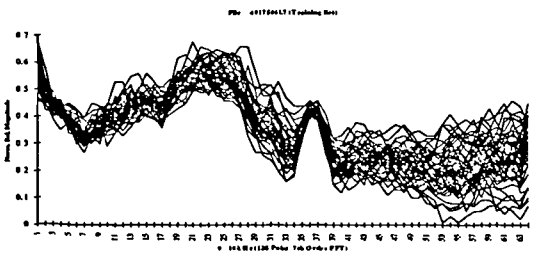


Figure. 8

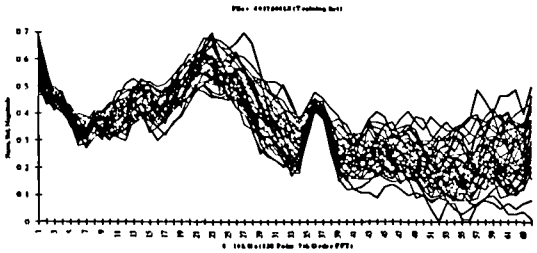


Figure. 9

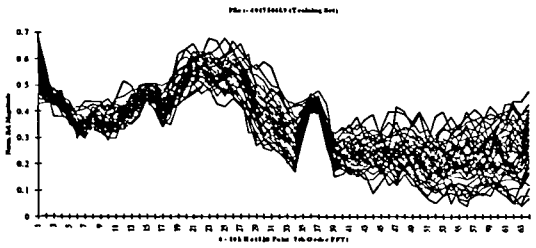


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

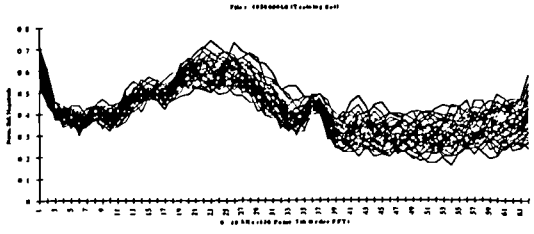


Figure. 1

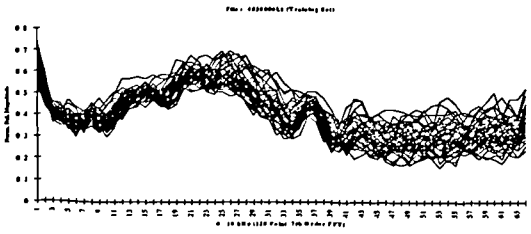


Figure. 2

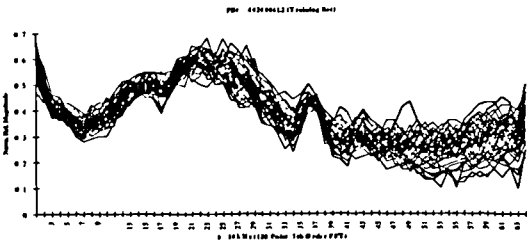


Figure. 3

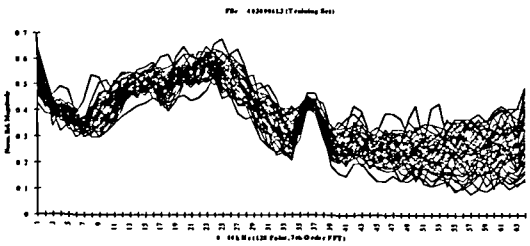


Figure. 4

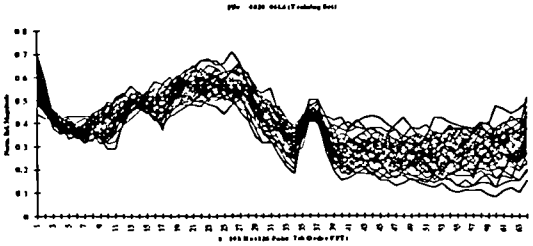


Figure. 5

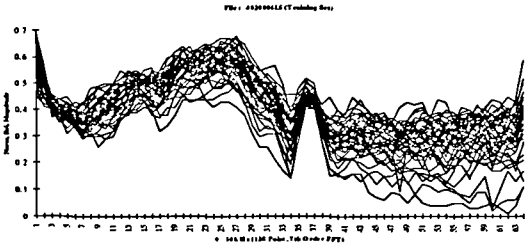


Figure. 6

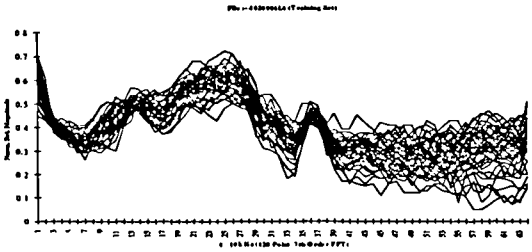


Figure. 7

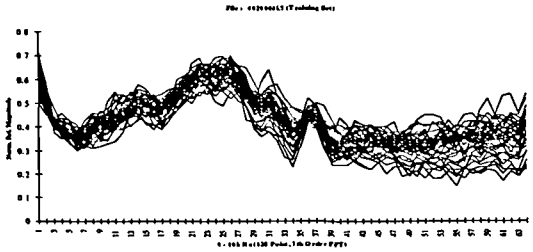


Figure. 8

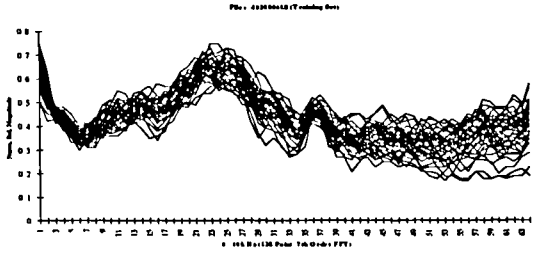


Figure. 9

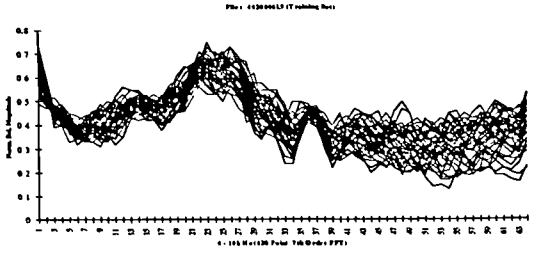


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

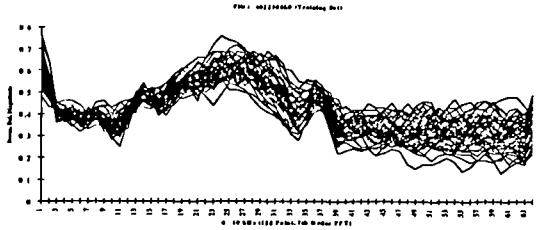


Figure. 1

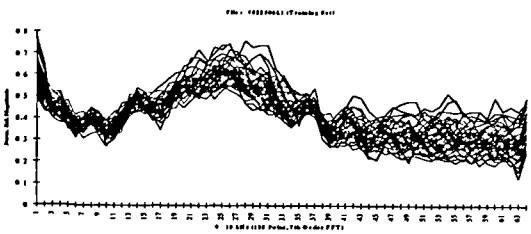


Figure. 2

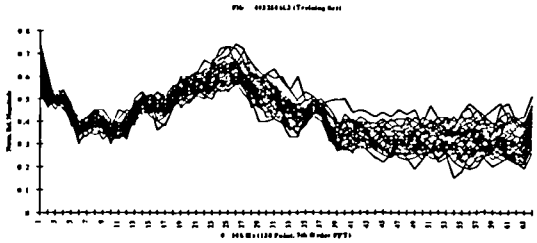


Figure. 3

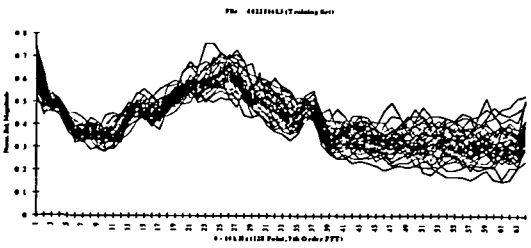


Figure. 4

Figure. 5

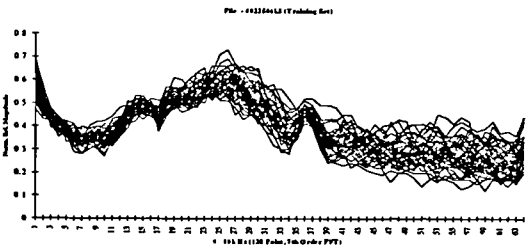


Figure. 6

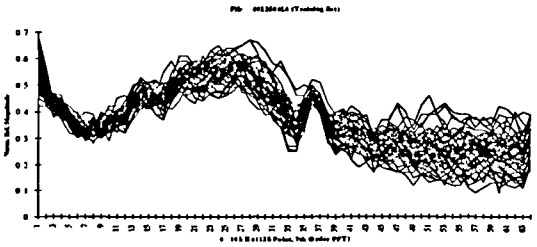


Figure. 7

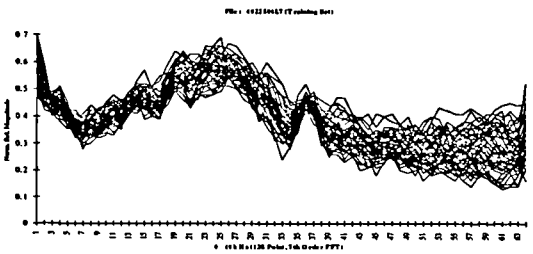


Figure. 8

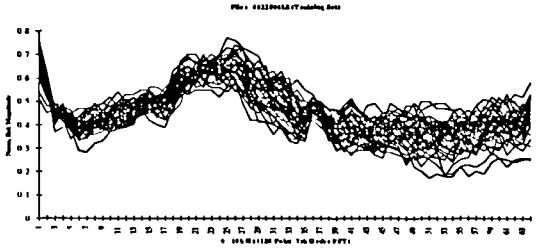


Figure. 9

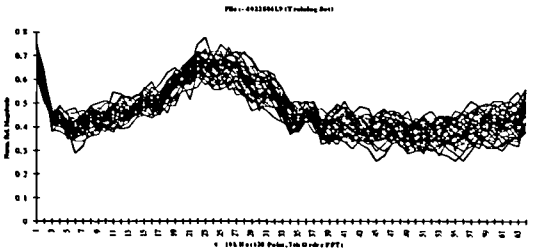


Figure. 10

7th Order Fast Fourier Transform Magnitude Plots

1st Level Input Patterns

Training Set (50 Frames / Plot)

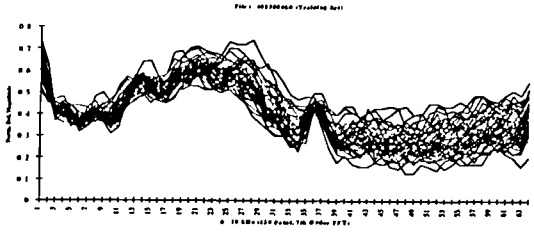


Figure. 1

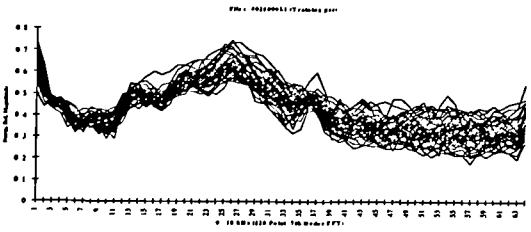


Figure. 2

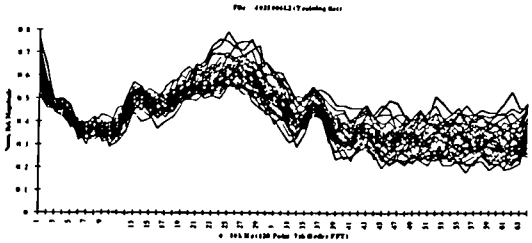


Figure. 3

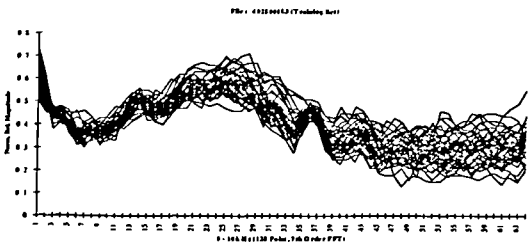


Figure. 4

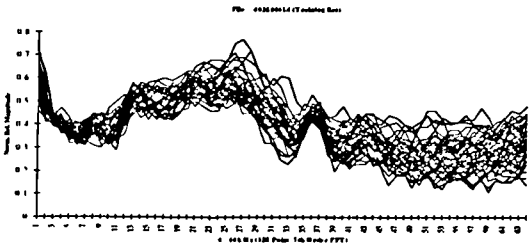


Figure. 5

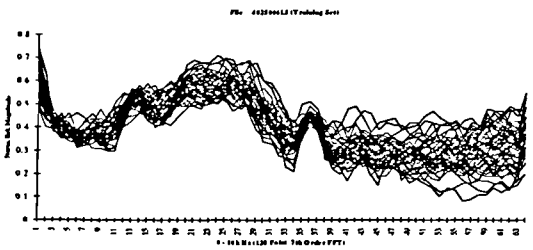


Figure. 6

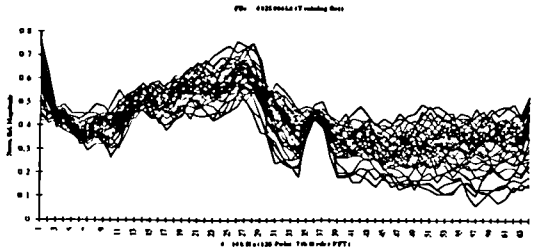


Figure. 7

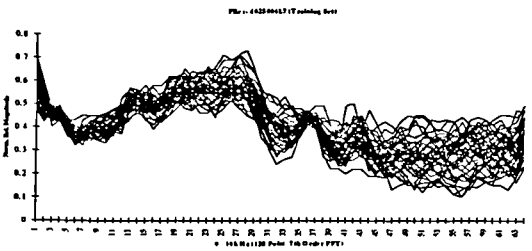


Figure. 8

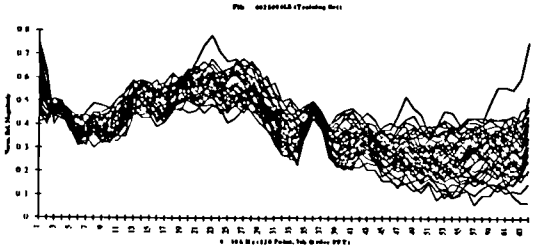


Figure. 9

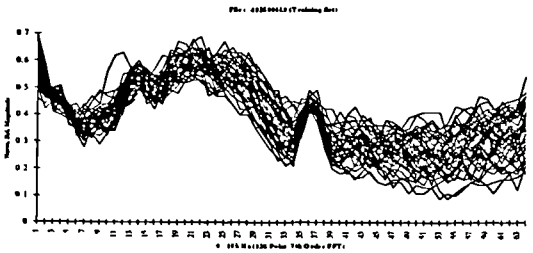


Figure. 10

APPENDIX 7

Typical 2nd Level Activation Map Training Data

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 25V, 130 Wfs, 6.5mm/s Ts

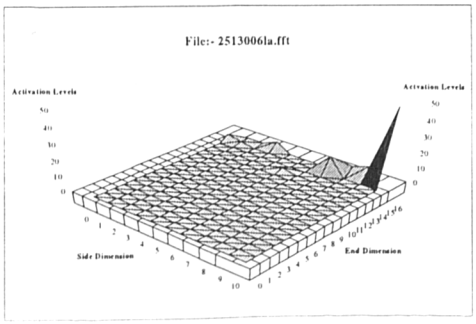


Figure. 1

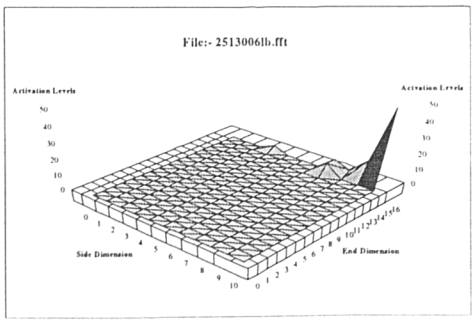


Figure. 2

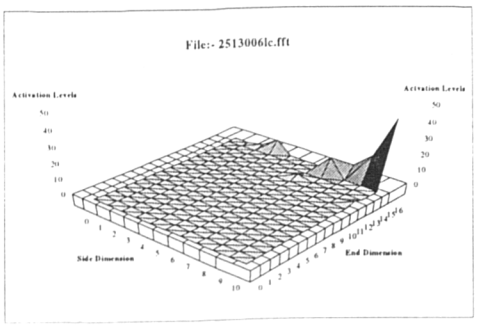


Figure. 3

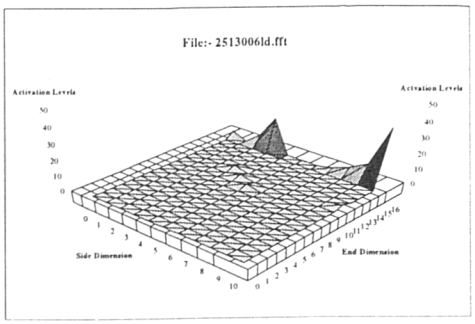


Figure. 4

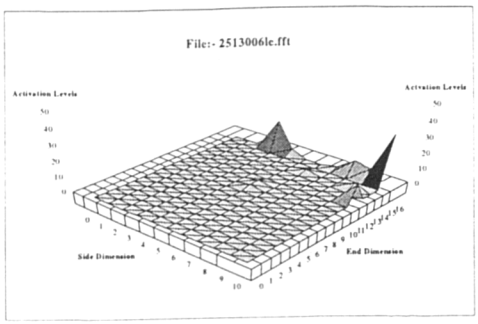


Figure. 5

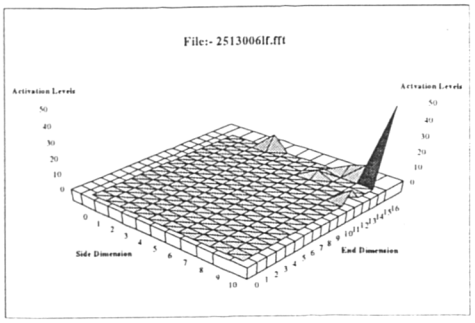


Figure. 6

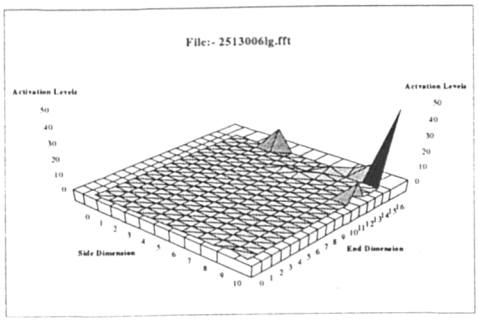


Figure. 7

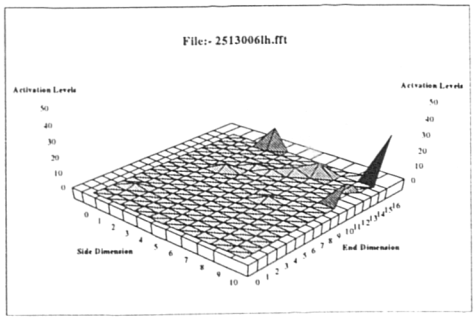


Figure. 8

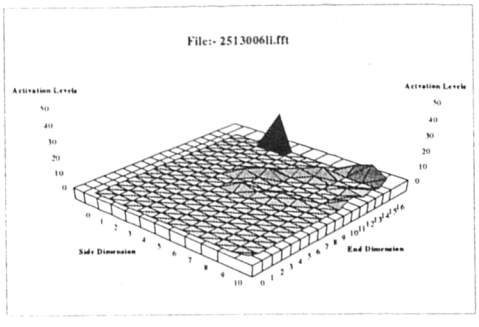


Figure. 9

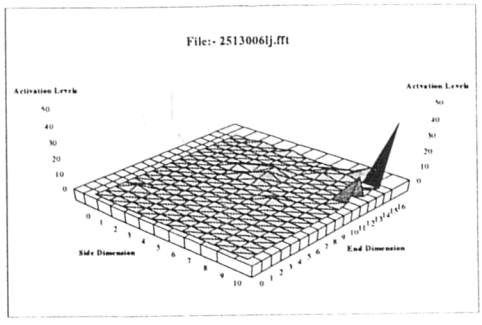


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 27V, 130 Wfs, 6.5mm/s Ts

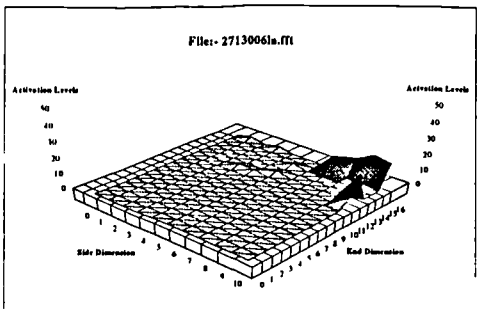


Figure. 1

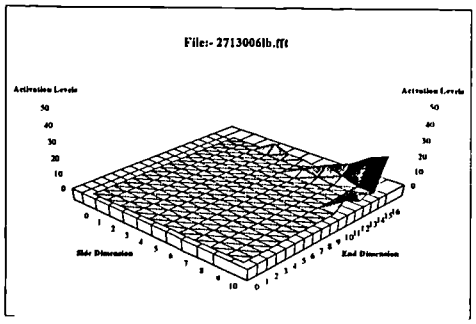


Figure. 2

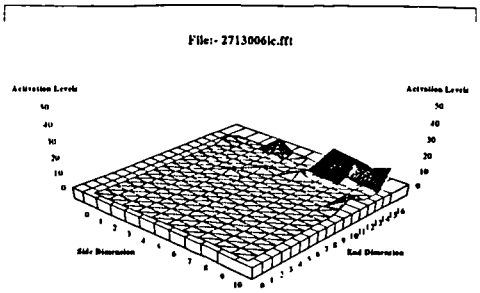


Figure. 3

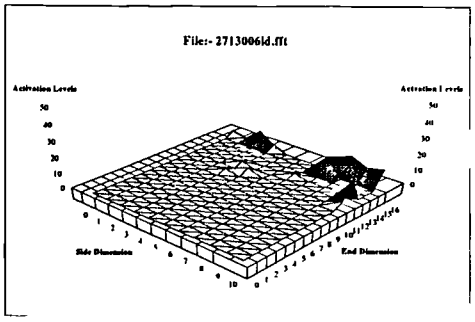


Figure. 4

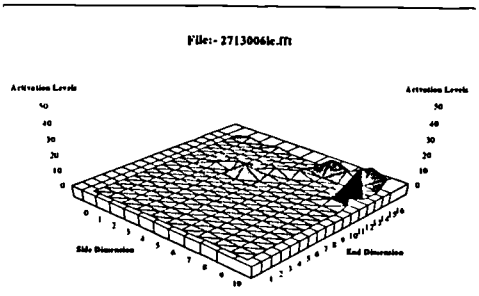


Figure. 5

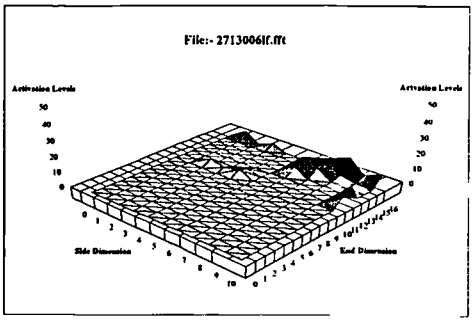


Figure. 6

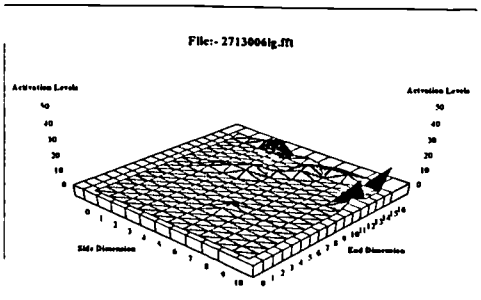


Figure. 7

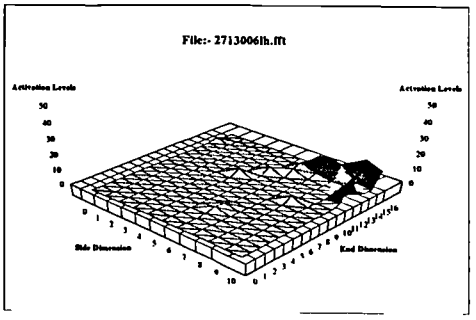


Figure. 8

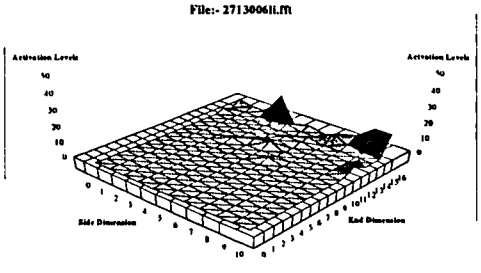


Figure. 9

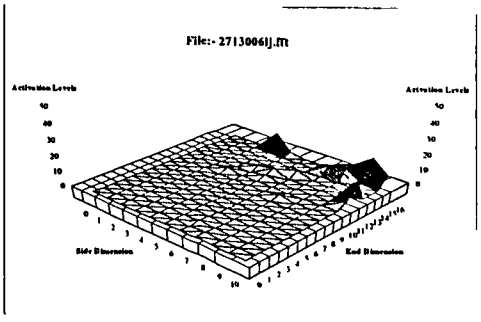


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 30V, 130 Wfs, 6.5mm/s Ts

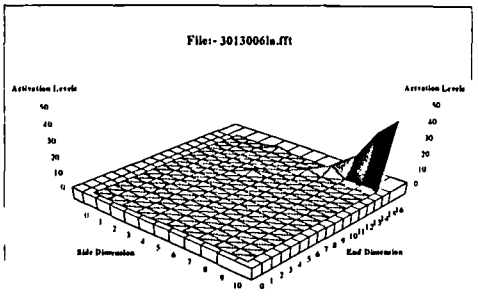


Figure. 1

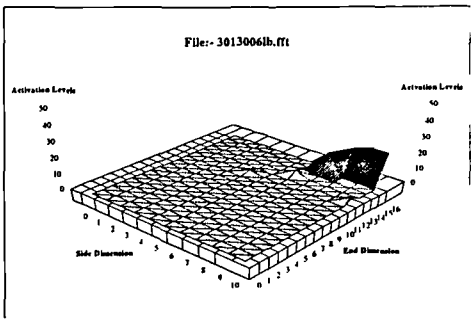


Figure. 2

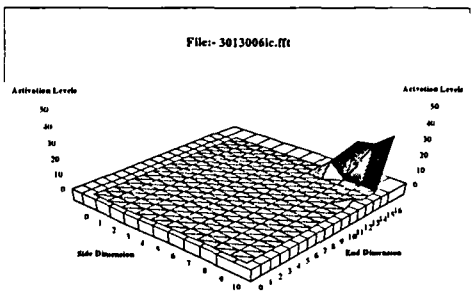


Figure. 3

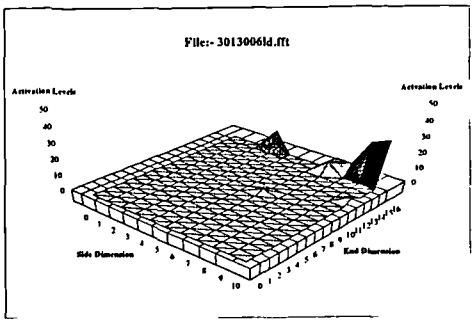


Figure. 4

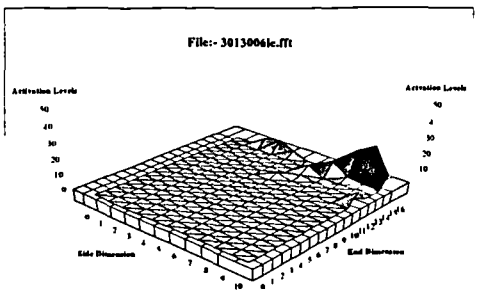


Figure. 5

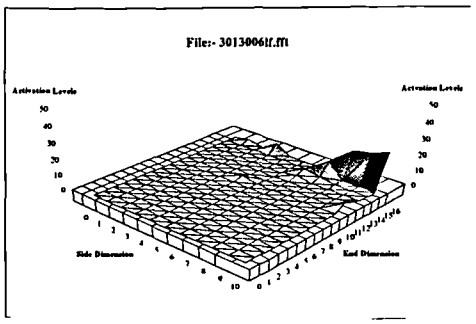


Figure. 6

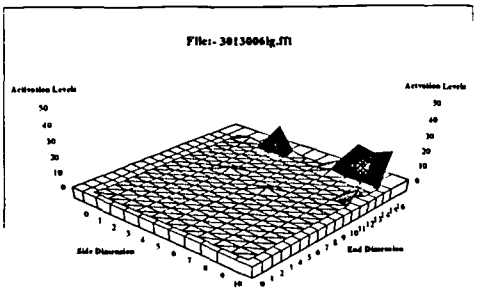


Figure. 7

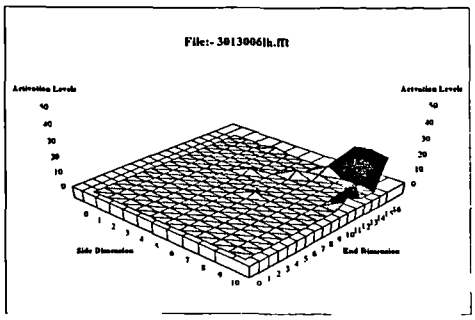


Figure. 8

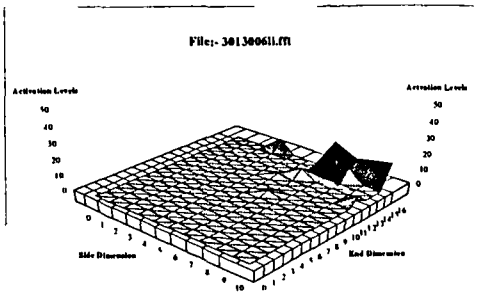


Figure. 9

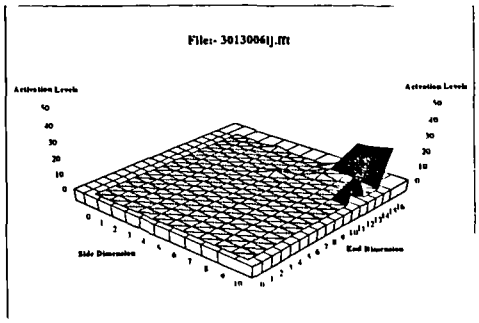


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 32V, 130 Wfs, 6.5mm/s Ts

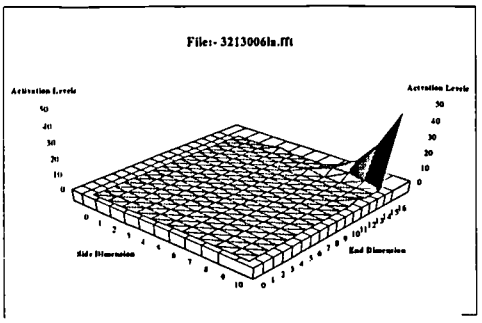


Figure. 1

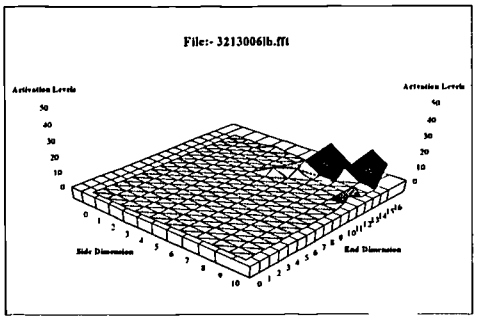


Figure. 2

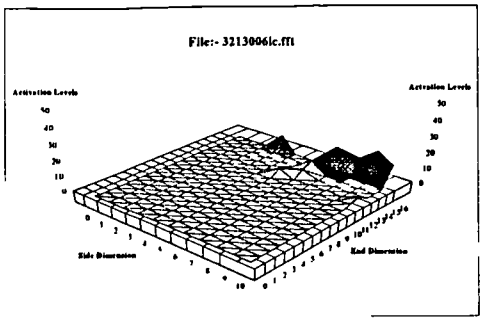


Figure. 3

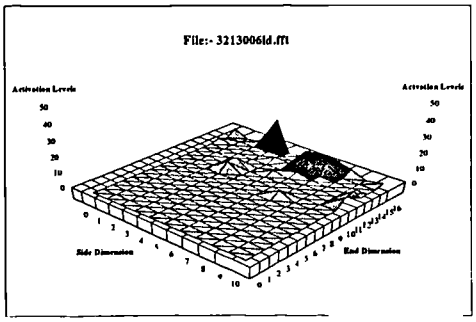


Figure. 4

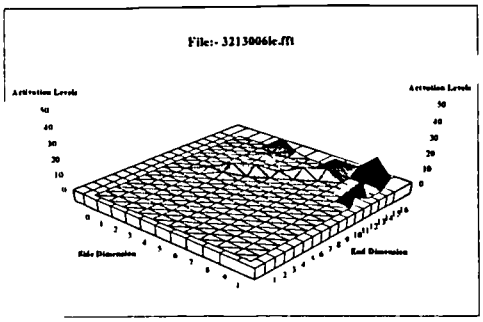


Figure. 5

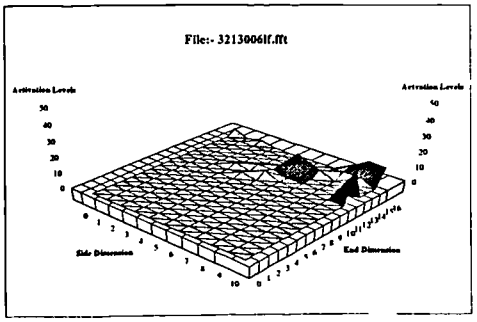


Figure. 6

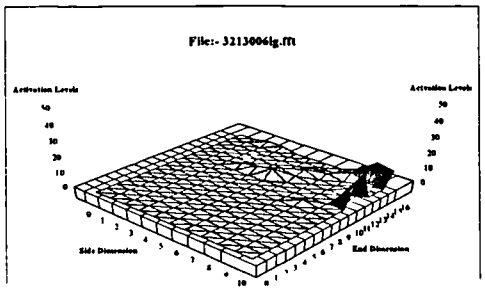


Figure. 7

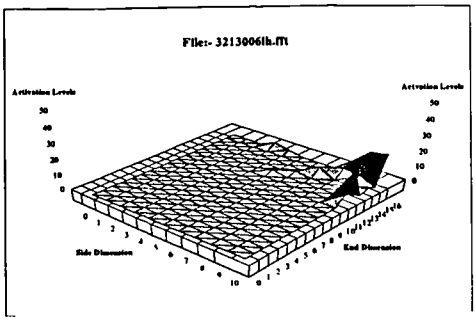


Figure. 8

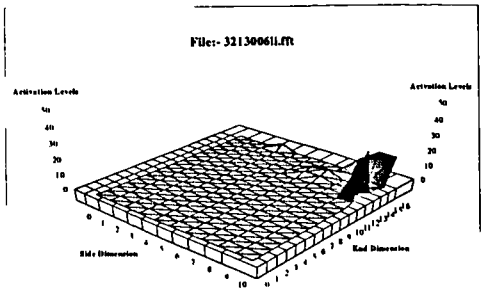


Figure. 9

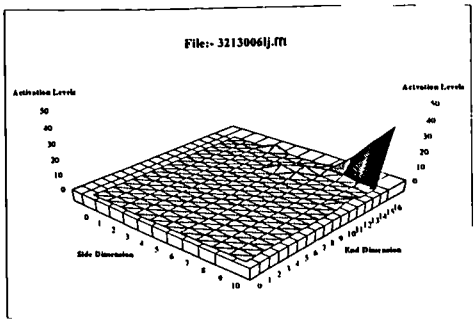


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 35V, 130 Wfs, 6.5mm/s Ts

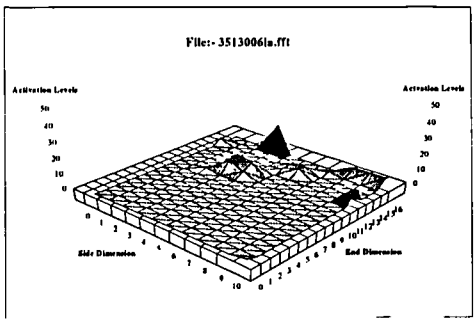


Figure. 1

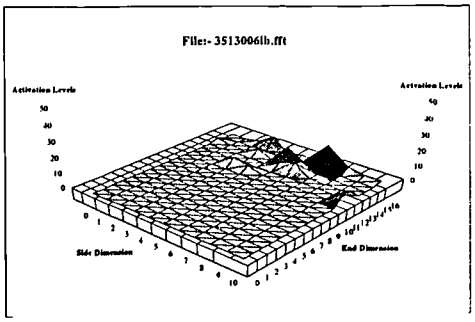


Figure. 2

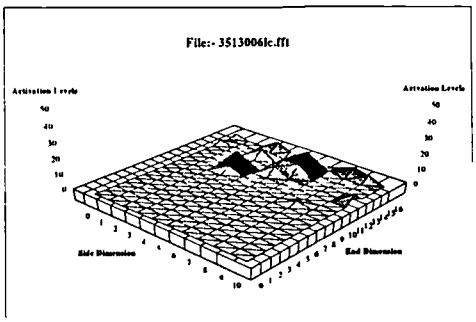


Figure. 3

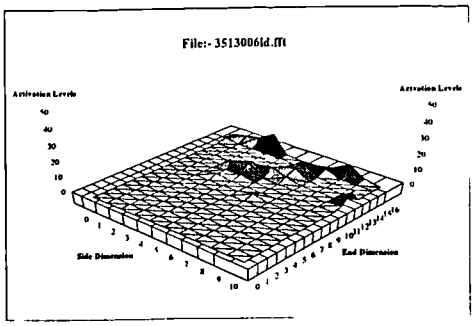


Figure. 4

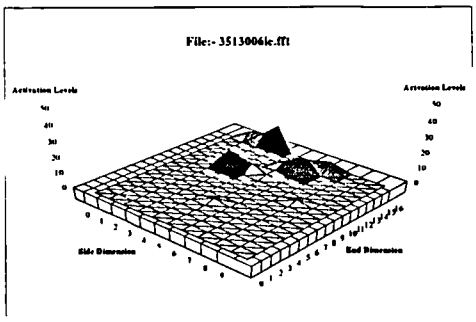


Figure. 5

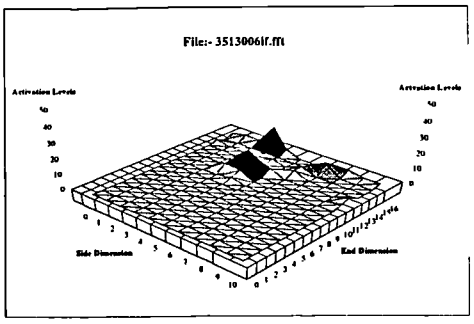


Figure. 6

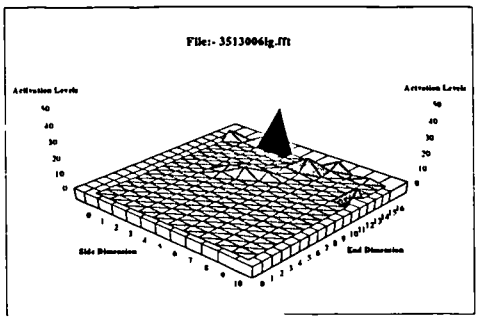


Figure. 7

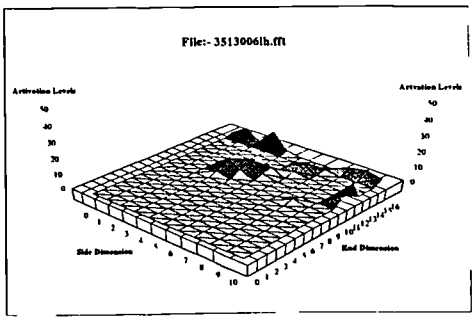


Figure. 8

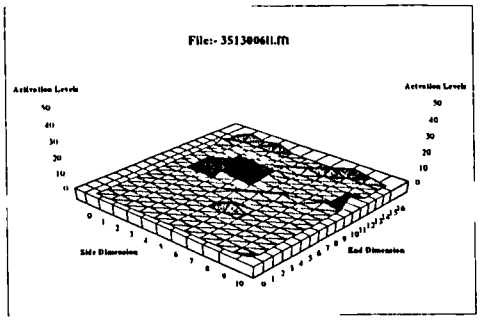


Figure. 9

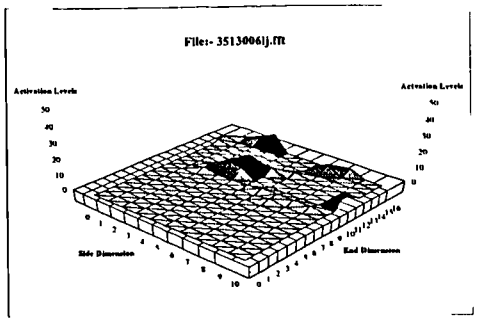


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 37V, 130 Wfs, 6.5mm/s Ts

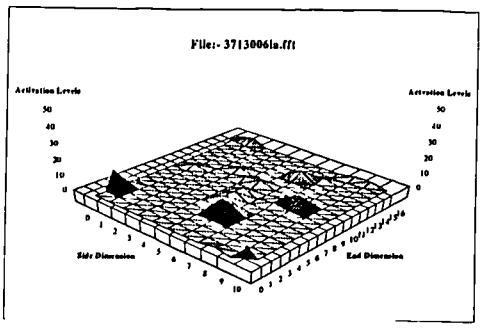


Figure. 1

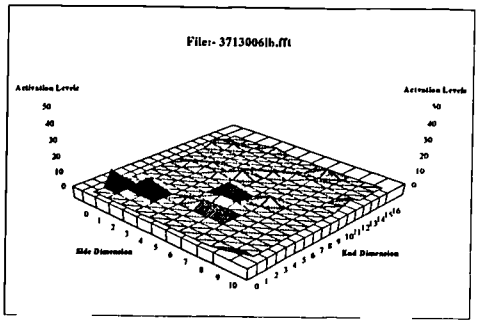


Figure. 2

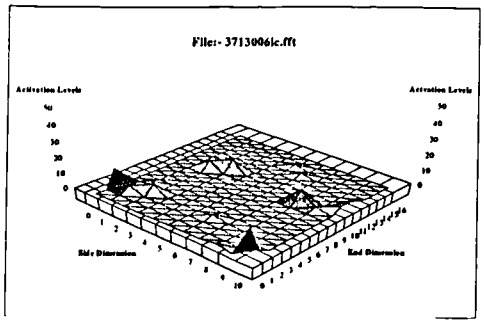


Figure. 3

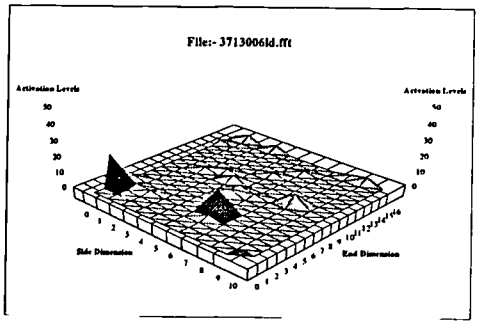


Figure. 4

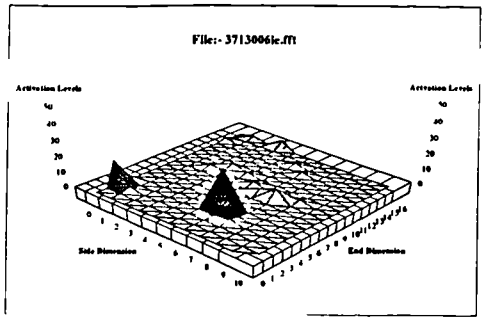


Figure. 5

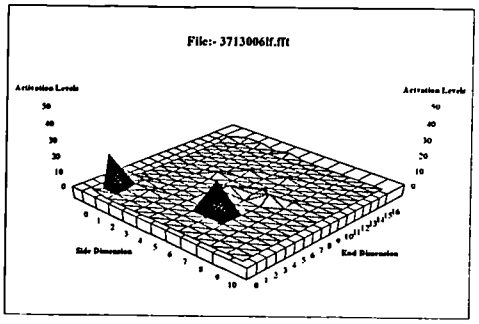


Figure. 6

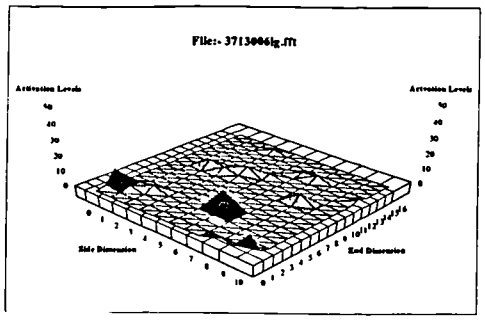


Figure. 7

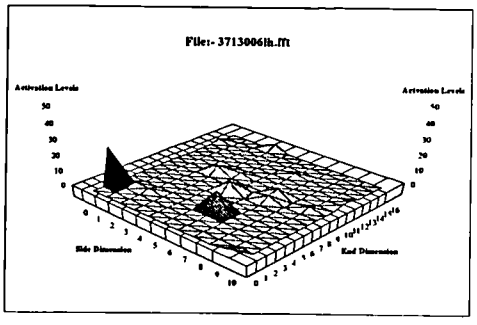


Figure. 8

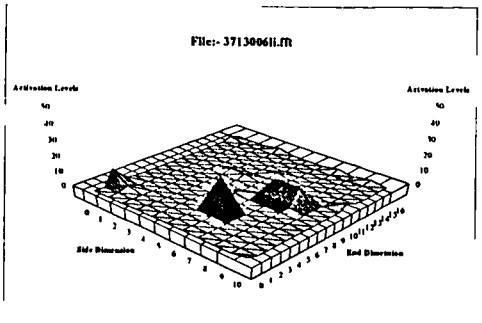


Figure. 9

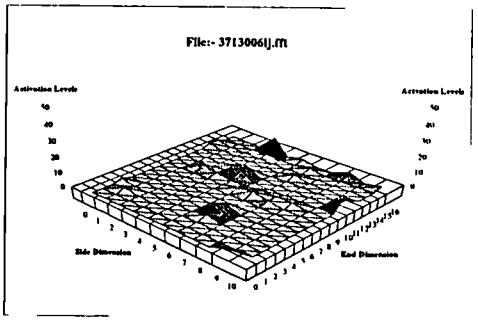


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 150 Wfs, 6.5mm/s Ts

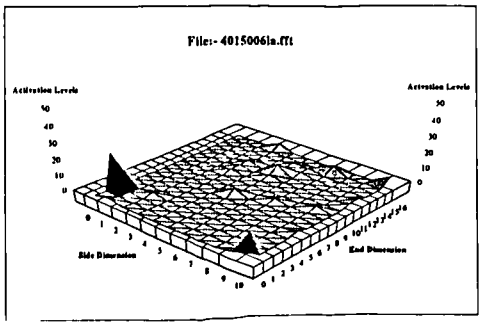


Figure. 1

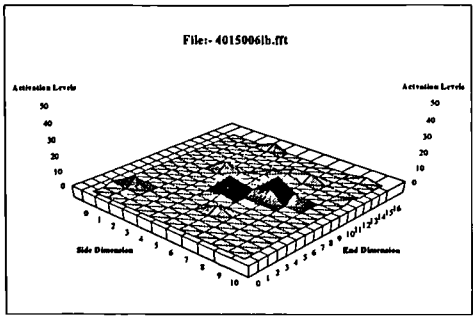


Figure. 2

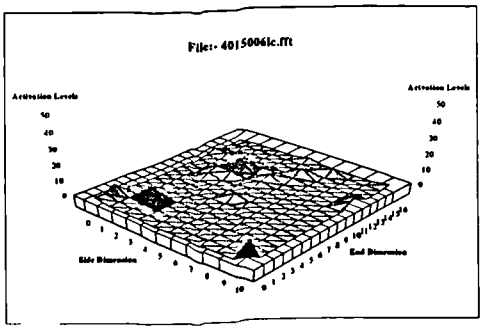


Figure. 3

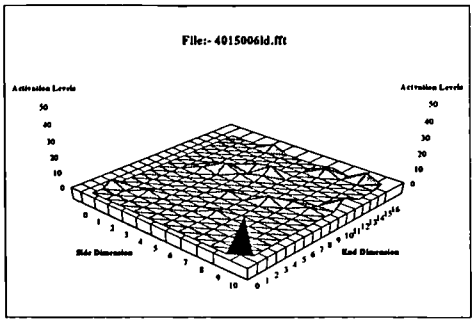


Figure. 4

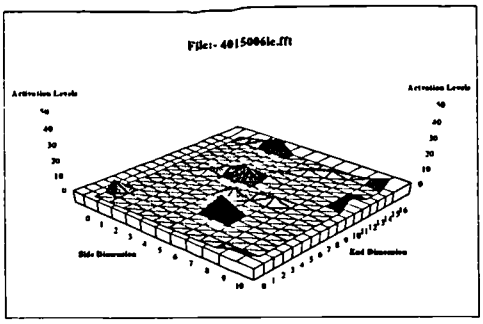


Figure. 5

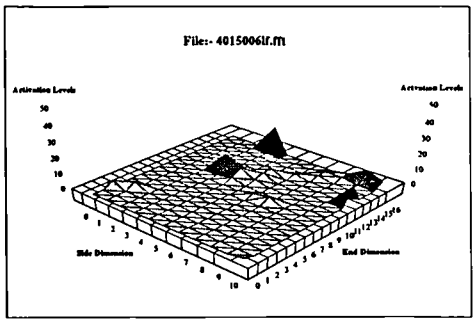


Figure. 6

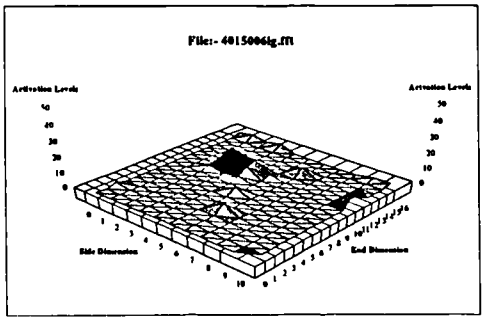


Figure. 7

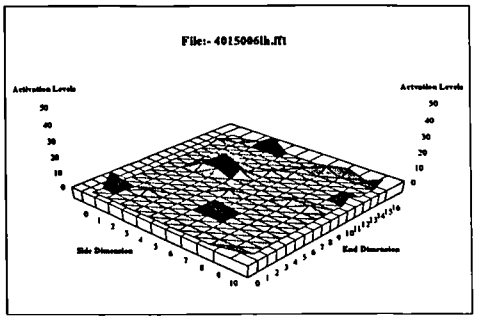


Figure. 8

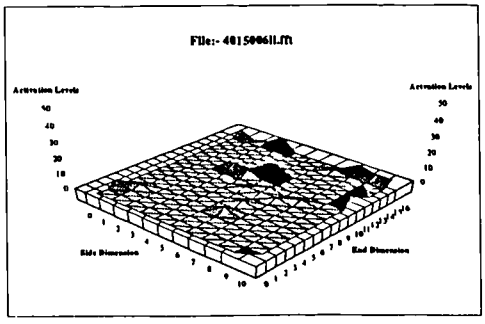


Figure. 9

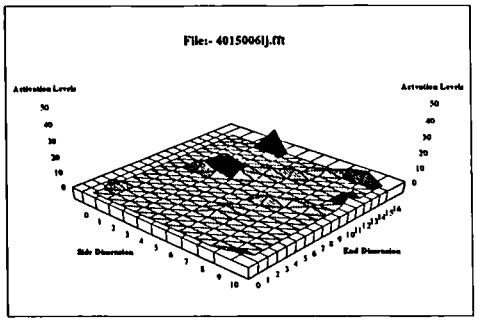


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 42V, 130 Wfs, 6.5mm/s Ts

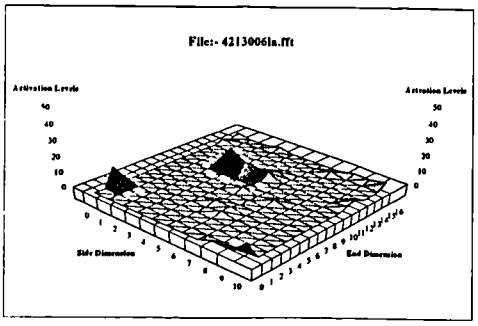


Figure. 1

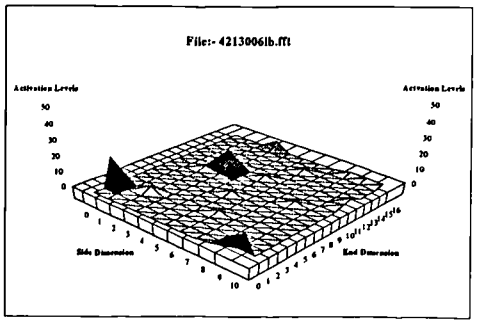


Figure. 2

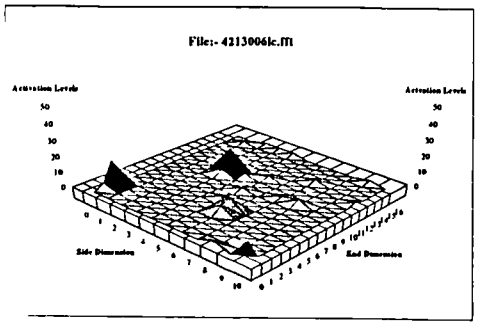


Figure. 3

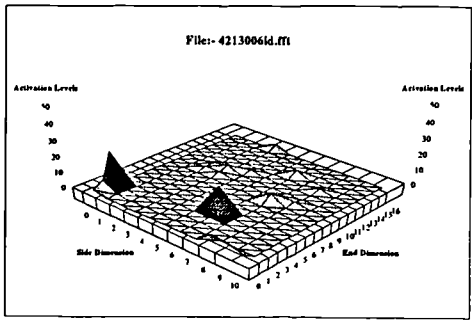


Figure. 4

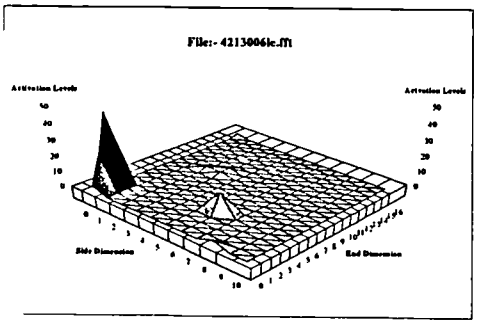


Figure. 5

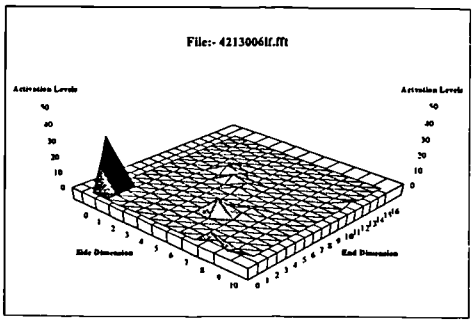


Figure. 6

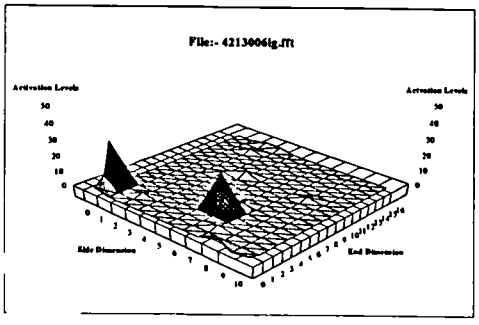


Figure. 7

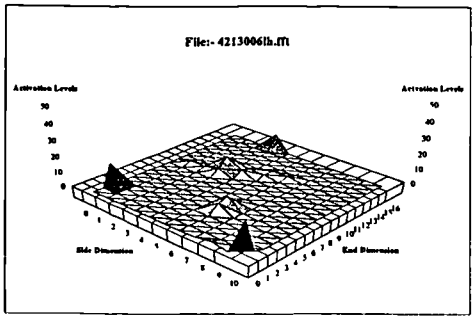


Figure. 8

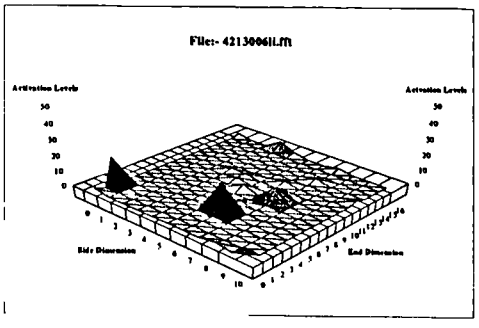


Figure. 9

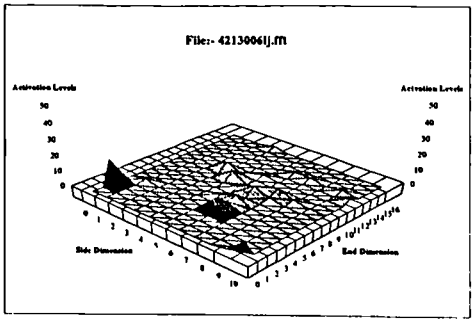


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 45V, 130 Wfs, 6.5mm/s Ts

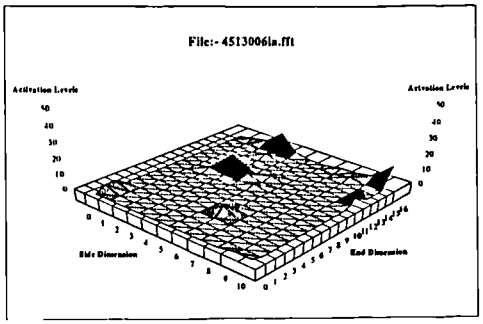


Figure. 1

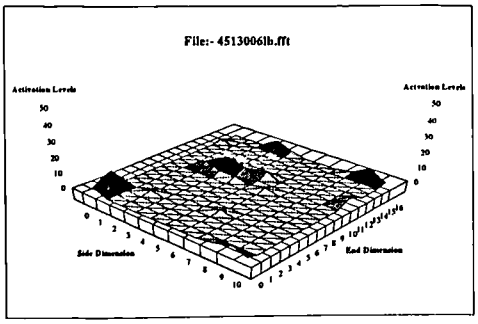


Figure. 2

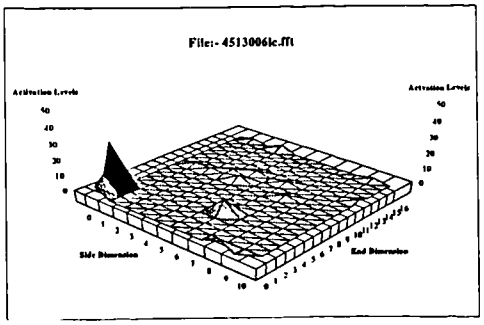


Figure. 3

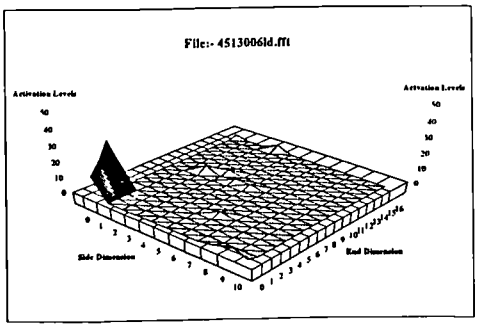


Figure. 4

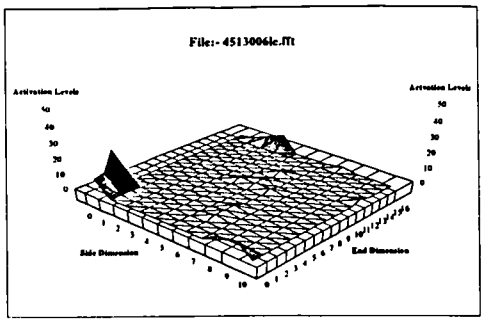


Figure. 5

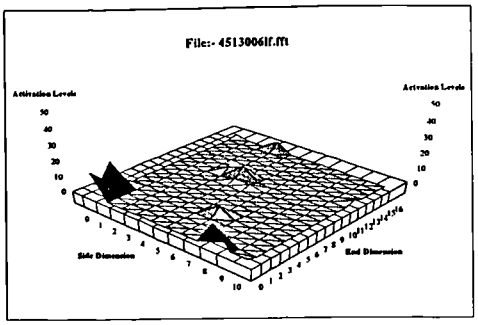


Figure. 6

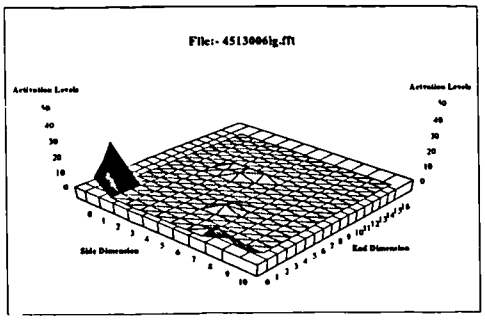


Figure. 7

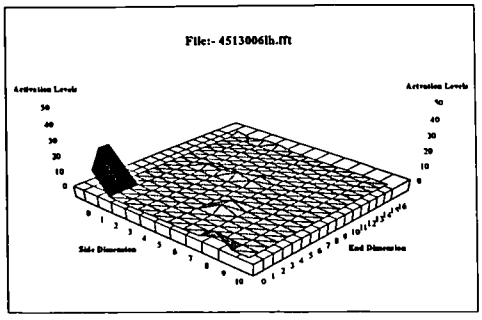


Figure. 8

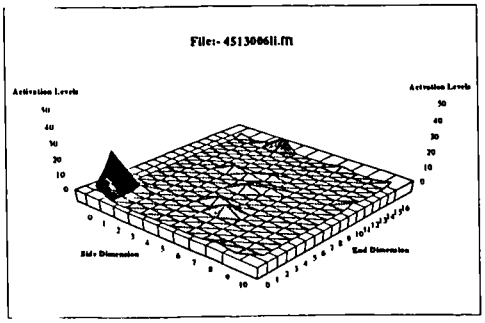


Figure. 9

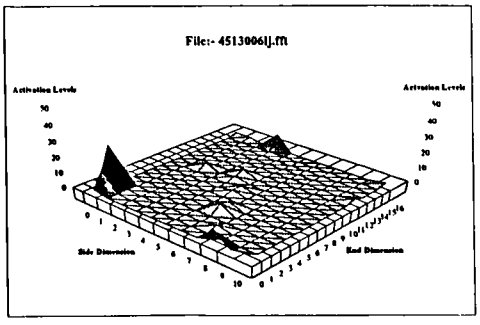


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 47V, 130 Wfs, 6.5mm/s Ts

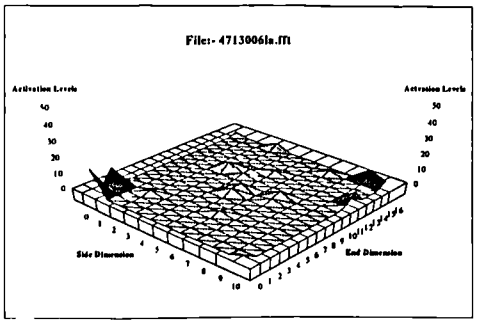


Figure. 1

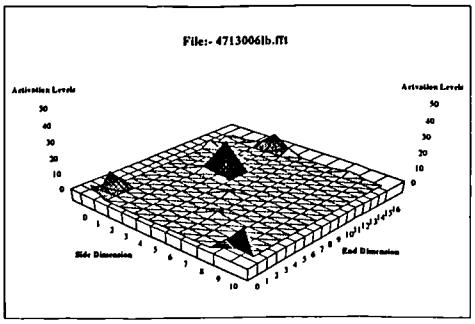


Figure. 2

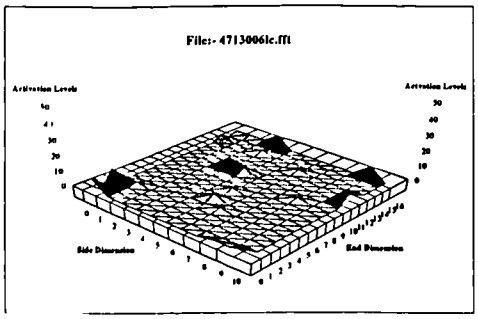


Figure. 3

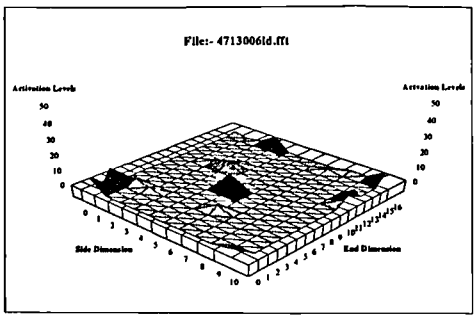


Figure. 4

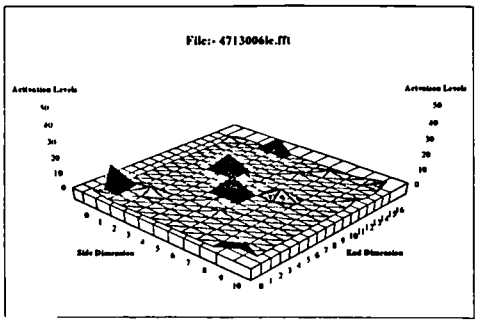


Figure. 5

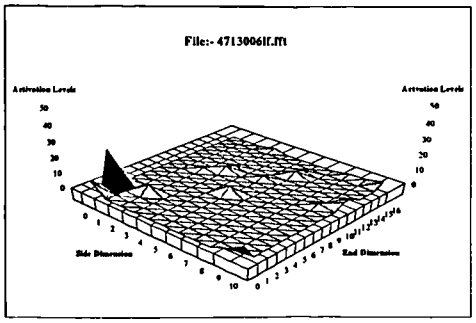


Figure. 6

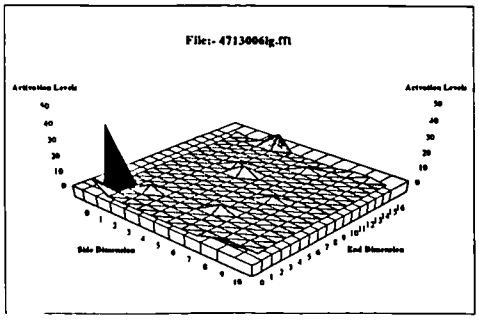


Figure. 7

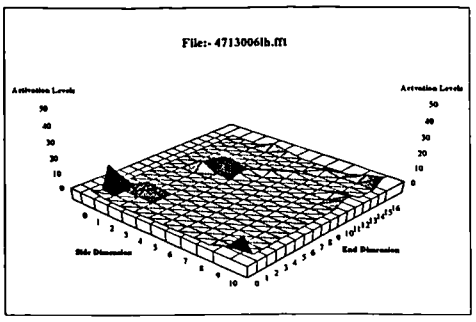


Figure. 8

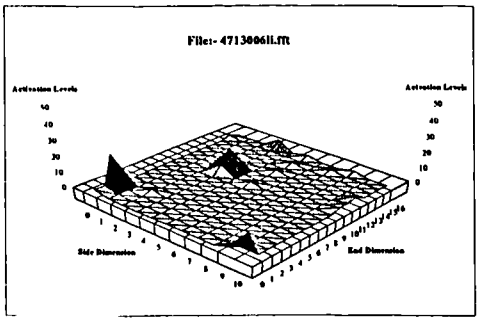


Figure. 9

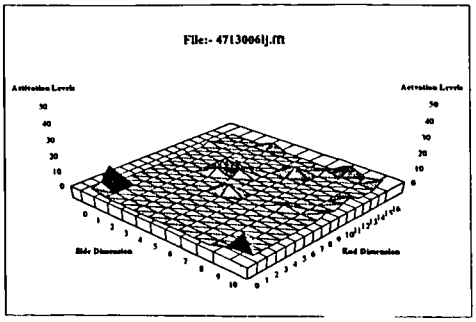


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 25 Wfs, 6.5mm/s Ts

Wire feed speed too low to support arc strike

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 50 Wfs, 6.5mm/s Ts

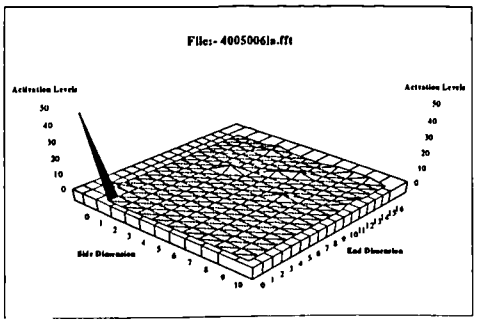


Figure. 1

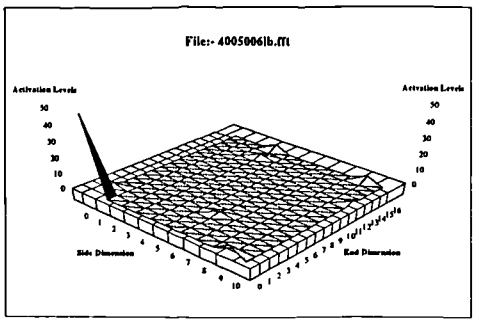


Figure. 2

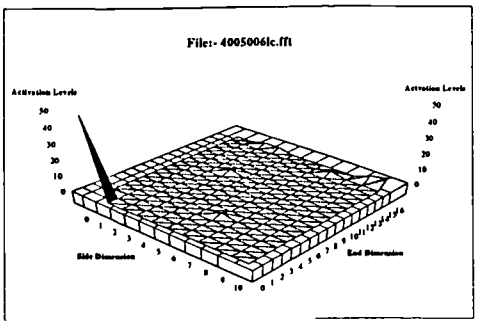


Figure. 3

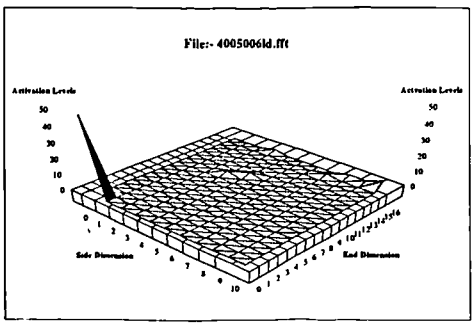


Figure. 4

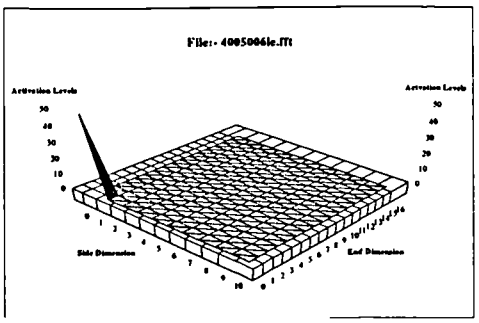


Figure. 5

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 75 Wfs, 6.5mm/s Ts

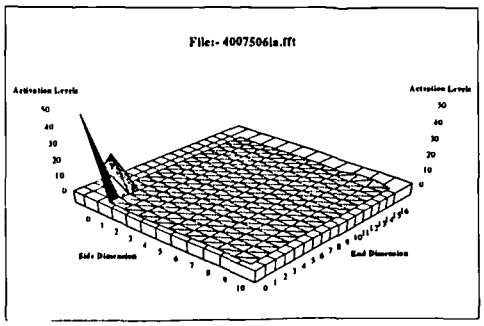


Figure. 1

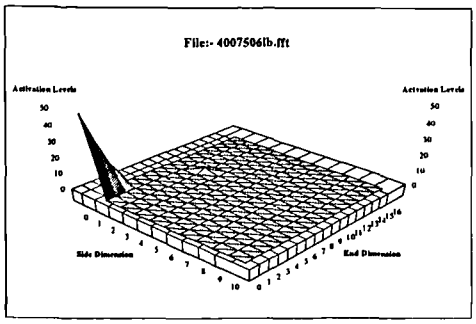


Figure. 2

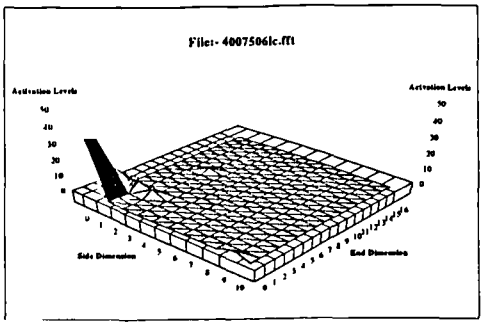


Figure. 3

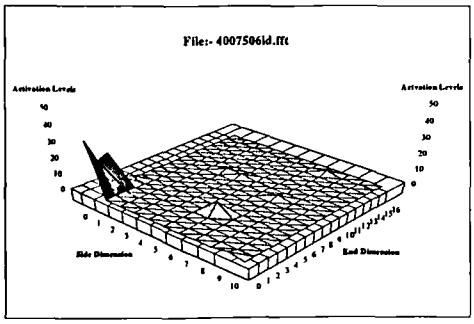


Figure. 4

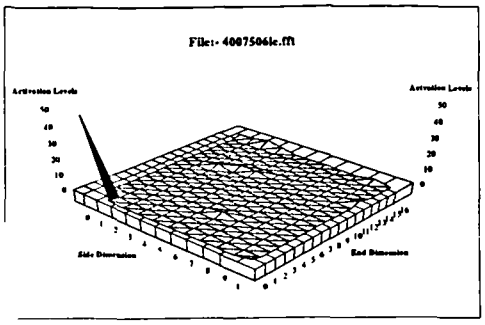


Figure. 5

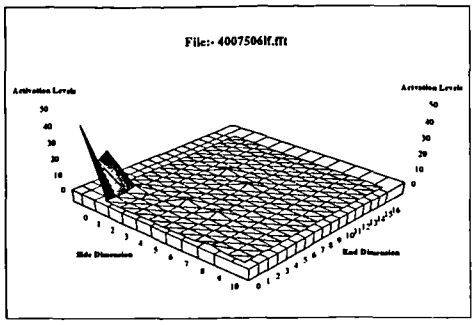


Figure. 6

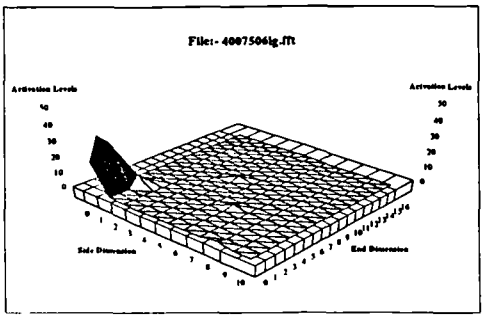


Figure. 7

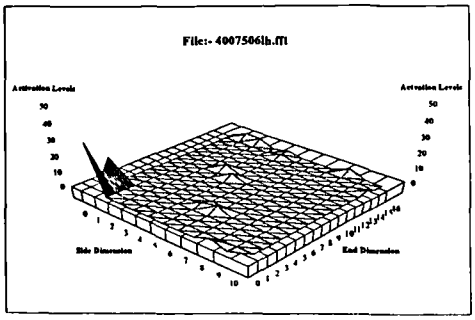


Figure. 8

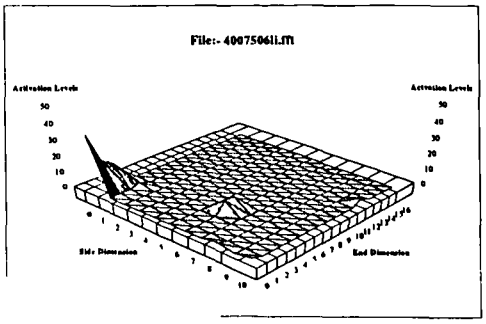


Figure. 9

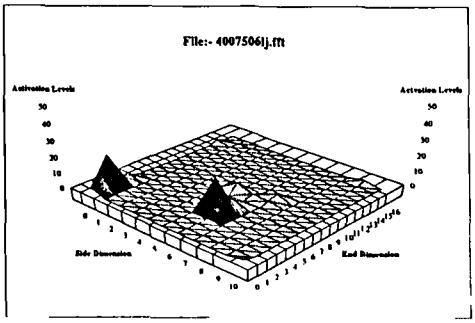


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 100 Wfs, 6.5mm/s Ts

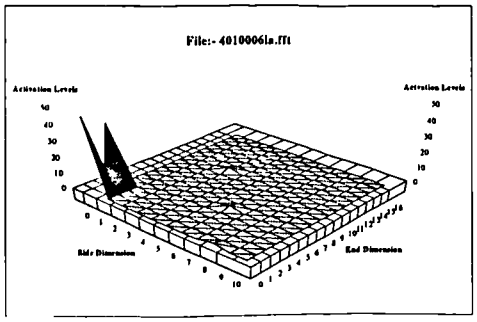


Figure. 1

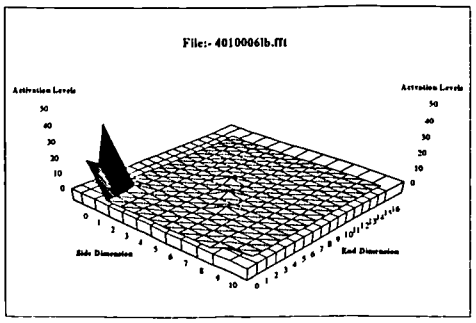


Figure. 2

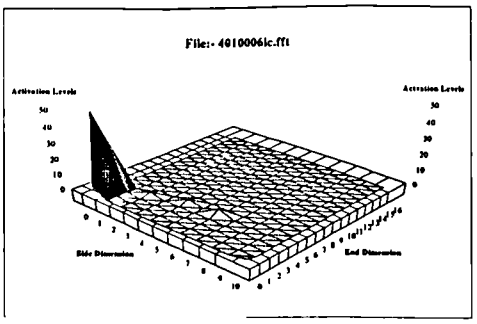


Figure. 3

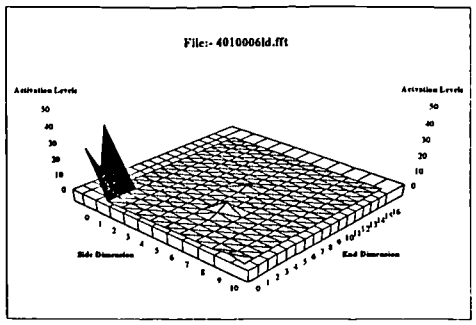


Figure. 4

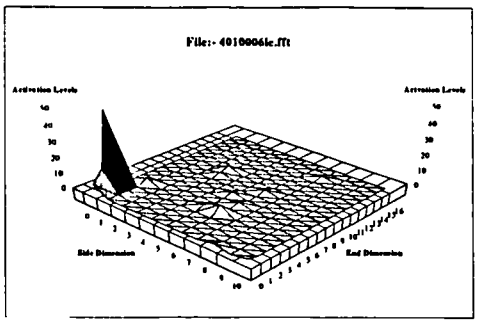


Figure. 5

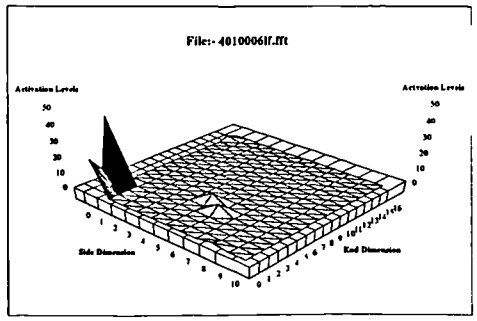


Figure. 6

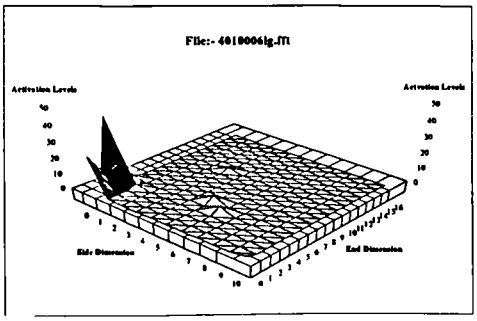


Figure. 7

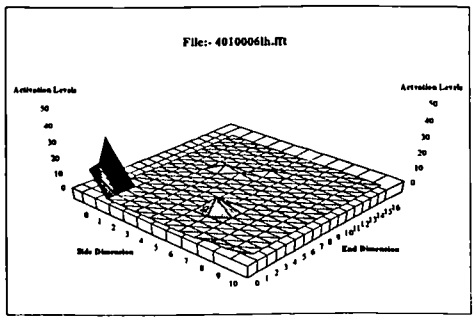


Figure. 8

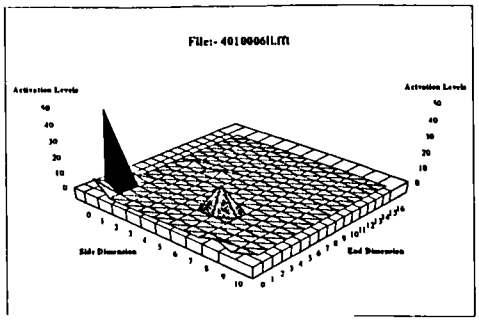


Figure. 9

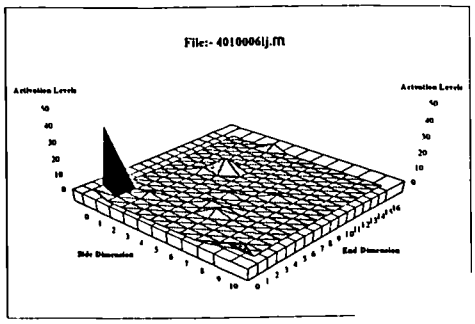


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 125 Wfs, 6.5mm/s Ts

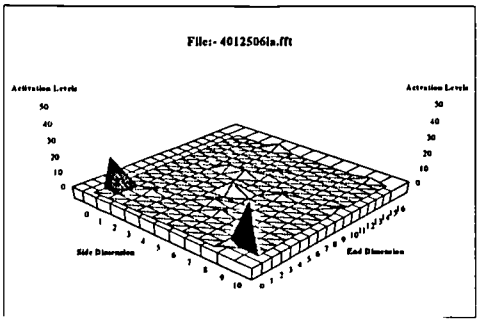


Figure. 1

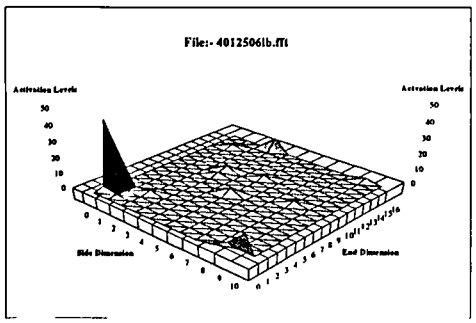


Figure. 2

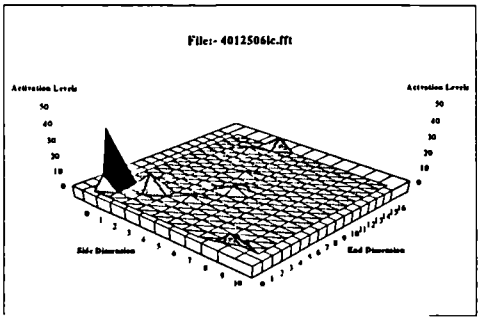


Figure. 3

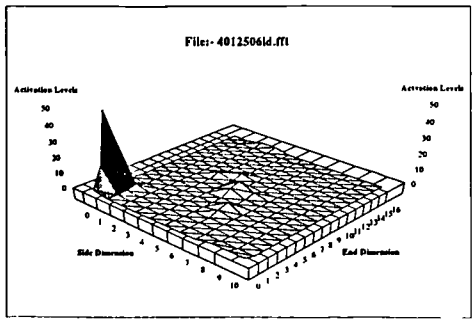


Figure. 4

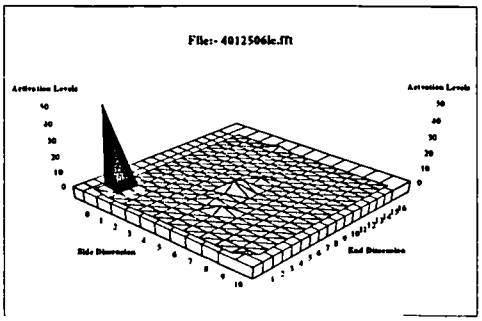


Figure. 5

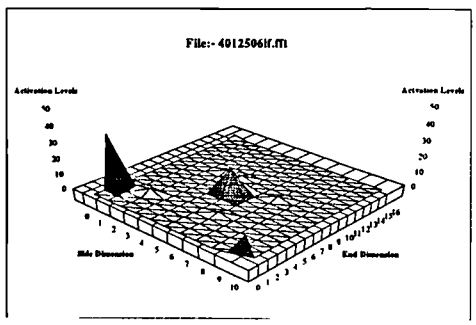


Figure. 6

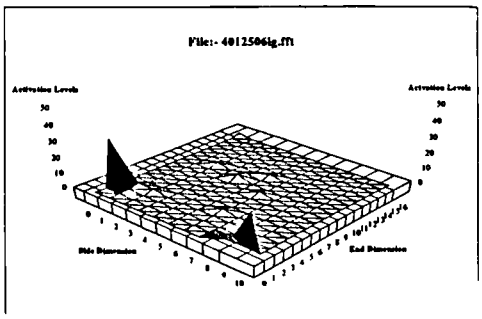


Figure. 7

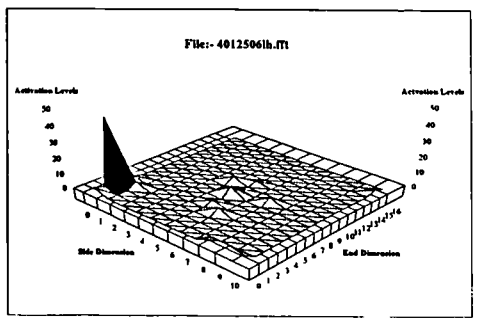


Figure. 8

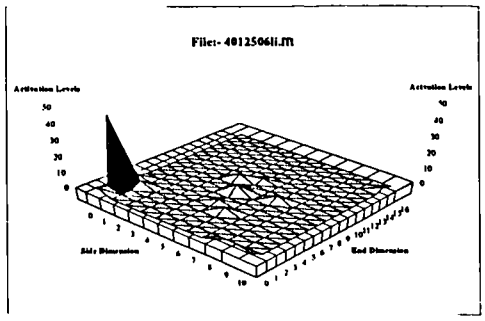


Figure. 9

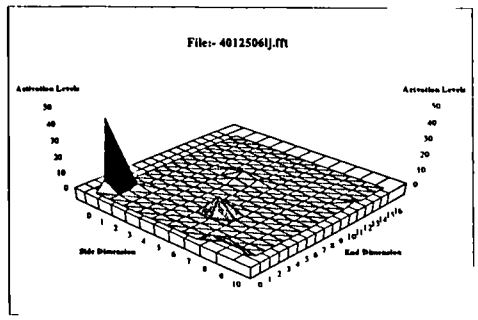


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 150 Wfs, 6.5mm/s Ts

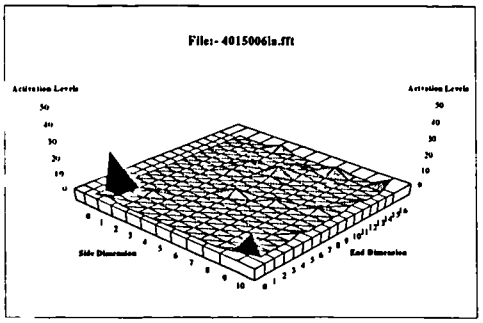


Figure. 1

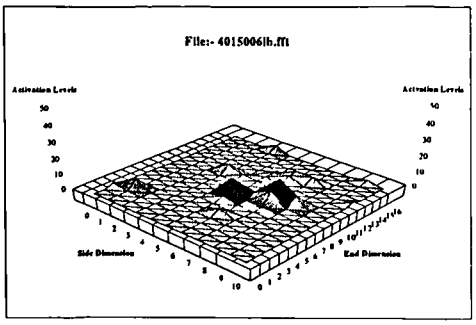


Figure. 2

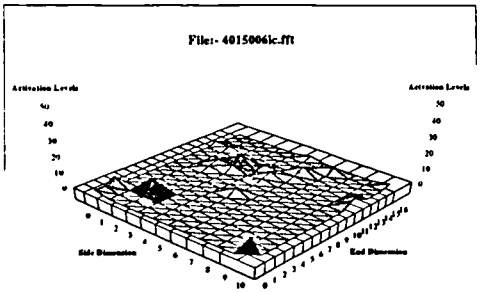


Figure. 3

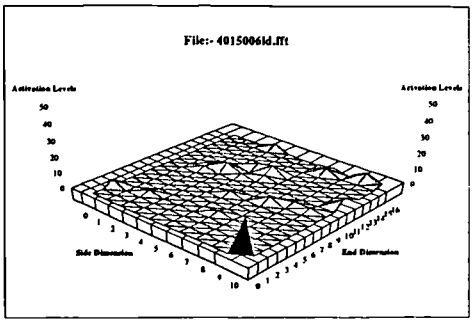


Figure. 4

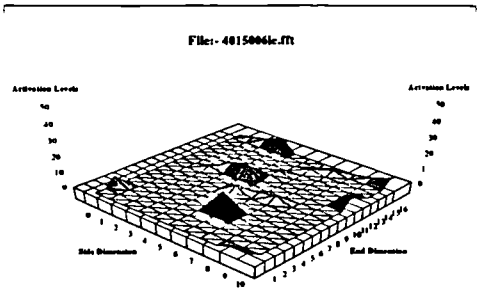


Figure. 5

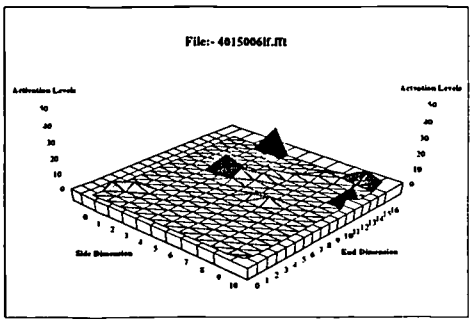


Figure. 6

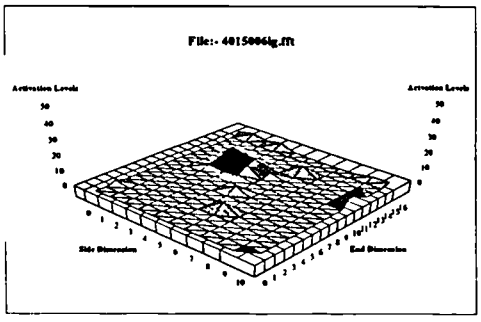


Figure. 7

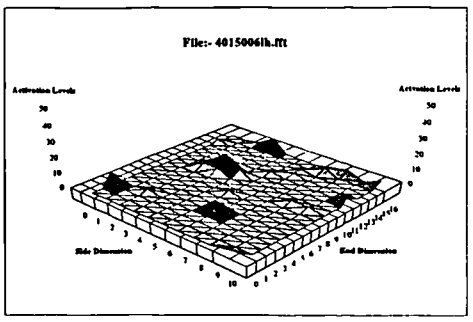


Figure. 8

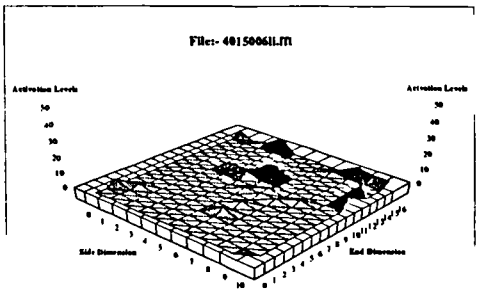


Figure. 9

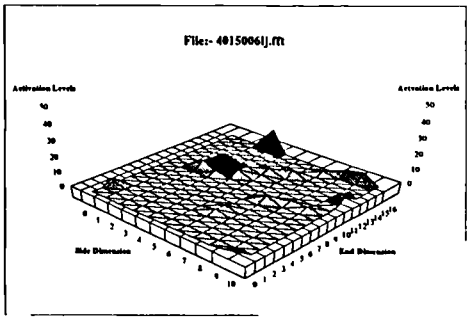


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 175 Wfs, 6.5mm/s Ts

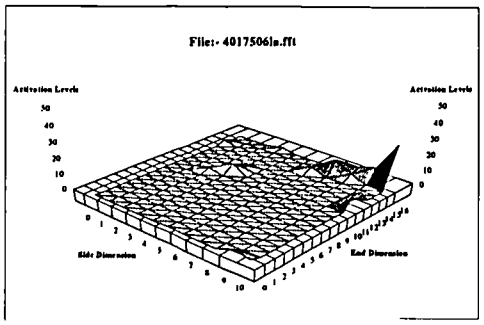


Figure. 1

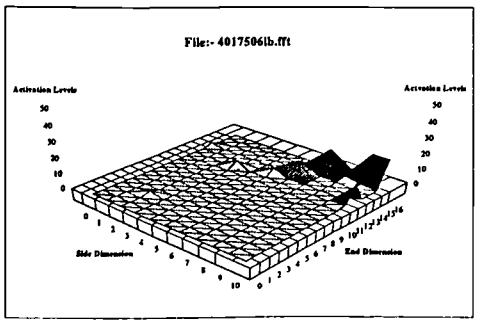


Figure. 2

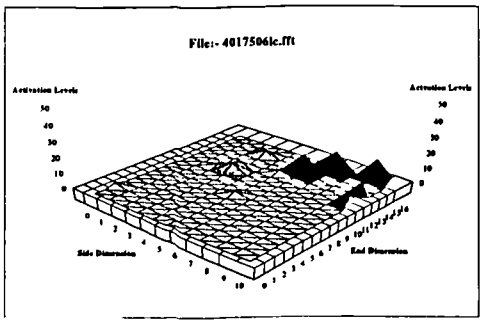


Figure. 3

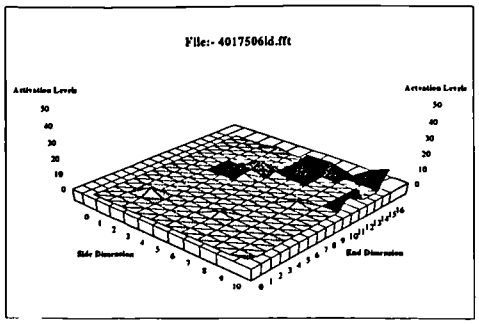


Figure. 4

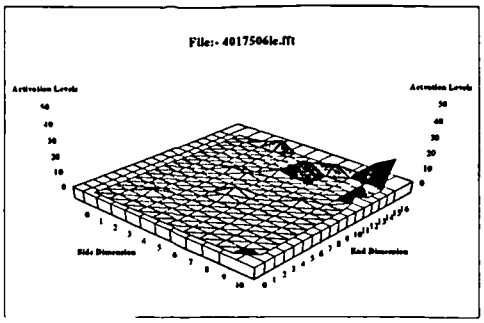


Figure. 5

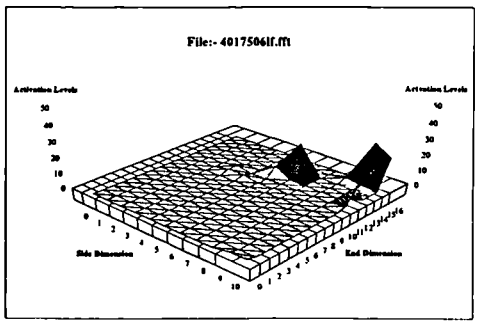


Figure. 6

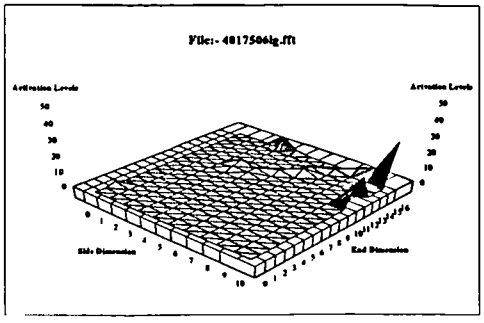


Figure. 7

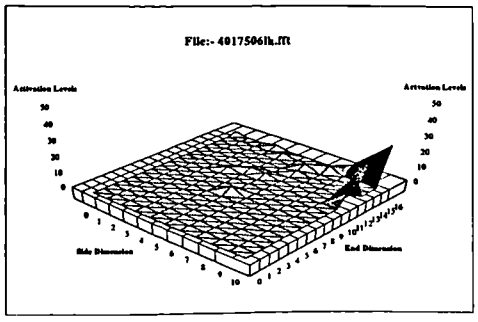


Figure. 8

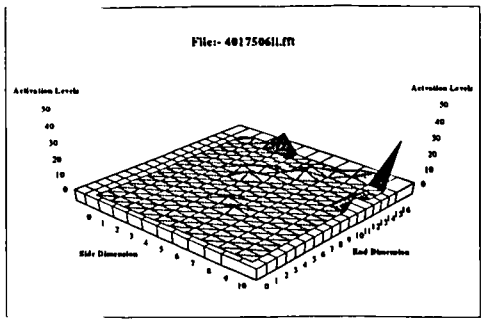


Figure. 9

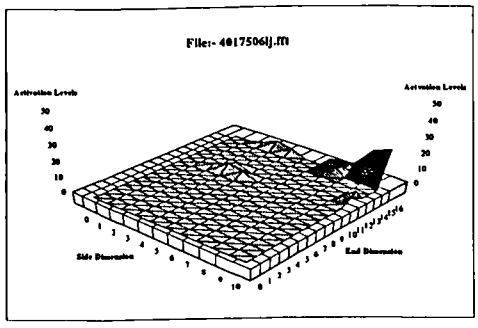


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 200 Wfs, 6.5mm/s Ts

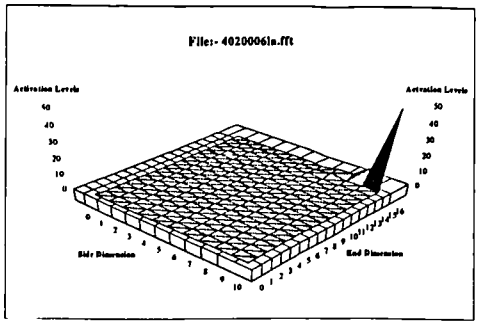


Figure. 1

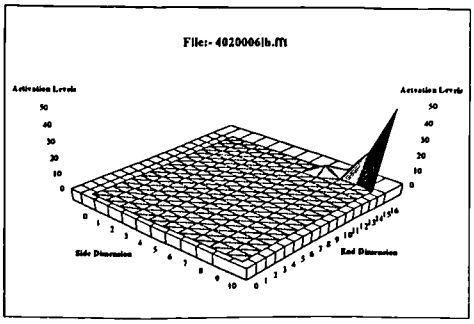


Figure. 2

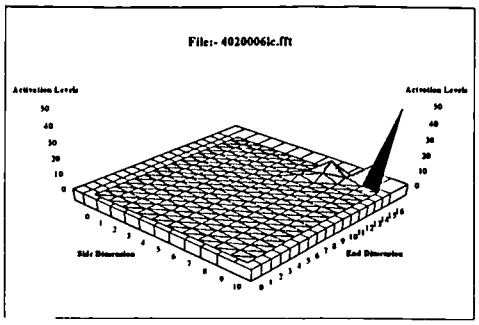


Figure. 3

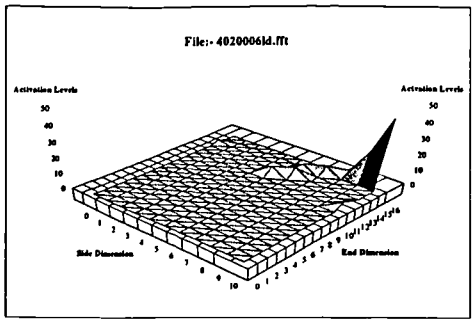


Figure. 4

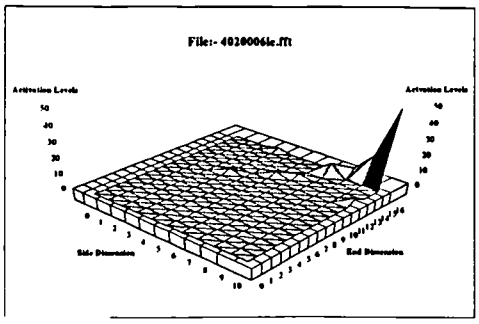


Figure. 5

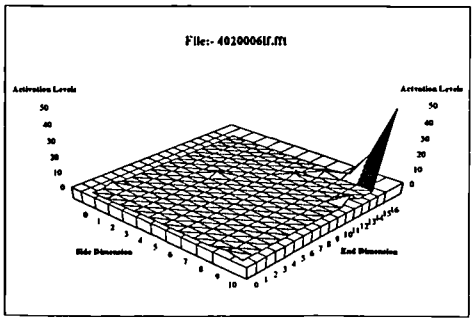


Figure. 6

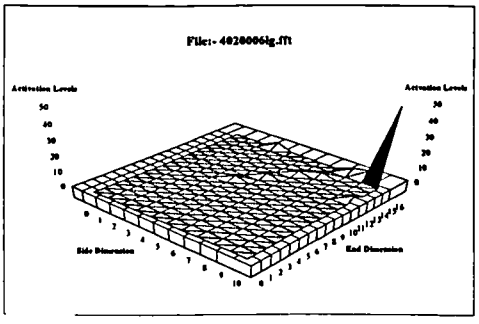


Figure. 7

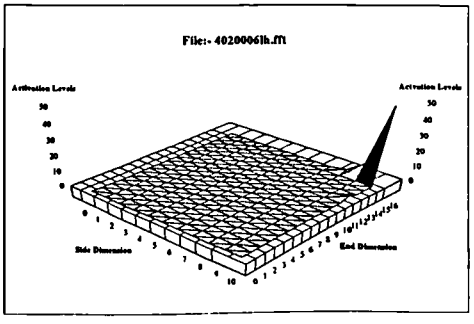


Figure. 8

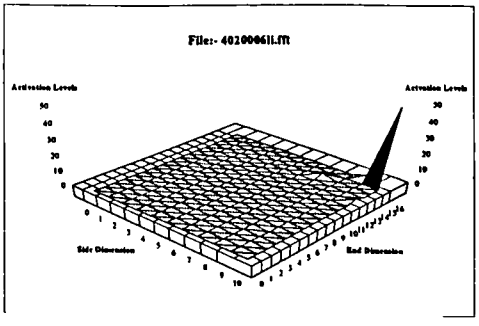


Figure. 9

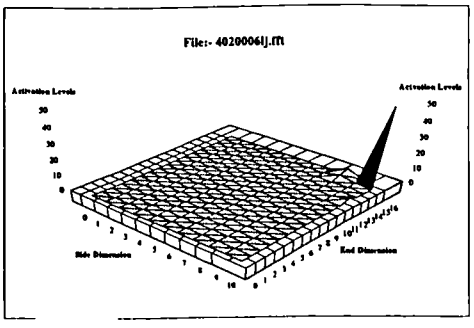


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 225 Wfs, 6.5mm/s Ts

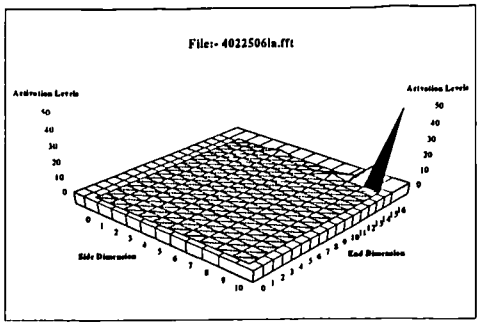


Figure. 1

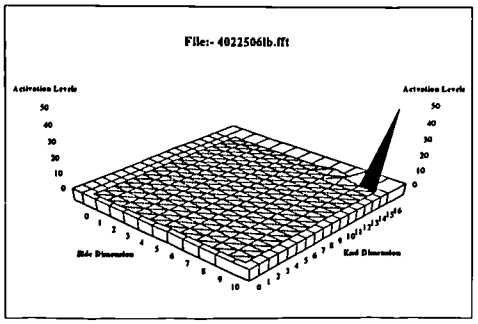


Figure. 2

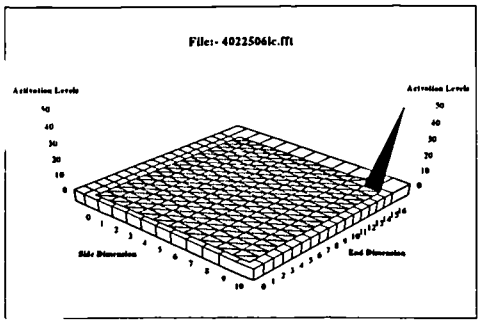


Figure. 3

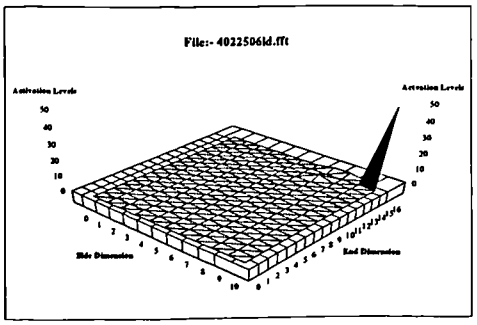


Figure. 4

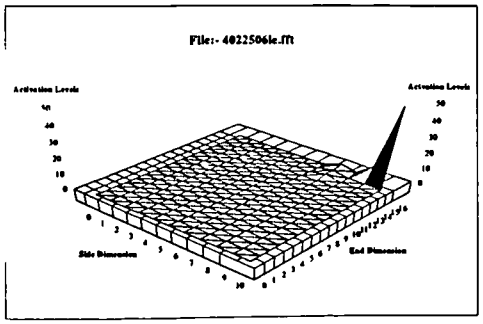


Figure. 5

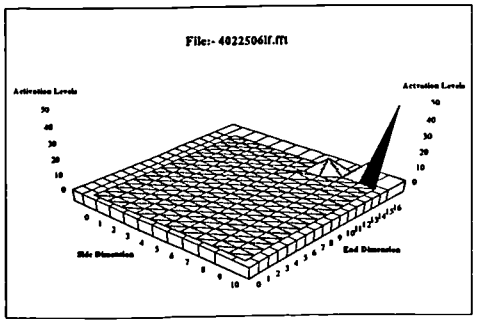


Figure. 6

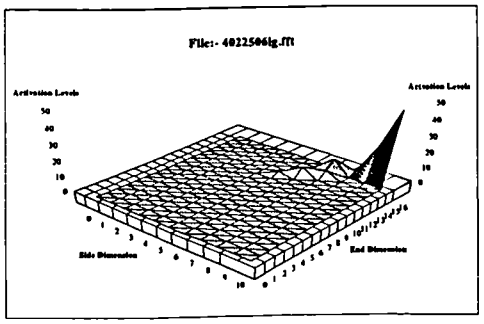


Figure. 7

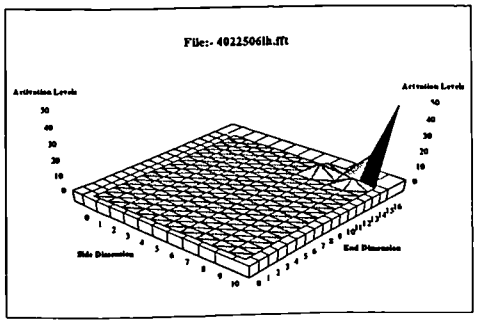


Figure. 8

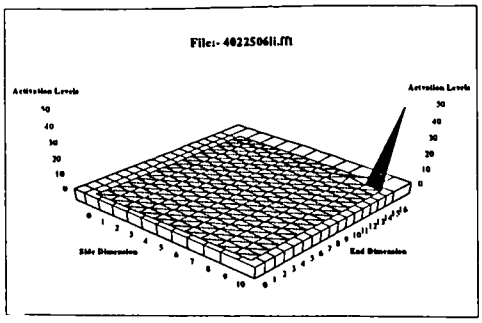


Figure. 9

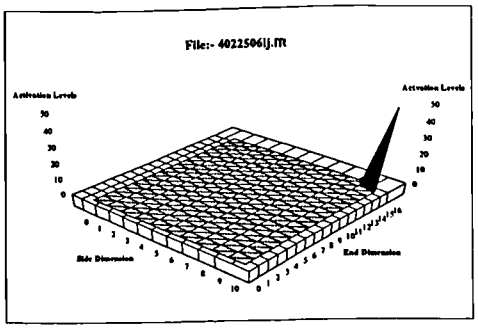


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 250 Wfs, 6.5mm/s Ts

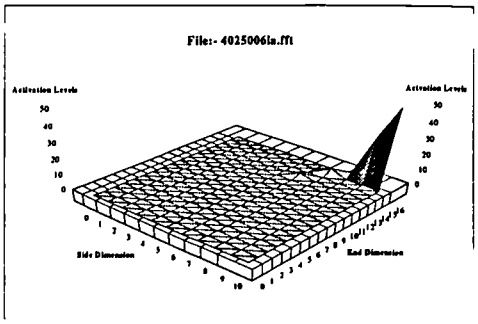


Figure. 1

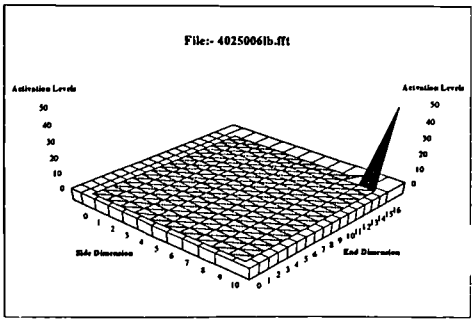


Figure. 2

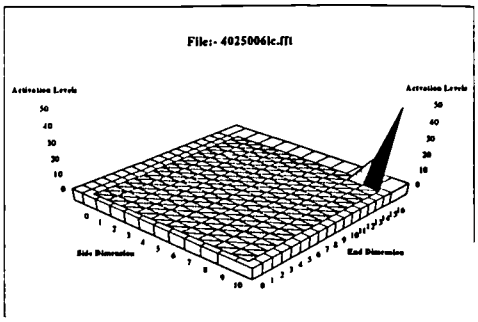


Figure. 3

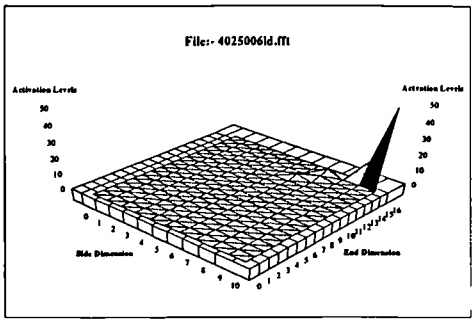


Figure. 4

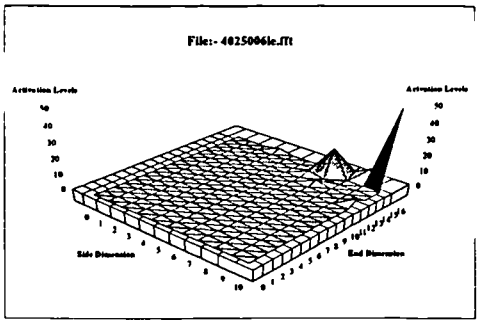


Figure. 5

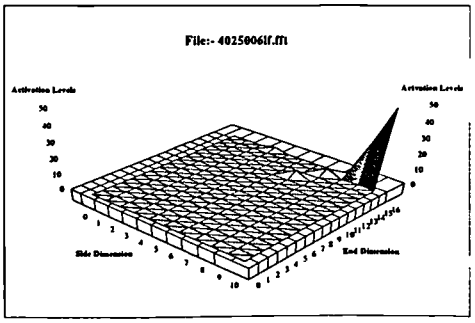


Figure. 6

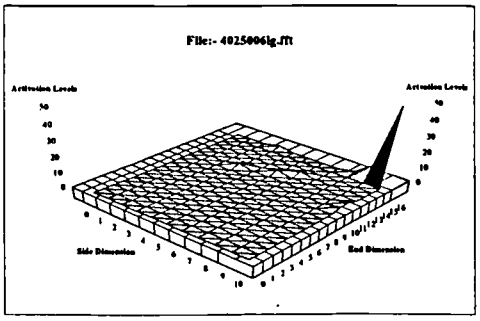


Figure. 7

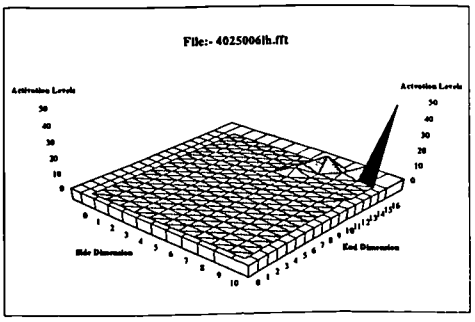


Figure. 8

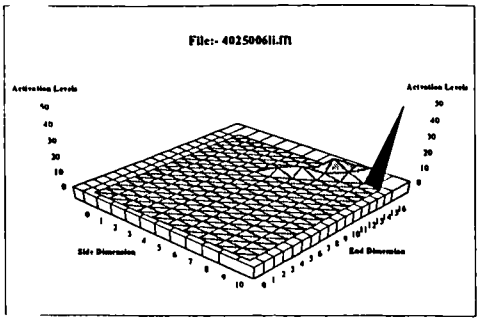


Figure. 9

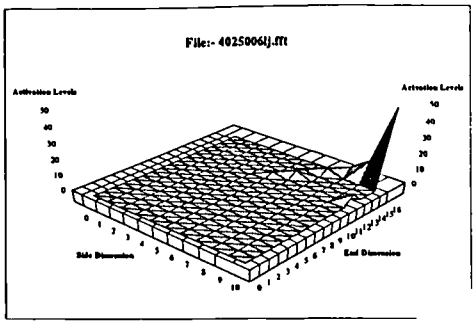


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 5mm/s Ts

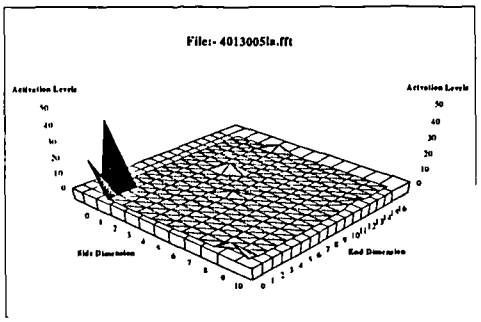


Figure. 1

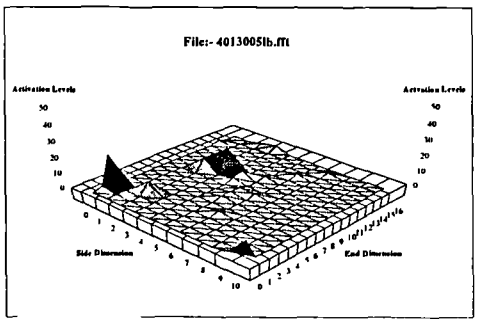


Figure. 2

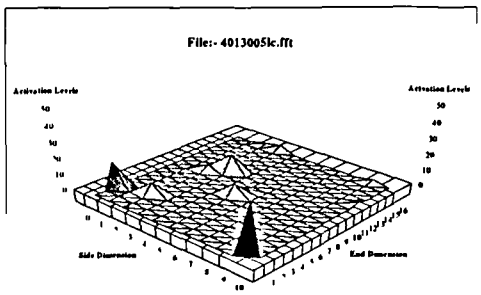


Figure. 3

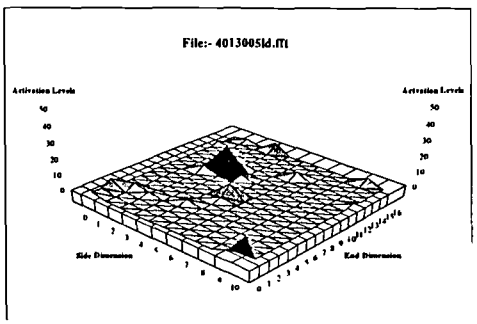


Figure. 4

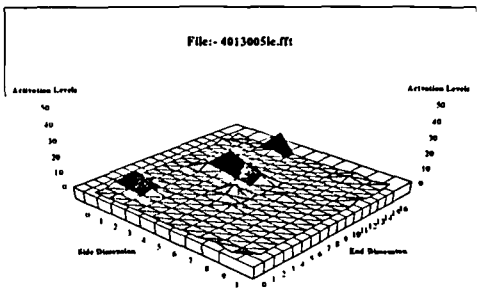


Figure. 5

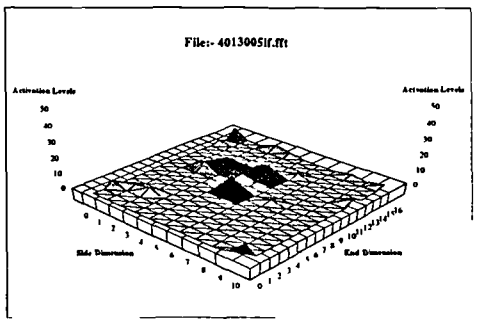


Figure. 6

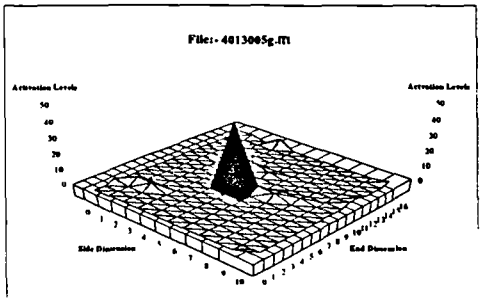


Figure. 7

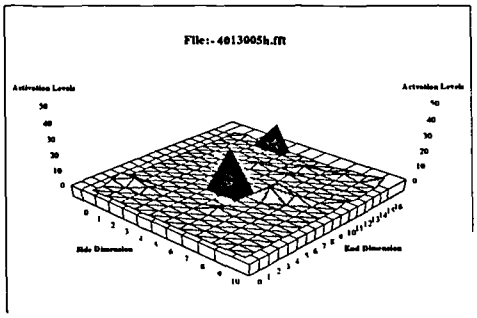


Figure. 8

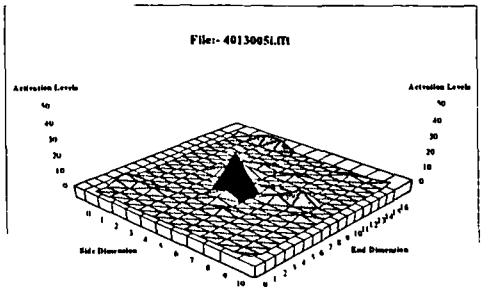


Figure. 9

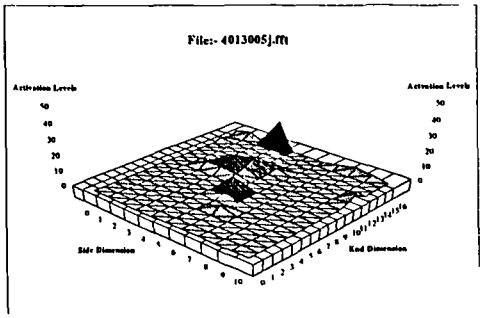


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 7.5mm/s Ts

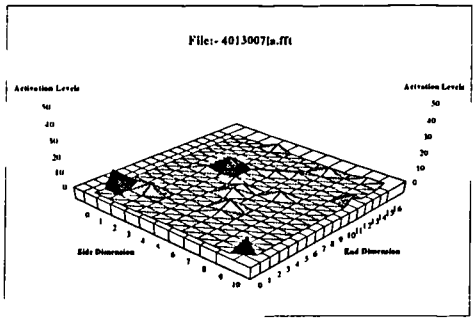


Figure. 1

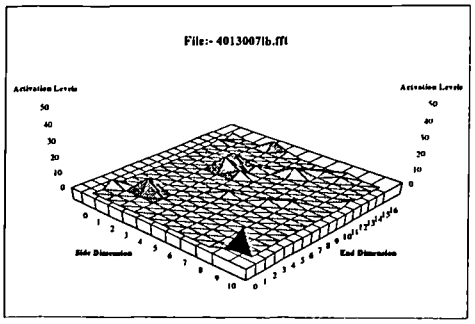


Figure. 2

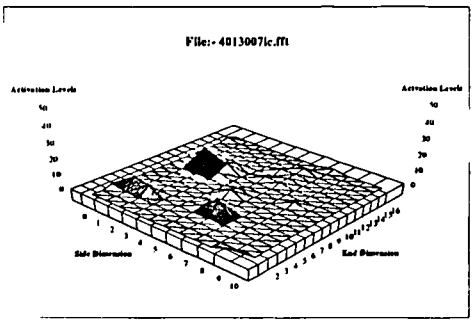


Figure. 3

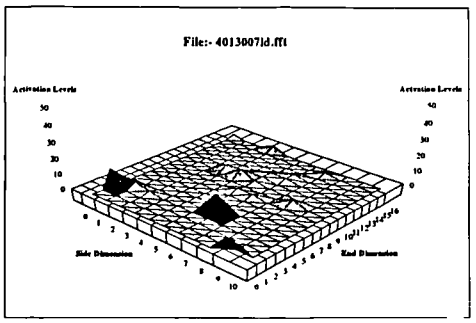


Figure. 4

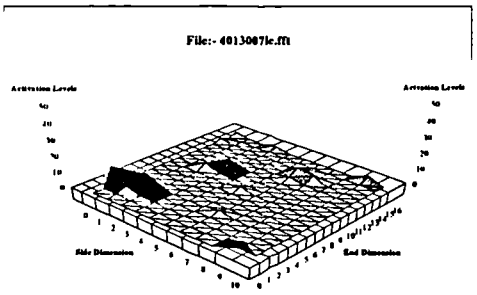


Figure. 5

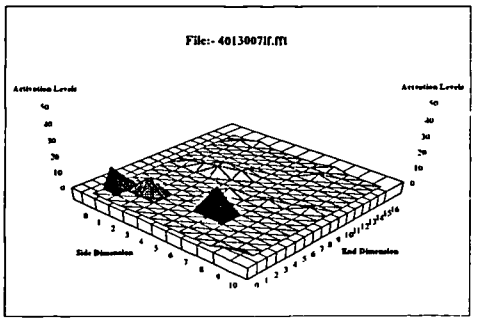


Figure. 6

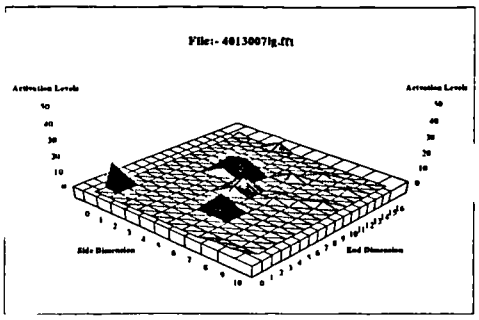


Figure. 7

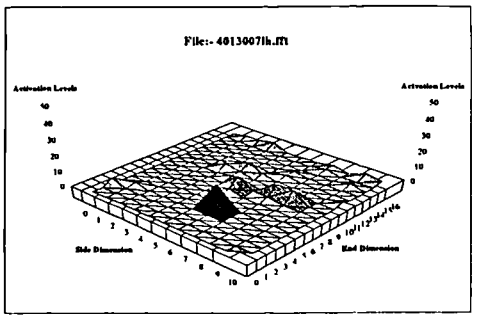


Figure. 8

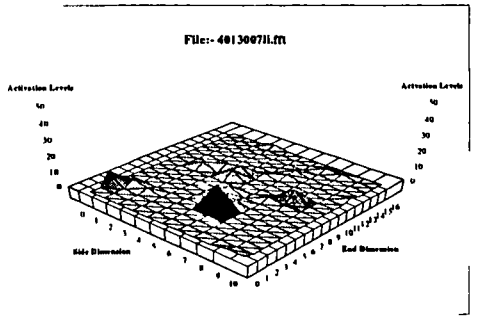


Figure. 9

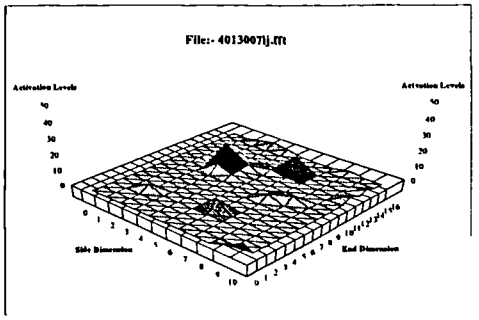


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 10 mm/s Ts

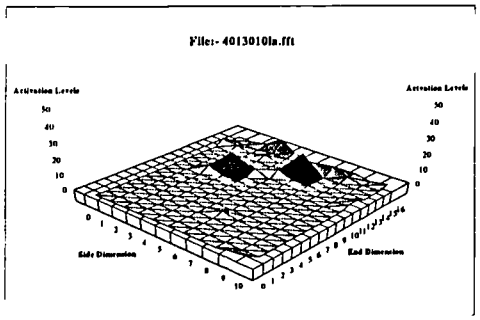


Figure. 1

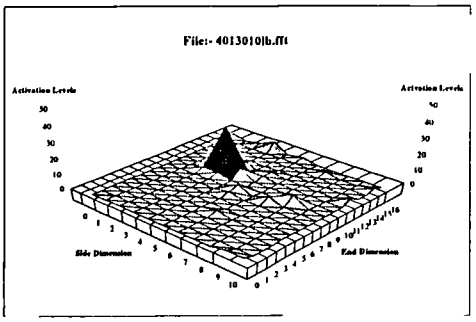


Figure. 2

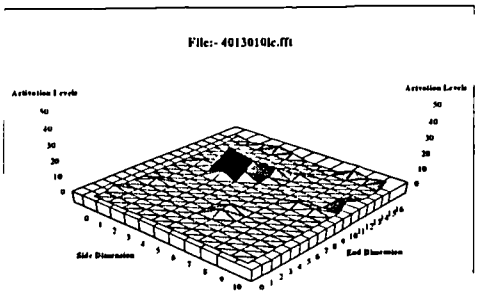


Figure. 3

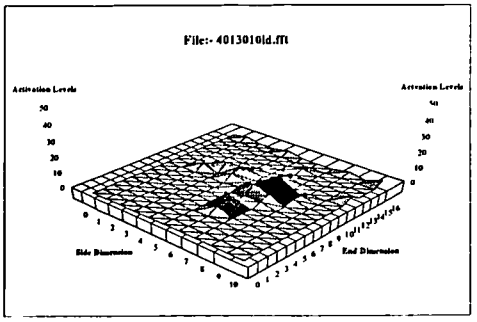


Figure. 4

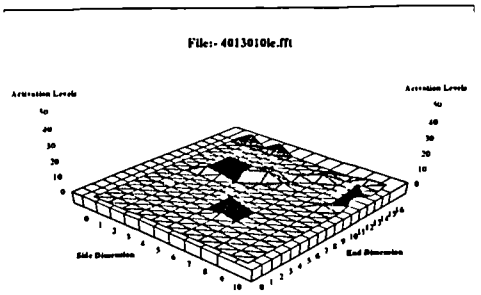


Figure. 5

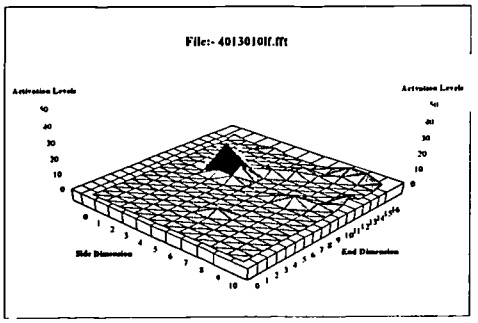


Figure. 6

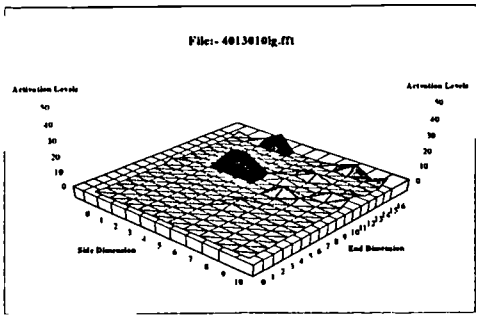


Figure. 7

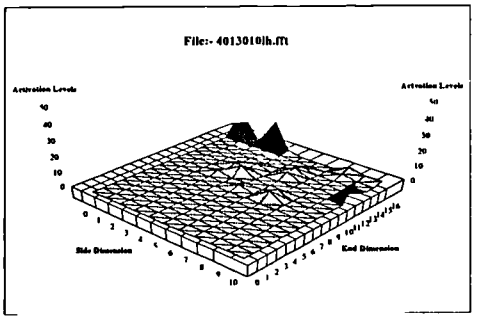


Figure. 8

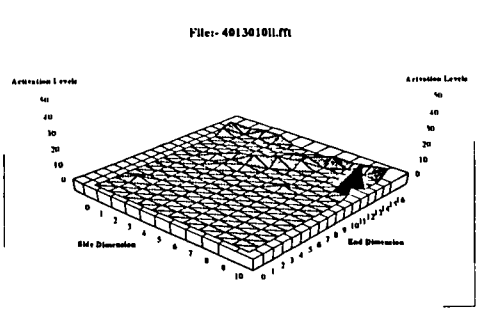


Figure. 9

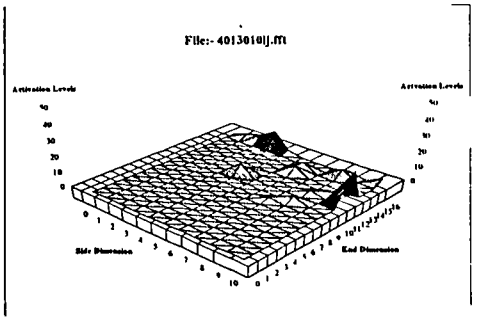


Figure. 10

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 12.5mm/s Ts

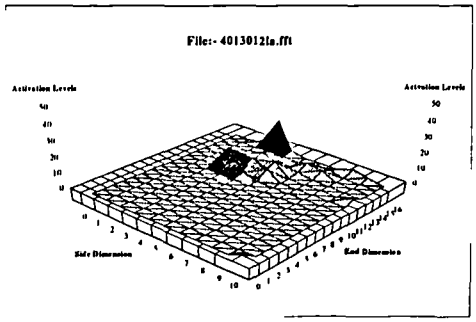


Figure. 1

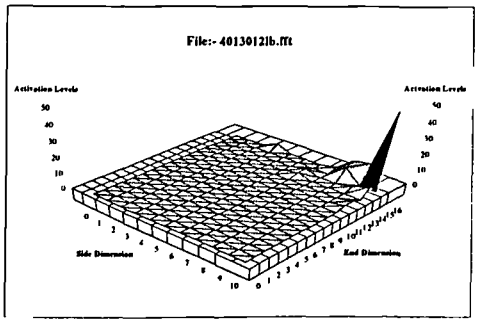


Figure. 2

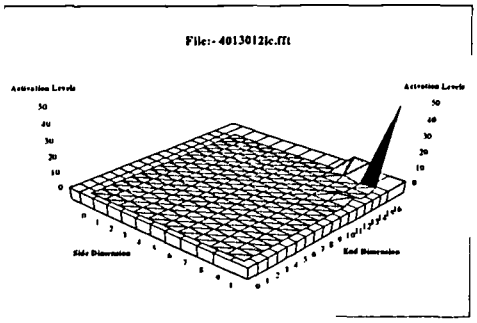


Figure. 3

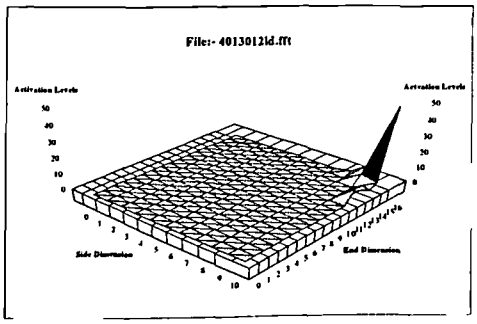


Figure. 4

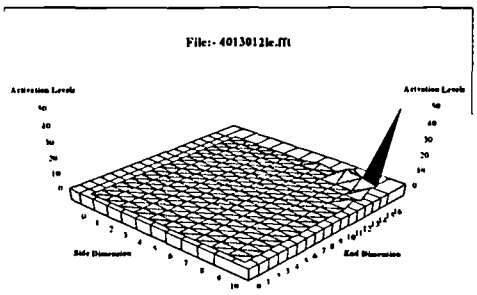


Figure. 5

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 15mm/s Ts

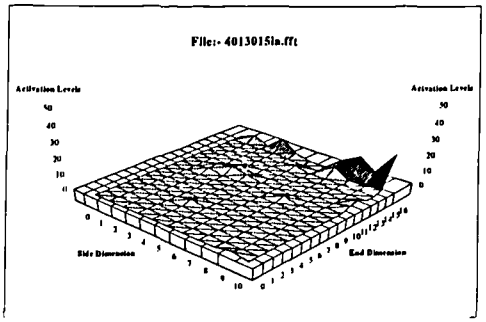


Figure. 1

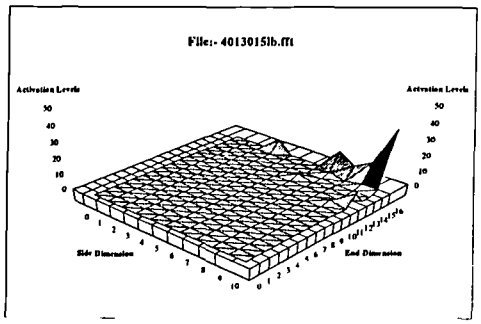


Figure. 2

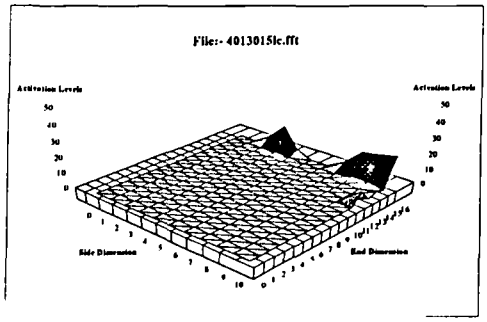


Figure. 3

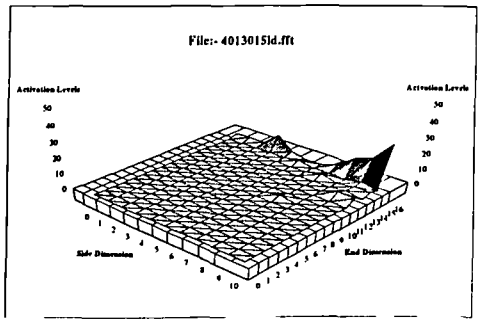


Figure. 4

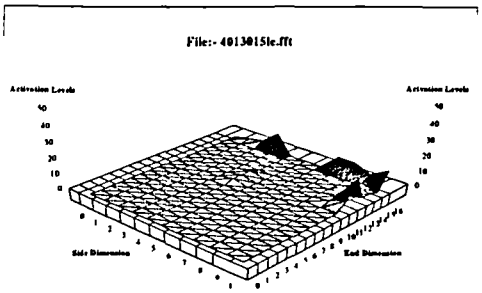


Figure. 5

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 17.5mm/s Ts

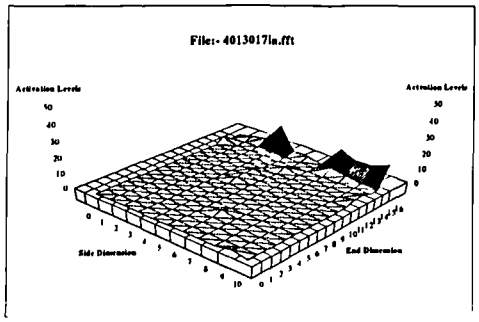


Figure. 1

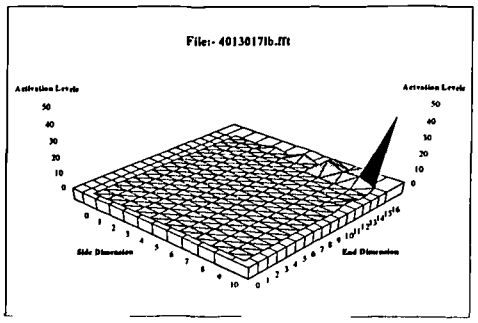


Figure. 2

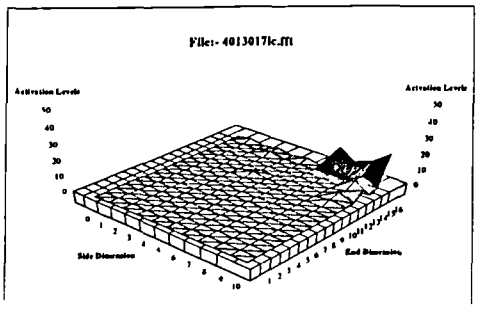


Figure. 3

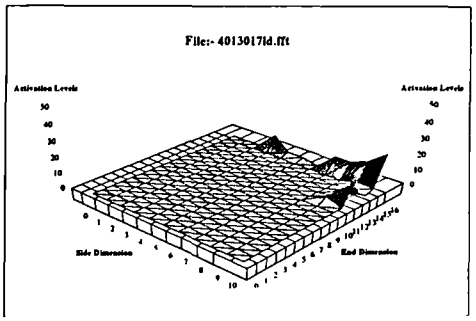


Figure. 4

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 20 mm/s Ts

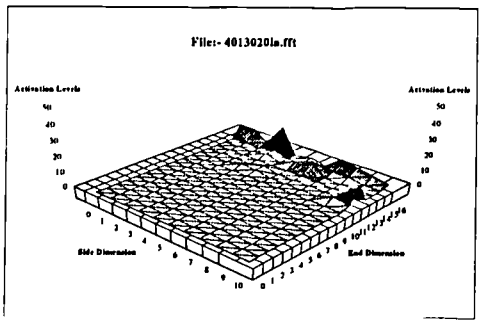


Figure. 1

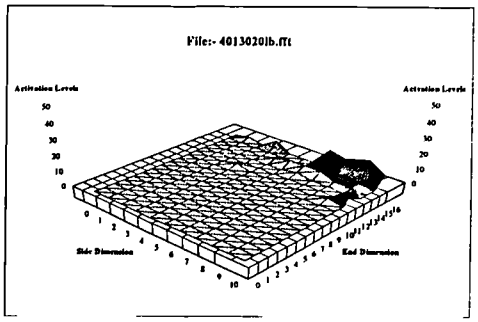


Figure. 2

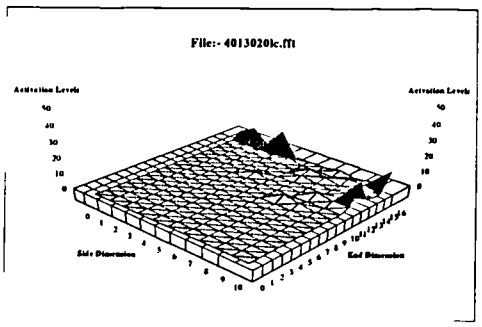


Figure. 3

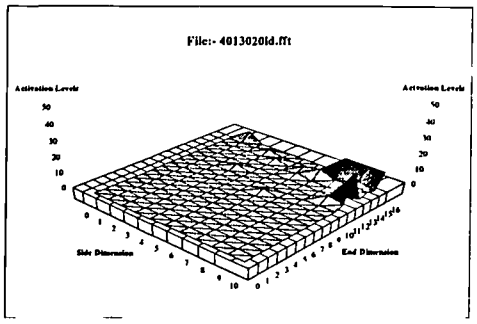


Figure. 4

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 22.5mm/s Ts

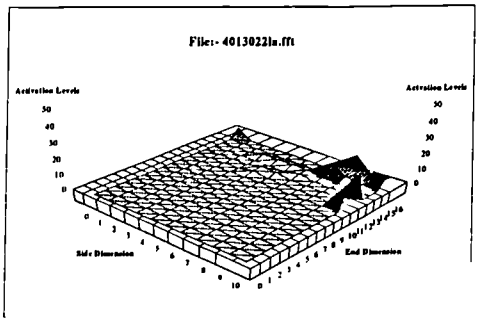


Figure. 1

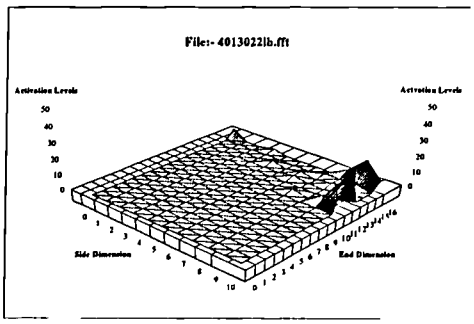


Figure. 2

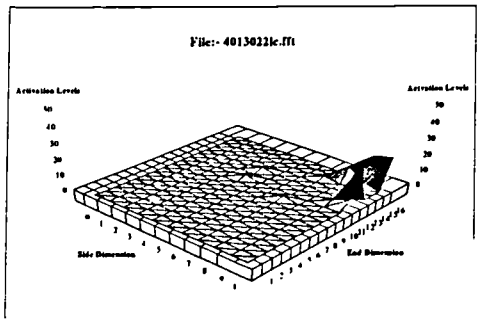


Figure. 3

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 25mm/s Ts

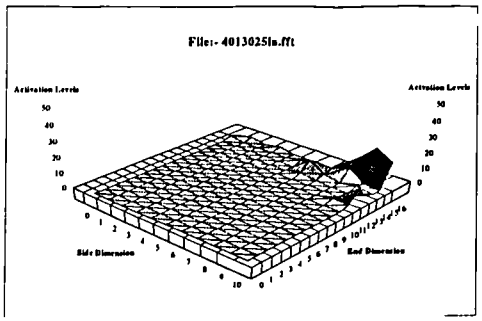


Figure. 1

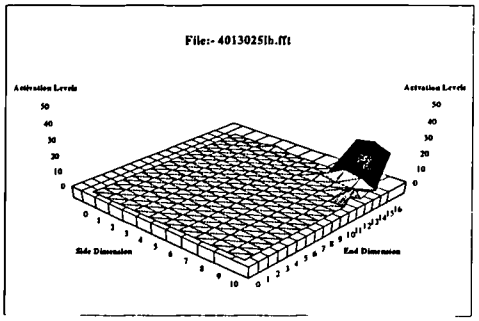


Figure. 2

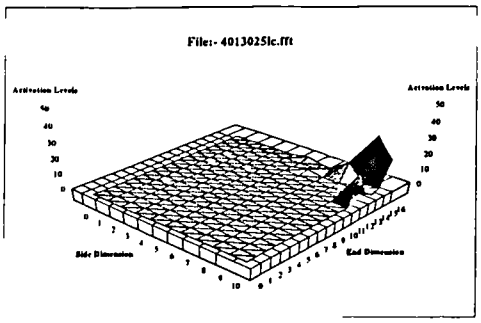


Figure. 3

Self Organising Feature Map Output Activations

(50 Patterns Presented)

Response to 40V, 130 Wfs, 27.5mm/s Ts

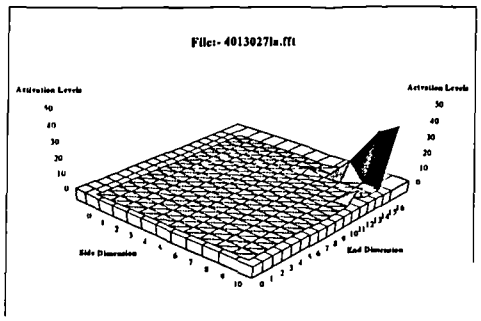


Figure. 1

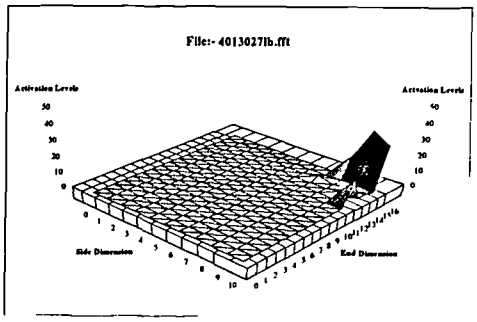


Figure. 2

APPENDIX 8

Published Papers

The International Association of Science and Technology for Development IASTED

" Computer Applications in Industry"

Alexandria, Egypt, May 5 - 7, 1992

pp24 - 27

THE APPLICATION OF NEURAL NETWORKS FOR THE CONTROL OF INDUSTRIAL ARC WELDING.

J.R. McCardle B.Sc., Karen Taylor Burge M.A., B.Sc., Ray R. Stroud M.Tech, Ph.D., Tom Harris B.Sc.

Department of Design, Brunel University, UK.

ABSTRACT

The use of automatic closed loop control is well established in all areas of manufacturing industry. New methods for measuring system variables, data processing and process control are being sought to improve system efficiency.

Skilled welders are able to subconsciously monitor a manual arc welding process by listening to the sound and repositioning the electrode in response to a change in arc noise.

This paper describes the real time monitoring of acoustic emissions from an automated submerged arc welding process and the application of Neural Networks to predict the point of instability of the process variables.

Keywords:- Neural Networks, Signal Processing, Real Time Control, Industrial Process Monitoring.

INTRODUCTION

The Design Department at Brunel University, UK, has aimed its research at the development of "intelligent" products incorporating Artificial Neural Networks (ANNs). One focus of attention has been the development of a fully integrated and hybrid control system for industrial welding processes.

To date ANNs have been successfully implemented to interpret ultrasonic scans of a submerged arc welding process to achieve real-time monitoring of the weld penetration and positional control of the welding head.[1,2,3].

Observations of skilled manual welders has revealed a subconscious tendency to alter the electrode angle and length of arc in response to a variation in the sound emitted from the process. Attempts have been made to analyse these acoustic emissions using conventional and expert system techniques.

Preliminary work carried out at The Cranfield Institute of Technology, UK, utilised an expert system in an attempt to interpret acoustic signals for the on-line control of automated welding equipment. The inability of expert systems to respond correctly to erroneous or novel data yielded discouraging results when attempting to isolate salient features from the erratic acoustic emissions.[4].

ANNs have been successfully implemented in systems which imitate human attributes

including visual and audio recognition [5,6,7,8].

The aim of this research is to propose a feasible ANN oriented system to automatically control an industrial submerged arc welding process by imitating the biological experience of a human operator.

THE WELDING PROCESS

Submerged arc welding is a semi-automatic process which utilises a sacrificial electrode to strike an arc under a cover of granulated flux.(FIG. 1). In an industrial situation the process is used to join steel plates of typical thicknesses between 6mm and 40mm in a "downhand" position. For the purposes of this experimentation, plates will be standardised at 25mm.

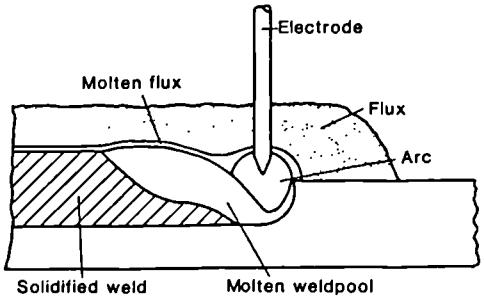


Figure 1. Submerged Arc Welding

There are six major parameters whose optimum settings dictate the overall quality of the the weld :-

i) weld plate preparation (fit up)
ii) welding voltage
iii) welding current (determined by the rate of feed of the sacrificialelectrode)
iv) position of the electrode (seam tracking)
v) rate of welding head travel along the seam
vi) continuity in feed of the granular flux.

A deviation from the optimum setting of any of the parameters can result in such welding faults as porosity, lack of fusion, insufficient root penetrations and undercuts resulting in stress raisers.

Although the nature of raw acoustic emissions is such that data acquired could contain usable information concerning parameters ii) to vi) it is considered that the prime directive of this research is the control of the welding voltage. This is to complement the existing ultrasonic weld penetration system which controls the welding current. The result will be a combined system that maintains an optimum balance of these parameters to obtain weld stability.

Secondary attention will, however, be given to the remaining parameters to serve a diagnostic function.

PROPOSED EXPERIMENTATION

Audible acoustic information has been successfully employed in diagnostic systems for pulsed laser welders [9] and internal combustion engines [10]. Acquisition of raw acoustic data will be achieved by use of an omni-directional electret condenser microphone (ECM). Such transducers exhibit a large bandwidth over the audible frequency range as well as being uninfluenced by induced noise caused by the magnetic flux emitted from the area of the arc.

The microphone and pre-amplifier will be appropriately noise shielded and mounted in the vicinity of the welding head and connected remotely to a series of bandpass filters. The filter set will be constructed to isolate the three main spectra of infra, audible and ultra sound (FIG.2). Although priority is given to the audible range consideration will be given to the possible influences of frequencies present in the infra and ultra sound ranges. The analysis of the ultra sound will be limited by the ADC sampling rate which for this case will be 1MHz providing a practical frequency limitation of circa 200kHz.

As human hearing deteriorates towards the extremes of the audible range this spectrum if further filtered by eight passbands in a Normal Distribution.

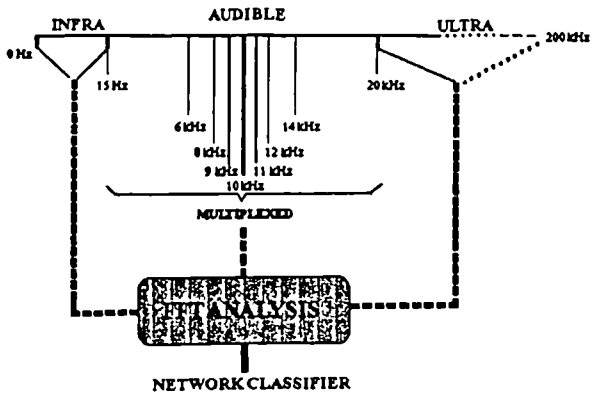


Figure.2 Audio Frequency Spectrum

The final array of passbands are multiplexed for optimum flexibility, to enable each bandpass to be isolated or used as stopbands.

Each bandpass is then subjected to an FFT analysis to isolate salient features within the frequency domain, when subjected to intentionally unstable inputs. Future developments will involve the filtration and isolation of such usable frequencies by hardware.

Isolated signals will then be digitised at a frequency of 1MHz and a resolution of 8 bits and downloaded to a PC for analysis by the software simulated ANN. The output of the analysis will interface with the controlling software of the welding parameters

The proposed experimental set up is shown in Figure 3.

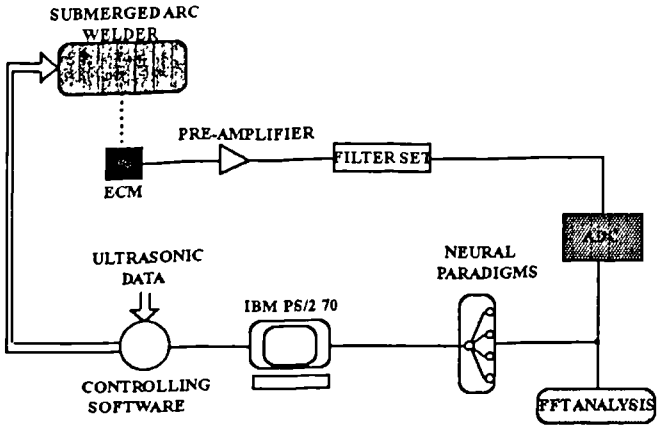
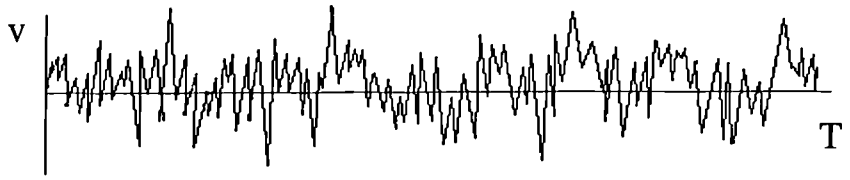


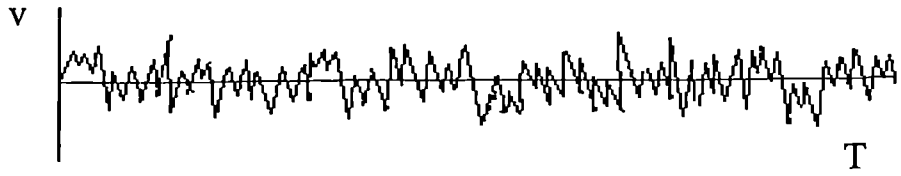
Figure 3. Experimental Set-up

APPLICATION OF NEURAL NETWORK PARADIGMS

Preliminary analysis of raw acoustic emissions yield signals typical of Figure 4. An amplitude degradation is evident, however, for use in real time control a signal processing system is required to isolate the transition band, detect its onset and consequently determine the point of instability of the weld parameter. This would involve the fast processing of noise affected data as well as the recognition of transition band characteristics associated with each weld parameter.



a) Optimum weld emission



b) Flux blockage signal

Figure 4. Acoustic Emissions

ANNs have been successfully utilised in the high speed signal processing of noisy or erratic data where conventional signal processing has failed [1,2,3,10]. Advantages of networks are enhanced by their ability to generalise and have been used in systems requiring predictive assessment of data. [11]

The general trend in ANN application has been towards the development of hybrid and multi-architectural systems[12,13]. In this way characteristics of certain network topologies are used in parallel or series with each other and/or expert systems.

It has been envisaged that for the application to this research three neural networks could be employed

i) Back PropagationNetworks[14,15]
ii) Kohonen self organising feature maps [14]
iii) Weightless or Logical Neural networks [16]

Back error propagation has been successful in bandpass functions and the filtering of noisy data especially where ambient noise creates a problem [5,15].

Pattern recognition problems are relieved by the extraction of salient features on which a pattern classifying network can base its considerations. Back error propagation can identify usable features automatically. This, therefore, can be considered the pre-processing network and when optimally trained could remove the necessity of the more time consuming FFT analysis.

The Kohonen paradigm, when primed with a set of prominent features, can identify the organisational relationships and map the similarities of the input patterns. It is anticipated that this network can provide a complex pattern recognition system for the identification of features associated with signal transition bands. The signal transition is indicative of single or multiple weld parameter fluctuation and consequently of weld instability.

The Logical Neural Network (LNN) was a system jointly devised by Brunel University and Imperial College, London. It utilises conventional RAM technology to store responses to "learned" patterns. If a summing device is used to process the output responses of a number of RAMs it is possible to discriminate between novel and learned patterns. Manipulation of the neural threshold functions will ensure that similar input patterns provide similar responses and hence exhibit properties of generalisation. This technique is used in a system known as WISARD which operates on real time video data to distinguish between facial images [8].

It is envisaged that the LNN could discriminate between successive input patterns from the Kohonen network and finally identify which weld parameter is potentially unstable.

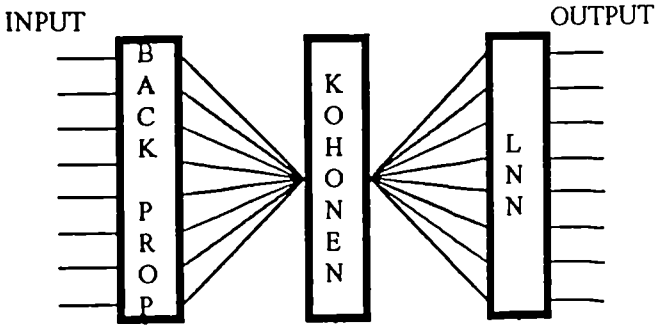


Figure 5. Neural System

The proposed neural system is summarised in Figure 5.

The ANNs are to be developed using either commercial software tools or custom compiled programs in 'C' and Prospero Pascal. The conventional RAM techniques utilised by LNNs provide ready adaptation to hardware.

CONCLUSIONS

ANNs have been successfully employed to interpret chaotic ultrasonic signals in real time and provide on line weld control [1,2,3].

Complex problems require the combination of knowledge based and neural computing techniques to reach an optimum solution.

It is feasible that a hybrid, multi-architectural system can interpret acoustic data in real time to predict the onset of weld parameter instability.

Research and development in the area of weld automation continues with the aim of providing a fully integrated welding system.

REFERENCES

[1] Stroud R.R., Harris T.J., Taylor Burge K.L., 'Neural Networks in Automated Weld Control', TWI, Gateshead, UK, 1991. Paper 22.

[2] Harris T.J., Stroud R.R. & Taylor Burge K.L., 'Neural Networks in a Weld Control System', AINN'90, Zurich, June 1990.

[3] Stroud R.R., Harris T.J.& Taylor Burge K.L., 'Applications of Neural Networks in Control Technology', Proc 1st ICANN, Helsinki, Finland, June 1991. Vol 2, p1703.

[4] BRITE EuRam project-BREU 0415 BE-4006-90

[5] Mead C., 'Analog VLSI And Neural Systems', Addison-Wesley, Reading MA, 1989

[6] Torkkola K., *et al*, 'Status Report OF The Finnish Phonetic Typewriter Project', ICANN-91, Espoo, Finland, June 1991. Vol 1, P771.

[7] Nellis J., Stonham T.J, 'A Fully Integrated Hand-Printed Characture Recognition System Using Artificial Neural Networks', IEE Second International Conference On Artificial Neural Networks, Conference Publication No. 349, Bournemouth, UK, November 1991. p219.

[8] Fukushima K., Imagawa T., 'Recognition And Segmentation Of Connected Characters In Cursive Handwriting With Selective Attention', ICANN-91, Espoo, Finland, June1991. Vol 1, p 105.

[9] Meyendorf N. *et al*, 'Monitoring of Pulsed Laser Welding by Acoustic Emissions', COMADEM '91, Southampton, July 1991. p515.

[10] Vu V.V, *et al*, 'Time Encoded Matrices as Input Data to Artificial Neural Networks for Condition Monitoring Applications', COMADEM '91, Southampton, July 1991.

p31.

[11] Tanaka T. & Endo H. *et al* 'Trouble forecasting by multi-neural network on continuous casting process of steel production', Proc 1st ICANN, Espoo, Finland, June 1991. Vol 1 p835.

[12] Fogelman Soulie, F., 'Neural Network Architectures and Algorithms: A Perspective', ICANN-91, Espoo, Finland, June 1991. Vol 1, P605.

[13] Picton P.D., Johnson J.H., Hallam N.J. 'Neural Networks in Safety-Critical Systems', COMADEM '91, Southampton, July 1991. p17.

[14] Dayhoff J, 'Neural Network Architectures An Introduction', Van Nostrand Reinhold, 1990.

[15] Hecht-Nielson R., ' Neurocomputing', Addison-Wesley Publishing Company, 1991.

[16] Aleksander I., 'Connectionism Or Weightless Neurocomputing?', ICANN-91, Espoo, Finland, June 1991. Vol 2, p991.

The Welding Institute - TWI

4th International Conference

" Computer Technology in Welding "

Cambridge, UK, June 3 - 4, 1992

Paper 35

THE USE OF NEURAL NETWORKS TO CHARACTERIZE PROBLEMATIC ARC SOUNDS

K.L. Taylor Burge	B.Sc, MA	Brunel University
T.J. Harris	B.Sc, GradWeldI	Brunel University
R.R. Stroud	MTech, PhD	Brunel University
J.R. McCardle	B.Sc	Brunel University

ABSTRACT

Automation of electric arc welding has been at the centre of considerable debate and the subject of much research for several decades. One conclusion drawn from all this effort is that there seems to be no single system that can monitor all of the variables and subsequently, fully control any welding process. To date there has been considerable success in the development of seam tracking systems employing various sensing techniques, good progress has been made in the area of penetration measurement and worthwhile use has been made of the integration of expert systems and modelling software within these control domains.

Skilled welders develop their own monitoring and control systems and it has been observed that part of this expertise is the ability to listen subconsciously to the sound of the arc and to alter the electrode position in response to an adverse change in arc noise.

Attempts have been made to analyse these sounds using both conventional techniques and more recently expert systems, neither have delivered any usable information. This paper describes a new approach involving the use of neural networks in the identification of sounds which indicate that the welding system is drifting out of control.

INTRODUCTION

Artificial Neural Networks (ANNs) offer potential as an alternative to standard computer techniques in control technology and have attracted a widening interest in their development and application. Although the conception of ANN theory predates that of the modern digital computer, the commercial success of Von Neumann systems and the utilization of Boolean logic has over-shadowed their development.

Due to the proliferation of the digital computer the use of ANNs in 'real' applications tend to be in the form of software neural simulators. The commercial availability of dedicated neural hardware is very limited. Consequently, the techniques employed for the proposed research would involve the application of commercial simulators as well as custom compiled software.

Research within the Department of Design at Brunel University has aimed at the development of intelligent control systems incorporating ANNs. One focus of attention has been industrial welding processes. To date ANNs have been successfully implemented to process ultrasonic scans of a submerged arc welding process, Fig.1, to achieve real time kinetic control of the welding head as well as monitoring weld penetration.(1,2,3)

It has been observed that skilled manual arc welders subconsciously change the angle of electrode and arc length in response to a variation in the sound produced from the process. Much evidence exists for the successful application of ANNs in systems which imitate human attributes including the development of the "Silicon Ear" and "Retina" by Carver Mead (12), Kohonen's "Phonetic Typewriter" (13) and hand written character recognition (14,15). The goal of this work is to propose a feasible ANN oriented system to automatically control an arc welding process and thereby imitate the biological expertise of a human operator.

THE WELD VARIABLES

The creation of an ideal weld is dependent upon the optimum settings for :-

i) weld plate preparation (fit up)
ii) welding voltage,
iii) welding current (determined by the rate of feed of the sacrificial electrode)
iv) position of the electrode

and

v) speed of travel of the welding head along the seam.

A sixth parameter which can drastically affect the quality of the weld is the feed of granular flux. This is generally gravity fed and hence not directly controllable. However a system which detects the onset of a blockage as a diagnostic feature is desirable.

Following the development of a positional control system for submerged arc welding by means of ultrasonics (1,2,3), the preliminary aim of this research is the control of parameters ii), iii) and v).

In an industrial scenario the submerged arc welder is used to join plates of between 6mm and 40mm. For initial research purposes under laboratory conditions plates of 25mm would be used. Optimum settings for each parameter would be preset by a controlling PC to obtain ideal weld penetration.

To compliment the existing ultrasonic weld penetration monitoring system (dedicated to the closed loop control of the welding current), it is considered that the prime directive of this research would be the control of the welding voltage to maintain an optimum balance and hence weld stability. However the nature of a raw acoustic emission is such that data acquired will contain information concerning other parameters which can be utilised within the diagnostic arena.

PROPOSED METHODOLOGY

Acoustic emissions, within the audible range, have been successfully employed to extract usable diagnostic information from systems such as pulsed laser welders (4) and IC engines (5). The acquisition of acoustic data creates little problem providing the transducer and associated amplifier is tailored to suit the application.

It is proposed that for this work an omni directional Electret Condenser Microphone (ECM) and pre-amplifier, are adequately noise shielded and resiliently mounted in the vicinity of the welding head. The ECM exhibits a large bandwidth with a uniform frequency response within the audible range of 15Hz to 20kHz as well as rejecting any induced noise from the 'Pinch Effect'(11) of the arc.

The signals could then be passed through a bandpass filter set to isolate the audible range from infra and ultra sound. An interim FFT analysis at this point would emphasise usable frequency ranges in response to intentionally unstable inputs to the welding system. Future development in this area will involve the hardware filtering and isolation of usable frequencies within the spectrum.

The signal could then be digitised at a frequency of 1MHz and a resolution of 8 bits and downloaded to a PC for analysis by the software simulated ANN. The outcome of the analysis will then interface the controlling software of the welding parameters.

The equipment proposed for the experimentation, Fig.4, comprises

i) SAF Devimatic submerged arc welding set
ii) IBM PS/2 70
iii) FFT analyzer
iv) Custom built transducer arrangement, filter sets and interfacing system.

Computer programs are to be compiled in 'C' and Prospero Pascal.

Neural Network Paradigms

Neural Networks are modelled on the architecture of the biological brain. The network is constructed of discrete neurons each providing an output to a given stimulus dictated by a mathematical function, Fig.5. The collective response of a network is dependent upon the topology of synaptic connections. The topology is subject to debate as various architectures exhibit differing characteristics of cognition and recognition. A simple example is shown in Fig.6. This illustrates a multi-layer fully connected system. The input layer where data is presented to the network, an output layer to communicate a response and a "hidden layer" which increases the processing ability of the system.

ANNs have been proven successful in high speed signal processing, especially in noisy or erratic systems where conventional signal processing failed.(1,2,3,5). Advantages of networks are further enhanced by their apparent ability to "generalise", which is to respond correctly to novel, erroneous or even incomplete data; an inability of expert systems.

The trend in ANN applications has steered towards the development of hybrid systems (6,7) where the characteristics of certain network topologies are used in conjunction with others or in parallel with expert systems. Complex problems require the combination of knowledge based and neural computing techniques to reach an optimum solution.

For the application to this work it was envisaged that two neural network architectures could be employed:-

i) Kohonen self organizing feature maps (8)
ii) Weightless or Logical Neural Networks (9)

The Kohonen paradigm possesses the inherent characteristic to self organize its topology by a method of competitive, "unsupervised learning". That is, the final pattern of synaptic connections is optimized by a self iterative process known as a "learning algorithm". The result is a system that can identify the organizational relationships between input patterns and map the similarities into "closeness" groups. The advantage over standard pattern recognition is its speed, because of the parallel nature of the neural process and the provision of a graphical representation of pattern relationships. This, therefore, can provide a complex pattern recognition system.(10)

The Logical Neural Network was a system devised jointly by Brunel University and Imperial College, London. It utilizes the contents of addresses within a random access memory (RAM) in its learning algorithm rather than the more usual iterative update of synaptic weights adopted by most paradigms. Hence the term a "weightless" network. If a summing device is used to process the responses from a number of RAMs a degree of similarity or "discrimination" can be established between novel input patterns and previously stored "learned" patterns. With manipulation of the neural threshold functions within the "discriminator" similar input patterns will produce similar responses and therefore exhibit properties of generalization. This technique was used in the WISARD system. Operating on real time video data it is able to distinguish between facial images and is consequently being applied to facial recognition purposes in security systems(8).

With the application of a Kohonen paradigm as a complex pattern "classifier" or feature extractor and a Logical Neural Network as a pattern "discriminator" an optimum solution for the real time processing of compound acoustic signals is proposed.

An ANN can be developed using either commercial software tools or custom compiled programs. Various topologies have proven successful in specific applications however the architecture has to be tailored to suit the application if an optimum solution is to be reached. Too many hidden layers or artificial neurons (sometimes known as Nodes) will lead to lengthy training periods and ultimately slower operating speeds.

CONCLUSIONS

It has been previously proven that ANNs have the ability to interpret chaotic ultrasonic signals in real time and provide on line weld control.(1,2,3)

With the application of hybrid systems it is feasible that real time acoustic data, both audible and inaudible, may be used to predict the onset of weld instability.

Neural Networks provide a key to a higher degree of process automation.

Research and development in the area of weld process and kinetic control continues with

the aim of providing a fully integrated and hybrid automated welding system.

REFERENCES

(1) Stroud R.R., Harris T.J., Taylor Burge K.L., 'Neural Networks in Automated Weld Control', TWI, Gateshead, 1991. Paper 22.

(2) Harris T.J., Stroud R.R. & Taylor Burge K.L., 'Neural Networks in a Weld Control System', AINN'90, Zurich, June 1990.

(3) Stroud R.R., Harris T.J.& Taylor Burge K.L., 'Applications of Neural Networks in Control Technology', Proc 1st ICANN, Helsinki, Finland, June 1991. Vol 2, 1703.

(4) Meyendorf N. *et al*, 'Monitoring of Pulsed Laser Welding by Acoustic Emissions', COMADEM '91, Southampton, July 1991. P515.

(5) Vu V.V, *et al*, 'Time Encoded Matrices as Input Data to Artificial Neural Networks for Condition Monitoring Applications', COMADEM '91, Southampton, July 1991. P31.

(6) Fogelman Soulie, F., 'Neural Network Architectures and Algorithms: A Perspective', ICANN-91, Espoo, Finland, June 1991. Vol1, P605.

(7) Picton P.D., Johnson J.H., Hallam N.J. 'Neural Networks in Safety-Critical Systems', COMADEM '91, Southampton, July 1991. P17.

(8) Dayhoff J, 'Neural Network Architectures An Introduction', Van Nostrand Reinhold, 1990.

(9) Aleksander I., 'Connectionism Or Weightless Neurocomputing?', ICANN-91, Espoo, Finland, June 1991. Vol 2, P991.

(10) Yoh-Han Pao., 'Adaptive Pattern Recognition And Neural Networks', Addison-Wesley, 1989. P182.

(11) International Institute Of Welding, 'The Physics Of Welding', Pergamon Press, 1984. Ch3

(12) Mead C., 'Analog VLSI And Neural Systems', Addison-Wesley, Reading MA, 1989.

(13) Torkkola K., *et al*, 'Status Report OF The Finnish Phonetic Typewriter Project', ICANN-91, Espoo, Finland, June 1991. Vol 1, P771.

(14) Nellis J., Stonham T.J, 'A Fully Integrated Hand-Printed Characture Recognition System Using Artificial Neural Networks', IEE Second International Conference On Artificial Neural Networks, Conference Publication No. 349, Bournmouth, November 1991. P219.

(15) Fukushima K., Imagawa T., 'Recognition And Segmentation Of Connected Characters In Cursive Handwriting With Selective Attention', ICANN-91, Espoo, Finland, June1991. Vol 1, P 105.

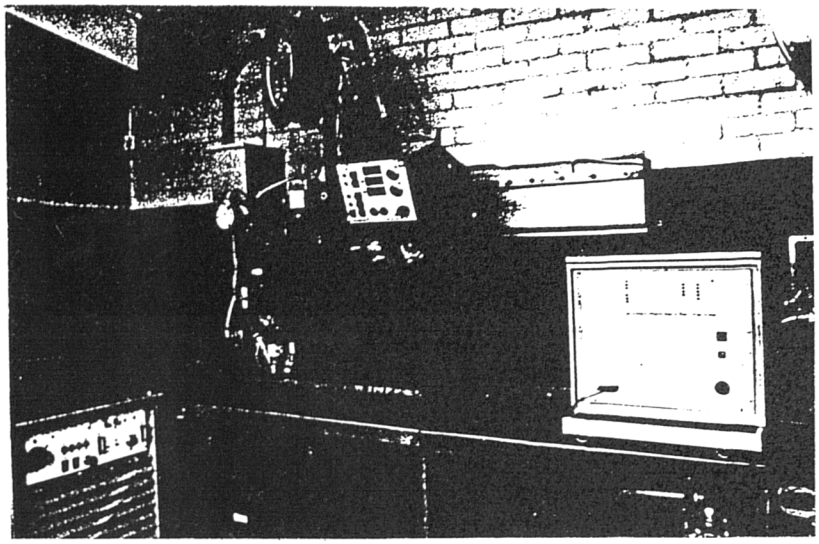


FIGURE 1.
SUBMERGED ARC WELDER

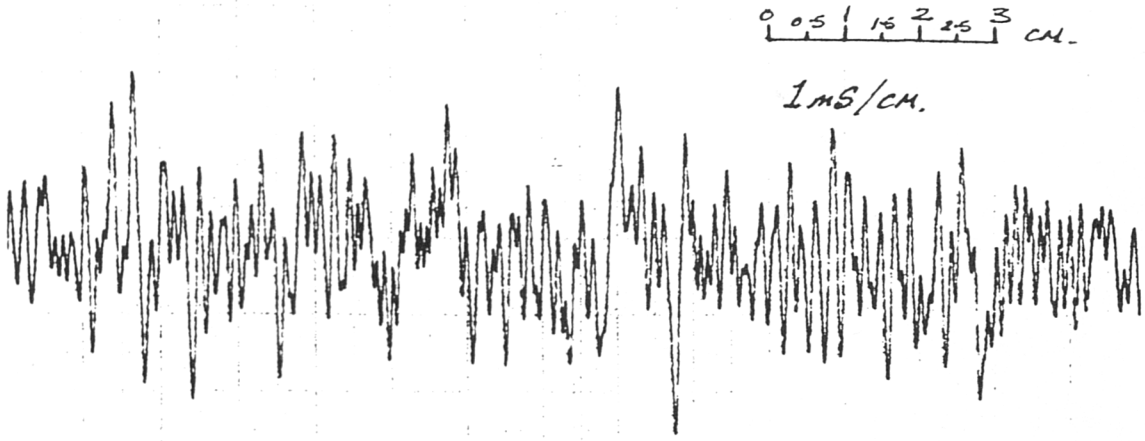


FIGURE 2.
ACOUSTIC EMISSION OF
AN OPTIMUM WELD

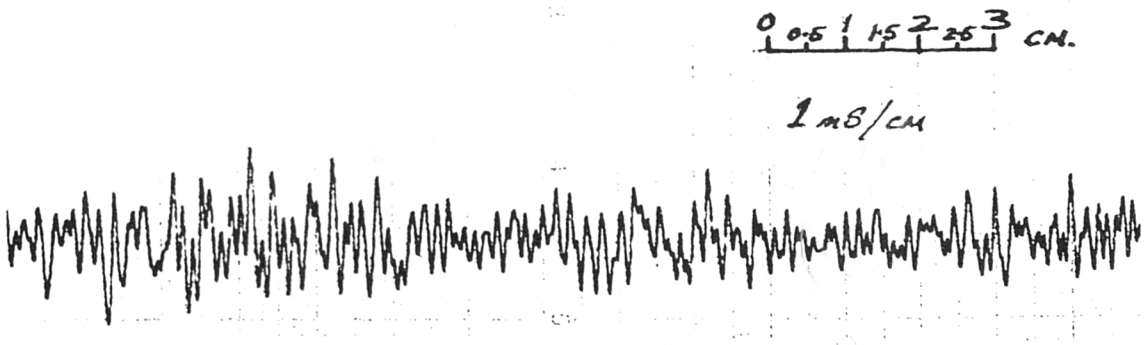


FIGURE 3.
ERRONEOUS EMISSION DUE TO
A FLUX BLOCKAGE

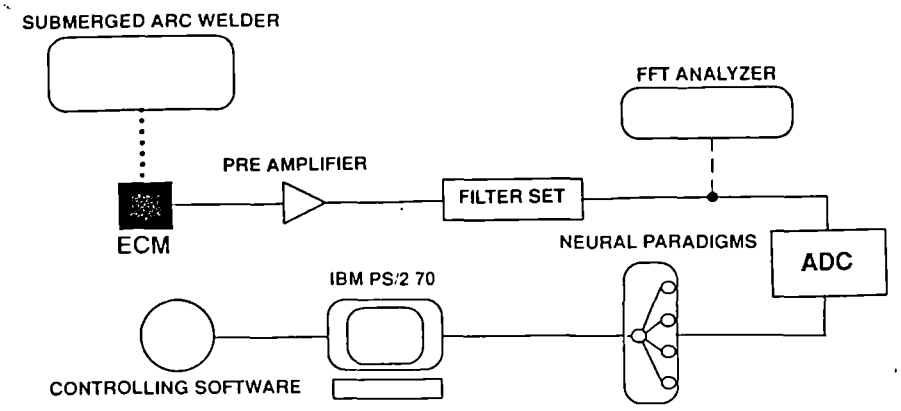


FIGURE 4.
EXPERIMENTAL SETUP

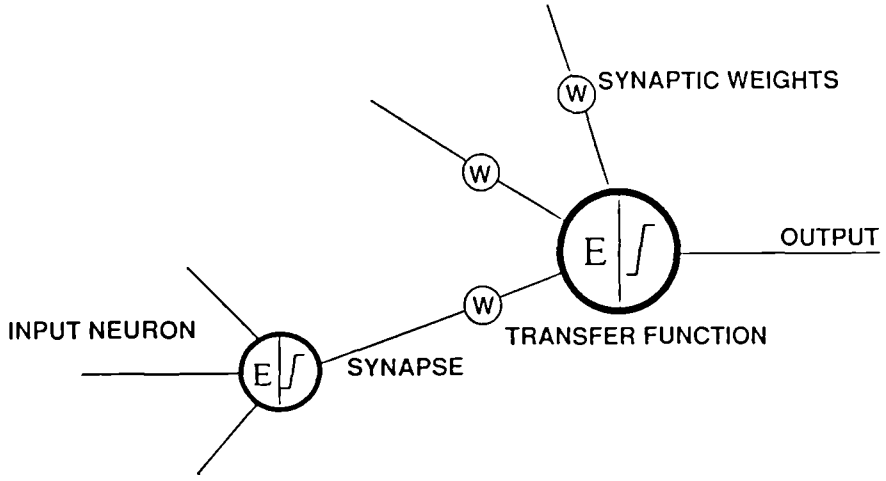


FIGURE 5.
ELEMENTS OF NEURAL CONNECTION

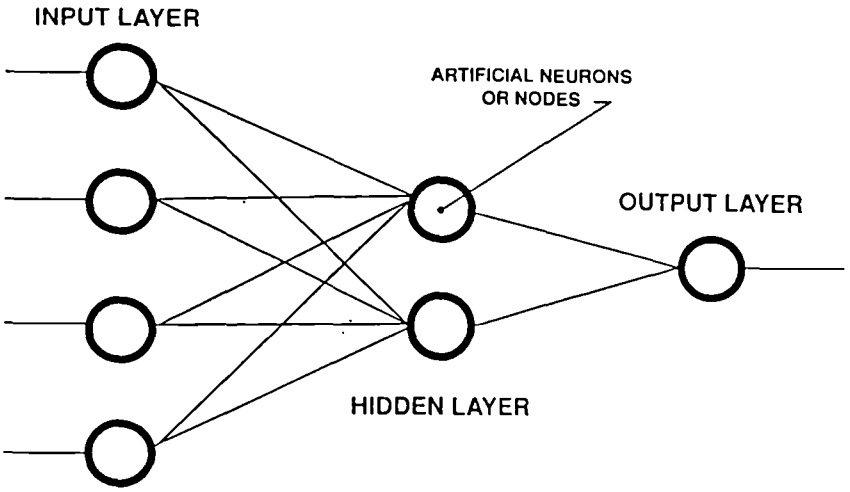


FIGURE 6.
SIMPLE MULTI-LAYER ARCHITECTURE

Condition Monitoring and Diagnostics Engineering Management - COMADEM

4th International Conference

Senlis, France, July 15 - 17, 1992

pp 94 - 99

THE MANAGEMENT OF INDUSTRIAL ARC WELDING BY NEURAL NETWORKS

J.R. McCARDLE., K.L. TAYLOR BURGE, R.R. STROUD, T.J.HARRIS
Department of Design, Brunel University, Egham, UK.

Abstract
New methods of monitoring industrial process variables are constantly being sought with the aim to improve control efficiency.

It has been observed that skilled welders subconsciously adapt their manual arc welding technique in response to a variation in the sound produced from the process.

This paper proposes an approach to the control of an automated submerged arc welding process using:-

1. Real time monitoring of acoustic emissions
2. The application of neural networks to predict the point of instability of the process variables.

Keywords: Process Control, Signal Processing, Neural Networks, Real Time Monitoring.

1 Introduction

Artificial Neural Networks (ANNs) are attracting a widening interest in their development and applications as an alternative to standard digital control and expert system techniques. Complex signal analysis with inherent noise characteristics have proven difficult, if not impossible, to process with expert systems often causing data explosions.(BRITE EuRam project).

The Department of Design at Brunel University, UK, has directed its research at the development of 'intelligent ' products and control systems incorporating ANNs. One focus of attention has been the development of a fully integrated , hybrid control system for industrial automated weld processes.

An approach successfully adopted has been the real time interpretation of ultrasonic scans of a submerged arc welding process. An ultrasonic transiever optimally positioned at the welding head can monitor the weld penetration and position of the weld pool in relation to the prepared seam (seam tracking). The ultrasonic data is processed by a weightless or Logical Neural Network (LNN) to achieve on line control.(Stroud R.R, *et al*, 1990-91, Harris T.J, *et al*, 1990).

Observations of skilled manual welders has shown a subconscious tendency to change the angle of the electrode and length of arc by listening to adverse fluctuations in process noise, thereby making intrinsic decisions based on a biological monitoring system. Attempts have been made to analyse these acoustic emissions by use of conventional signal processing and expert system techniques. Preliminary research undertaken at the Cranfield Institute of Technology, UK, utilised digital signal processing and expert systems in an attempt to interpret audible acoustic data for on line control. The inability

of these systems to respond correctly to erroneous, novel or incomplete data yielded negative results when faced with identifying salient features within the erratic acoustic emissions.(BRITE EuRam project).

Evidence exists for the successful implementation of ANNs systems, both software simulated and hardware modelled, to imitate human attributes. The 'Silicon Ear' and 'Retina' (Mead C, 1989), the 'Phonetic Typewiter' (Torkkola K. & Kohonen T, *et al*,1991) and the advancements in hand written character recognition illustrates the success in non discrete data processing (Nellis J, *et al,* 1991.).

The aim of this research is the development of a feasible ANN oriented system which isolates features relating to weld parameter fluctuation within acoustic data in real time. It is anticipated that the proposed method will complement the existing ultrasonic monitor to advance the development of a fully integrated on line weld control system.

2 The Welding Process

Submerged arc welding (SAW) is a semi-automated process which uses a sacrificial electrode to strike an arc under a cover of gravity fed granulated flux. In an industrial situation it is used to join structural steel plates between 6mm and 40mm in thickness in a 'downhand' position (FIG. 1). Existing systems utilise software data tables to set optimum parameter values prior to the commencement of the weld. Certain weld parameters can change steadily during the process.

There are six major variables whose optimum settings dictate the final quality of the weld, these are :

welding voltage
welding current (determined by the rate of feed of the sacrificial electrode)
position of the electrode (seam tracking)
speed of the welding head along the seam
continuity in feed of the granular flux
weld plate preparation (fit up)

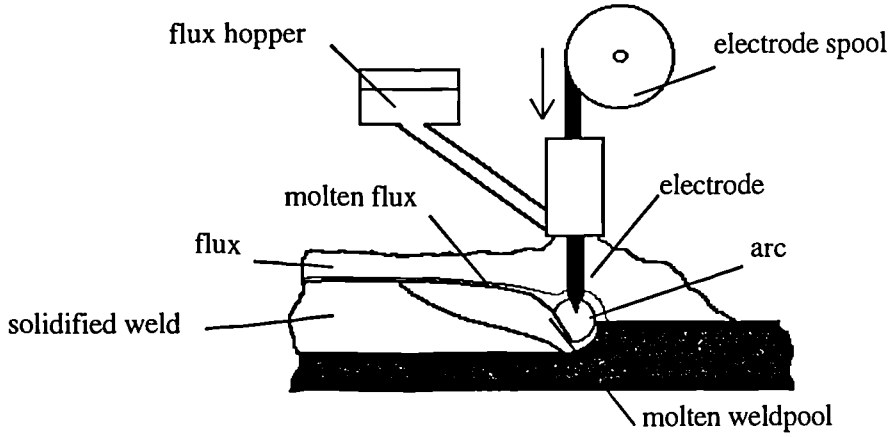


Fig. 1. Submerged arc welding process

The main aim of this research is a system capable of the real time monitoring of the welding voltage to operate in conjunction with the existing ultrasonic weld penetration and seam tracking equipment. This would render a system capable of maintaining an optimum balance between the weld voltage and current parameters and hence achieve weld stability.

The nature of raw acoustic data however, is such that emissions contain information

concerning other parameters which can be utilised for diagnostic purposes and investigation will be given to the identification of each parameters acoustic signature.

2 Proposed experimentation

Audible acoustic emissions have been successfully employed in machinery diagnostic systems such as pused laser welding and internal combustion engines (Vu V.V.*et al* 1991). The proliferation of acoustic transducers renders the collection of data relatively simple. Preliminary experimentation has highlighted the effective use of Electret Condenser Microphones (ECMs). Such transducers exhibit a large bandwidth over the audible frequency range as well as being uninfluenced by induced noise created by the magnetic flux emitted from the arc area.

The ECM and associated pre-amplifier, suitably noise shielded, will be mounted in the vicinity of the weld head. Signals collected will be transmitted to a series of banpass filters to isolate the three spectra of infrasound, 0Hz - 15Hz, audible sound, 15Hz - 20kHz and ultrasound, >20kHz as illustarated in Figure 2. Although the main consideration is given to the audible frequency range, investigations will include the possible influences of infra and ultrasound frequencies. The upper limit of ultrasound will be dictated by the sampling rate of the ADC of 2MHz providing a practical frequency limitation of circa 800kHz.

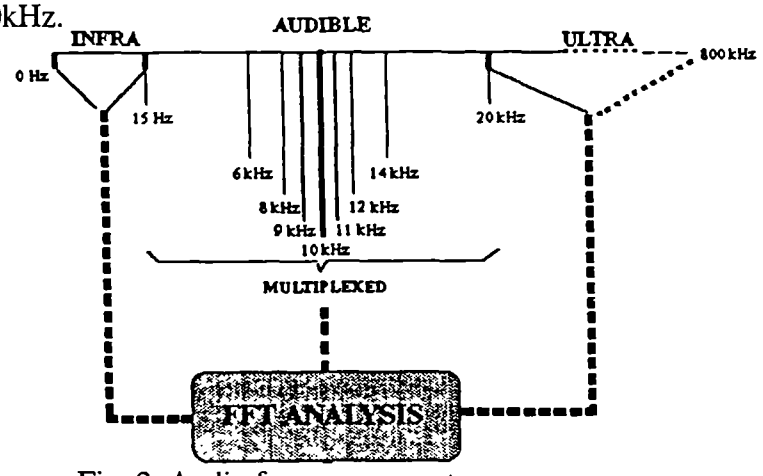


Fig. 2. Audio frequency spectrum

The deterioration of human hearing towards the extremes of the audible spectrum indicates that manual welders possibly extract information from the central range. Therefore, this sectrum is further filtered by eight bands in a Normal Distribution.

The final set of bands are multiplexed via an analogue switch array to enable each band to be utilised as a stop or passband to maintain experimental flexibility.

Isolated bandpasses are to be subjected to an FFT analysis to extract salient features

within the frequency domain when the system is stimulated by intentionally unstable inputs. Once identified, the frequencies associated with the acoustic signatures of each parameter can be isolated by means of hardware filtration.

Multiplexed signals will then be digitised at a frequency of 2MHz and a resolution of 8 bits and interfaced with software simulated or hardwired ANNs. The output of the neural analysis will dictate the corrections to be made by the controlling software of the welding parameters.

The experimental equipment is summarised in Figure 3.

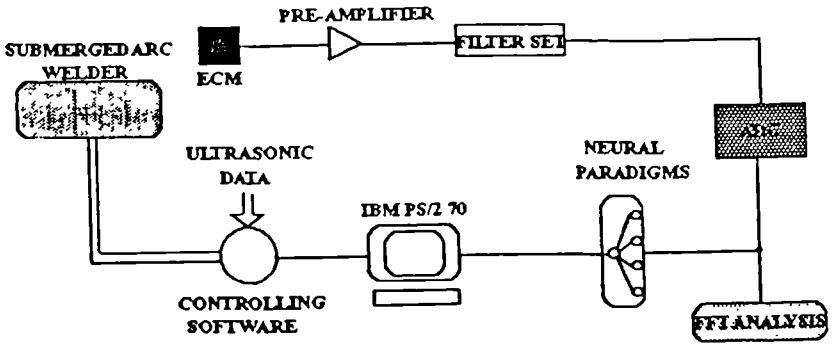


Fig. 3. Experimental schematic

3 Neural Networks

To achieve real time control of the welding process a fast signal processing system is required to recognize the characteristics associated with weld parameter fluctuation from noise polluted data. Evidence exists for the successful implementation of ANNs in systems where the predictive assessment of data is required (Tanaka T & Endo H, *et al*, 1991) as well as generalisng on noise affected signals (Vu V.V, *et al*, 1991).

Preliminary analysis of raw acoustic data provide signals typical to that shown in Figure 4. Figure 4.a. illustrates the emission of an optimum weld at a point where all set parameters remained stable and post weld inspection indicated a good quality weld. Figure 4.b. was recorded at a point where an unplanned flux blockage created weld instability. An amplitude degredation is evident, however, the signal processing system is required to predict the onset of the transition between stable and unstable signals as well as the recognition of the parameter/s causing the error.

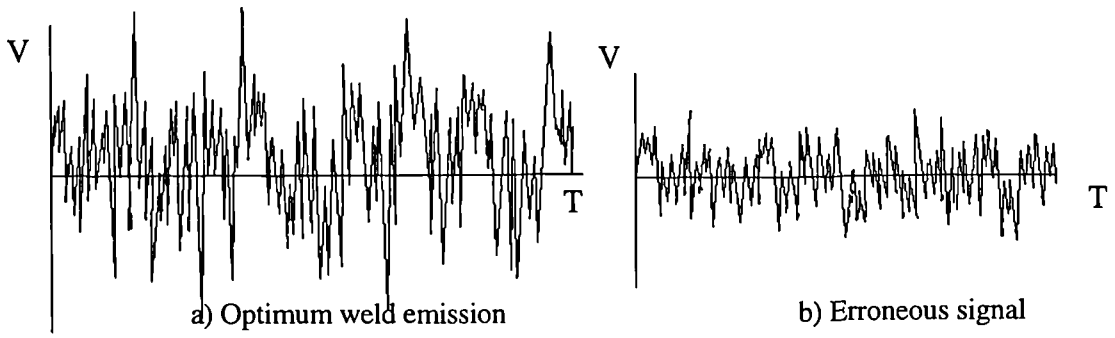


Fig. 4. Acoustic emissions

Complex signal processing and pattern recognition problems require the utilisation of

both neural and knowledge based techniques. The trend in ANN application has steered towards the development of hybrid and multi-architectural systems, exploiting the characteristics of certain topologies in series with others and in series or parallel with expert systems. (Fogelman Soulie F., 1991.).

For the parallel processing of noise inflicted acoustic data it is envisaged that three neural networks could be employed in series:

Back propogation networks (Hecht-Nielson R., 1991, Dayhoff J., 1990).
Kohonen self organising feature maps (Dayhoff J., 1990).
Weightless or logical neural networks (LNNs) (Aleksander I., 1991).

The recognition of usable patterns within complex erratic signals requires the extraction of salient features on which a pattern classifying network can base its considerations. Back propogation networks have proven successful in bandpass functions and signal enhancement. (Hecht-Nielson R., 1991) and have the inherent property of automatically isolating usable features. This technique will form the initial layer and can be considered the pre-processing network. Investigation will be given to the possible use of this technique to replace the FFT analysis and consequent hardware filtering of the acoustic data.

The Kohonen network forms the second layer of the system. This self organising paradigm can identify similarities in input patterns when primed with a set of suitably prominent features. The similarities isolated will group the signals common to unstable weld parameters and separate them from stable parameter inputs. A capability of thus recognizing the features associated with the signal transition bands, indicative of global instability and hence of single or multiple parameter fluctuation.

The logical neural network is a system devised jointly by Brunel University and Imperial College, UK. Utilising conventional RAM technology to store responses to trained patterns, a series of hardwired RAMs can be used to discriminate between novel and learned patterns. Manipulation of the threshold of a triggering function ensure that similar patterns yield similar responses therefore exhibiting generalising characteristics. This system has been proven in the real time application to video data for facial and written character recognition (Nellis J. & Stonham T.J., 1991, Aleksander I., 1991). This system forms the final neural layer to discriminate between successive Kohonen generated features to identify a potentially unstable parameter.

The neural system is summarised in Figure 5.

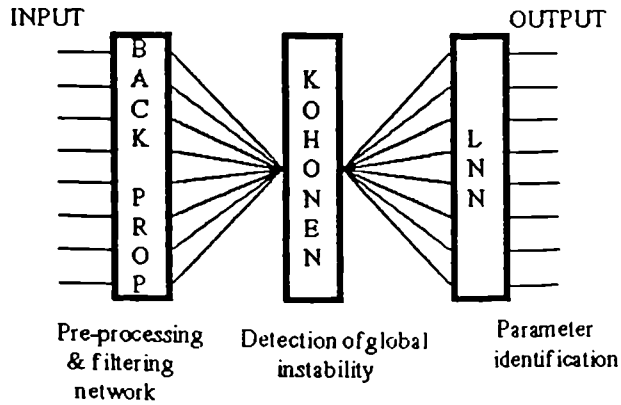


Fig. 5. Hybrid Neural system

4 Conclusion

Successful applications of ANNs to interpret erratic video and ultrasonic data in real time provide the basis for a feasible multi-architectural system to process acoustic data. The solution of complex pattern recognition problems often require the combination of neural network characteristics and expert system techniques.

The speed of parallel processing offered by ANNs provide advancement in the area of automated weld control.

Research and development in the area of weld automation continues with the aim of providing a fully integrated welding system.

5 References

Aleksander I., **Connectionism Or Weightless Neurocomputing?**, ICANN-91, Espoo, Finland, June 1991. Vol 2, p991.

BRITE EuRam project-**BREU 0415 BE-4006-90**

Dayhoff J, **Neural Network Architectures An Introduction**, Van Nostrand Reinhold, 1990.

Harris T.J., Stroud R.R. & Taylor Burge K.L., **Neural Networks in a Weld Control System**, AINN'90, Zurich, June 1990.

Hecht-Nielson R., **Neurocomputing**, Addison-Wesley Publishing Company, 1991.

Mead C., **Analog VLSI And Neural Systems**, Addison-Wesley, Reading MA, 1989.

Nellis J., Stonham T.J, **A Fully Integrated Hand-Printed Characture Recognition System Using Artificial Neural Networks**, IEE Second International Conference On Artificial Neural Networks, Conference Publication No. 349, Bournemouth, UK, November 1991. p219.

Stroud R.R., Harris T.J.& Taylor Burge K.L., **Applications of Neural Networks in Control Technology**, Proc 1st ICANN, Helsinki, Finland, June 1991. Vol 2, p1703.

Tanaka T. & Endo H. *et al* **Trouble forecasting by multi-neural network on continuous casting process of steel production**, Proc 1st ICANN, Espoo, Finland, June 1991. Vol 1 p835.

Torkkola K., *et al*, **Status Report 0f The Finnish Phonetic Typewriter Project**, ICANN-91, Espoo, Finland, June 1991.Vol 1, P771.

Fogelman Soulie, F., **Neural Network Architectures and Algorithms: A Perspective**, ICANN-91, Espoo, Finland, June 1991. Vol 1, P605.

Vu V.V, *et al*, **Time Encoded Matrices as Input Data to Artificial Neural Networks for Condition Monitoring Applications**, COMADEM '91, Southampton, July 1991. p31.

The International Conference on the Joining of Materials - JOM

6th International Conference

Helsingor, Denmark, April 5 - 7, 1993

pp 60 - 67

THE REAL TIME ANALYSIS OF ACOUSTIC WELD EMISSIONS USING NEURAL NETWORKS

J.R. McCardle B.Sc., K.L. Taylor-Burge B.Sc., M.A., R.R. Stroud M.Tech., Ph.D., T.J. Harris B.Sc., Ph.D.

Neural Applications Group, Department of Design, Brunel University, United Kingdom

Synopsis.

Artificial Neural Networks (ANNs) are becoming an increasingly viable computing tool in control scenarios where human expertise is so often required. The development of software emulations and dedicated VLSI devices is proving successful in real world applications where complex signal analysis, pattern recognition and discrimination are important factors.

An established observation is that a skilled welder is able to monitor a manual arc welding process by subconsciously changing the position of the electrode in response to an adverse change in audible process noise. Expert systems applied to the analysis of chaotic acoustic emissions have failed to establish any salient information due to the inabilities of conventional architectures in processing vast quantities of erratic data at real time speeds.

This paper describes the application of a hybrid ANN system, utilising a combination of multiple ANN architectures and conventional techniques, to establish system parameter acoustic signatures for subsequent on line control.

Introduction.

The Neural Applications Group at Brunel University, UK, has aimed its research at the development of autonomous products and control systems utilising state of the art technologies. One area of continuing study has been the development of a fully integrated, hybrid control system for automated welding processes; in particular industrial semi-automated submerged arc welding (SAW).

The successful application of Artificial Neural Networks (ANNs) to interpret ultrasonic echoes of the welding head in real time has yielded a system capable of seam tracking and monitoring weld penetration, highlighting the ANNs computational power when dealing with erratic, noise polluted data [1].

The relationship between the process variables of a weld and acoustic emission (AE), both audible and inaudible, has been the subject of various much research. It is an established observation that a skilled manual welder is able to intrinsically position the electrode and vary the arc length in response to adverse fluctuations of process noise.

Recent attempts to establish salient relationships between AEs and process parameters have utilised conventional digital signal processing techniques and expert systems to classify the data [2]. The inability of these systems to respond correctly to novel and erratic inputs was compounded by speed limitations in manipulating vast quantities of data in real time, causing data explosions and consequently negative results [3].

The aim of this research is the development of a hybrid system incorporating conventional DSP methods and ANN techniques to classify features within acoustic data relating to weld parameter fluctuations, in real time. It is envisaged that the acoustic method will complement the existing ultrasonic array in ultimately providing a single system able to monitor and control all the process variables.

The Weld Parameters

The creation of an optimum submerged arc weld is dependent upon six primary parameters;

weld plate preparation (fit up)
position of the electrode (seam tracking)
rate of travel along the seam
consistency of gravitational feed of the granular flux
weld voltage
weld current (determined by the rate of feed of the sacrificial electrode).

The nature of raw acoustic data is such that emissions could contain clues relating to all these parameters and eventual investigation will be given to achieving all possible acoustic signatures for diagnostic purposes. The primary objective, however, is the identification of signal patterns associated with changes in weld voltage. This method, when run in conjunction with the existing ultrasonic weld penetration monitor, would render a system capable of maintaining an optimum balance between voltage and current and hence achieve on line weld stability.

Data Acquisition

The transducer element consisted of three omni-directional electret condenser microphones (ECMs). The ECMs selected exhibited uniform frequency response characteristics suitable for infra, audible and ultrasound detection. The operation of ECMs negate the possible influence of induced noise created by the magnetic flux of the welding arc [4]. The transducer was mounted 300mm from the welding head together with suitably noise shielded pre-amplification.

The signals were then subjected to a series of eight active bandpass filters within the audible range, a low pass filter to isolate infrasound (break frequency 15Hz) and a high pass filter (break frequency 20kHz) to emphasise ultrasonic frequencies.

The data manipulation was achieved with the TMS 320C30 system board hosting Hypersignal Workstation software. The transient capture sampling frequency was set to 40kHz and 100ksamples enabling 2.5 sec of data capture at a 16 bit resolution. Samples were taken of ambient conditions as well as singular and combined readings of welding ancillary equipment such as the mains transformer, fume extractor fan and the kinetic control system. This provided sound intensity levels and features which could be identified within the final weld recordings.

Three welding runs were then initiated, with preset parameters and ten readings taken from each. The initial weld parameters were set at the default, the second run with low voltage and the final with high voltage in relation to weld current as shown in table 1.

	V	I
Weld Run 1 (default)	41.1	820A
Weld Run 2	29.7	785A
Weld Run 3	61.2	815A

Table 1.
Weld Voltage & Current Settings

The captured data was then subject to a 1024 point FFT analysis and a contour map obtained as shown in Figs 2, 3 & 4. Further manipulation enabled a power spectra analysis by performing an rms of the 715 FFT frames over the 2.5 sec capture period. This effectively removed transient noise pollution and yielded the most prevalent features in the frequency domain. Real time frequency and power spectrums were then monitored to gauge consistency over a complete weld run of 1 Metre lengths.

The Neural Classification Approach.

ANNs have been successfully implemented in scenarios where high speed signal processing of noisy or corrupted data has thwarted expert systems [2, 3, 5, 6]. The difference in the two systems stems from their architectures. The conventional computer relies on a single powerful processor passing synchronised data, whereas, neural systems contain many simple processors (neurons) heavily interconnected. This enables the parallel processing necessary to deal with complex signals at high speed. The advantages of networks lie in their ability to *generalise* and have consequently proven themselves in predictive systems requiring the assessment of novel data [7].

Observations of biological neural systems have shown that different neurons and topologies perform different functions. Similarly, differing ANN architectures and learning algorithms exhibit different characteristics of cognition and recognition. These characteristics can be exploited to great effect if the correct paradigm is selected for the situation. Pattern recognition problems are relieved by the extraction of salient features, by pre-processing, on which a pattern classifying network can base its considerations [8].

The well established ANN architecture of Back Error Propogation [9], often refered to as the 'work-horse' of neural computing, could be exploited in the area of signal processing. They have been successfully employed in bandpass functions and signal enhancement and could be used in pre-processing raw acoustic signals prior to DSP operation.

The basis of the signal pattern classifier for this research is the Kohonen Self - Organising Feature Map [10]. It is a two layered network of fully connected neurons that can self organise, from a random starting point, a topological map showing the natural relationships of the patterns used in its training data sets. This is achieved by the competitive activation of an output neuron dictated by the learning algorithm. The output neurons are arranged in a matrix so that each has theoretical neighbours. The activation threshold of neighbouring neurons are adjusted proportionally to the distance in vector space from the winning neuron. In this way each neuron is inhibited or encouraged to activate when presented with the next training set. The training continues

until a state of equilibrium occurs or an acceptable level of accuracy is reached. This system has proven successful in the interpretation of speech phonemes [11] and the diagnosis of lung infections via AEs [12]. In the same manner it is anticipated that this paradigm is capable of isolating similarities and grouping the signals common to unstable welds and separating them from stable inputs. Further sub-divisions within the topological map would include the remaining parameters which influence unstable welds.

Fig.1 illustrates the experimental set up. Research in this field of application continues.

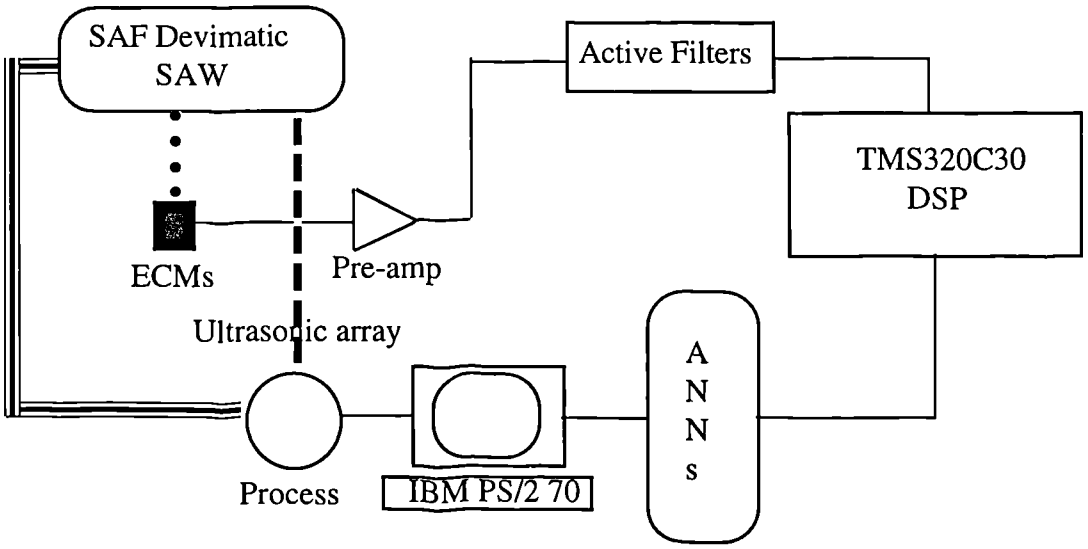


Figure 1
Experimental set up

Results

So far research has concentrated on establishing usable characteristics in the frequency domain for introduction to a suitably tailored Kohonen classifier network.

Figures 2, 3 & 4 illustrate frequency contour mappings of the recorded data for weld runs 1, 2 & 3 (see table 1) respectively, from which the following observations were made:-

1. The vertical readings indicate noise from gas leakage through the flux. The concentration of this noise being most apparent on the low voltage run.
2. The high voltage and default runs show a consistent frequency feature in the 14kHz to 16kHz range.
3. The frequency intensities within this bandwidth are greater in the high voltage weld.
4. There is a substantial absence of low frequencies, up to 4kHz, on the default weld setting.

Figures 5, 6 & 7 illustrate the comparisons of real time freeze frame FFT analysis taken from a continuous weld run. The signals tend to enforce the observations made from the frequency contour map. The salient features are further enhanced with the real time power spectra analysis giving a 1024 point FFT 25 frame rms. An example for a high voltage setting is shown in Fig. 8.

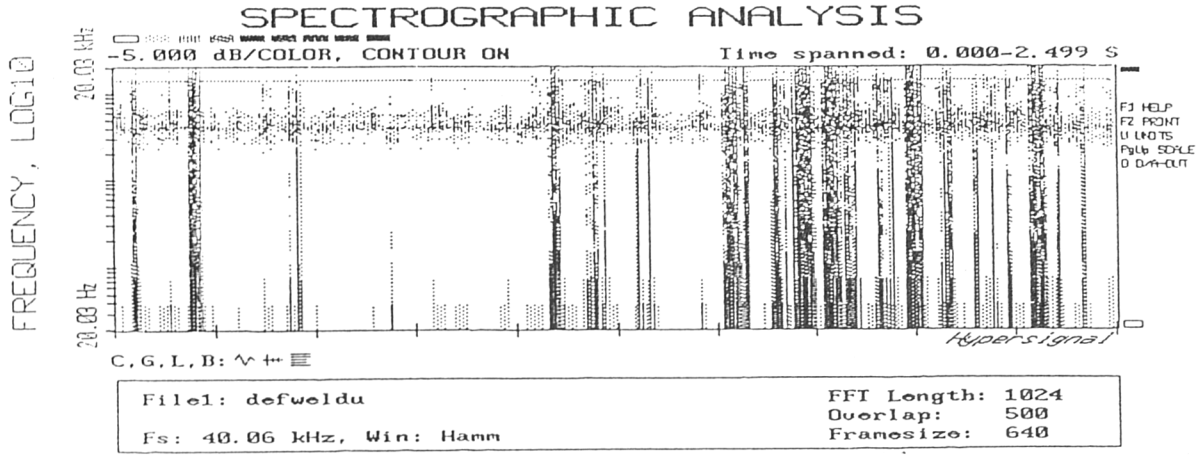


Figure 2.
Frequency Contour Map, Default Weld.

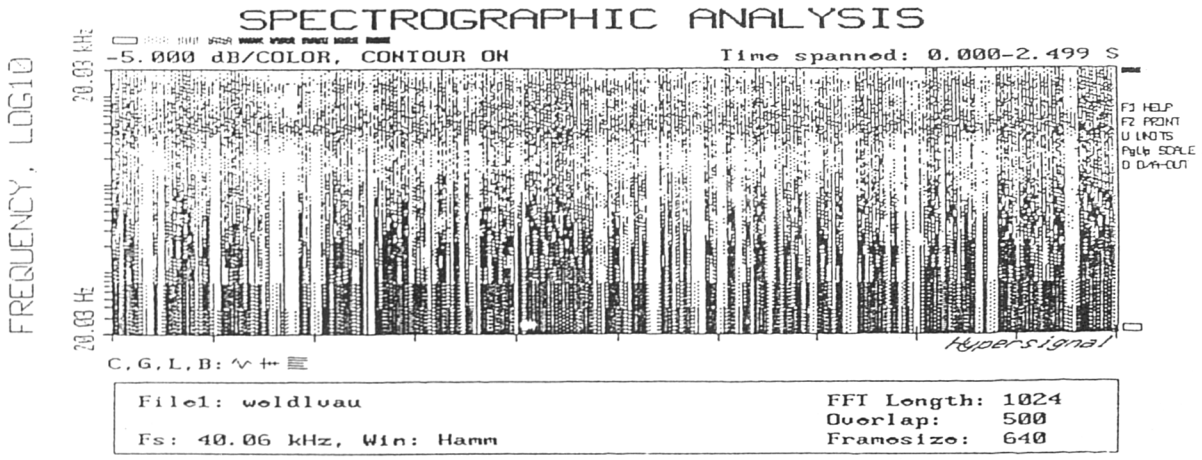


Figure 3.
Frequency Contour Map, Low Voltage.

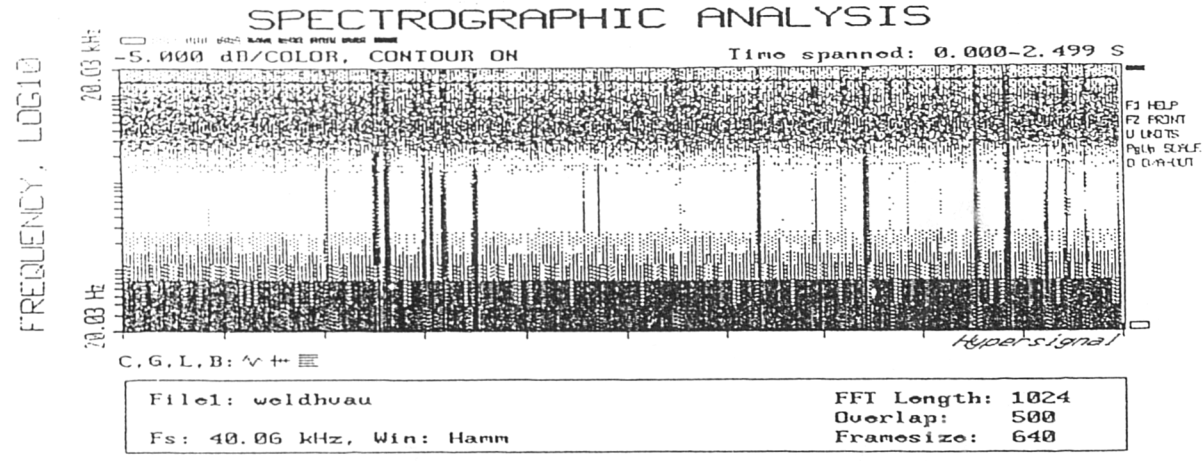


Figure 4.
Frequency Contour Map, High Voltage.

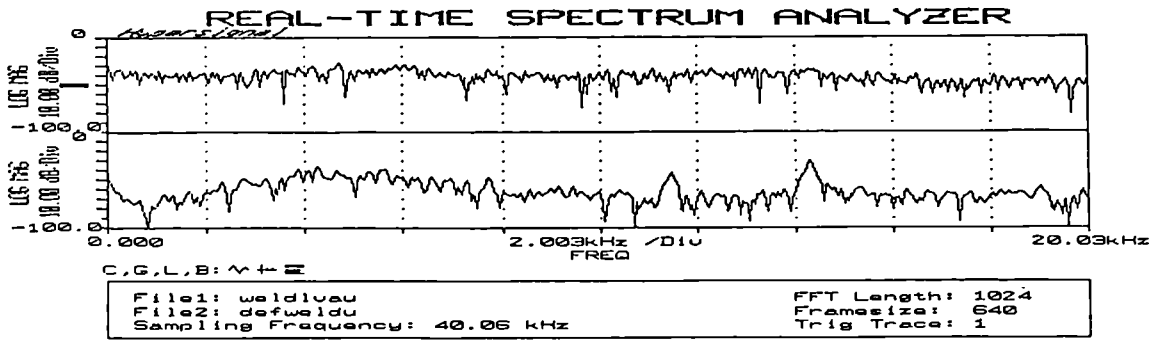


Figure 5.
Comparative Freeze Frame FFT, Low Voltage weld & Default Weld

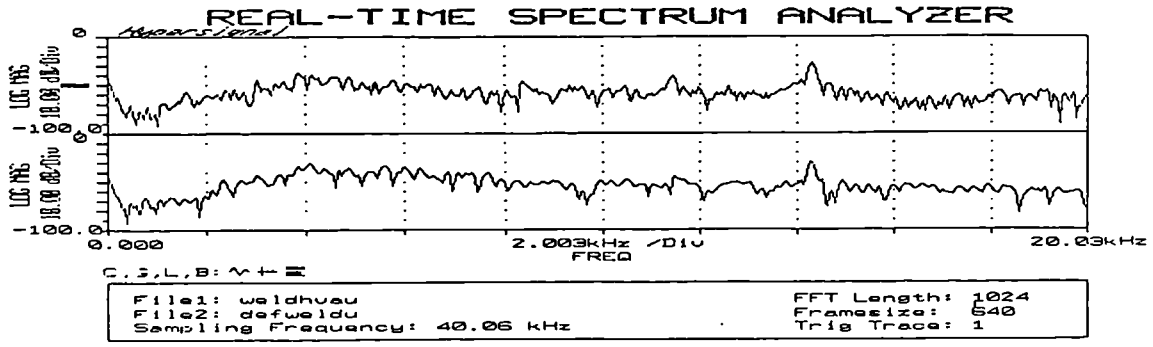


Figure 6.
Comparative Freeze Frame FFT, High Voltage Weld & Default Weld

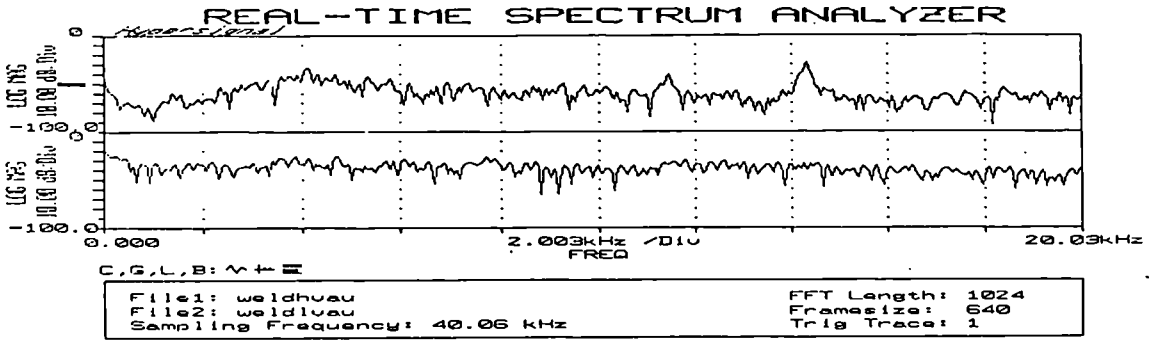


Figure 7.
Comparative Freeze Frame FFT, High Voltage & Low Voltage Welds

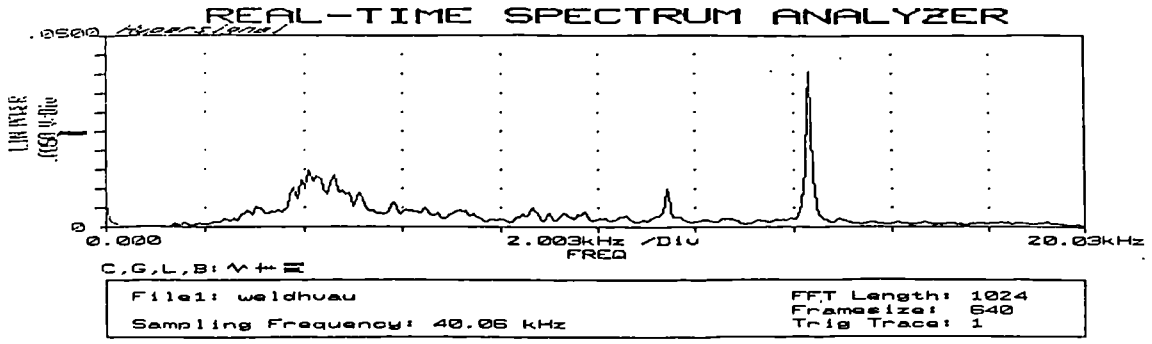


Figure 8.
Power Spectra, High Voltage. (1024 point FFT, 25 frame averaging.)

Conclusions

Conventional DSP methods utilising the TMS320C30 are capable of providing real time signals, in the frequency domain, which yield distinguishable features associated with non-optimum weld voltage settings for SAW.

It is feasible that a Kohonen Feature Map could be generated to indicate stable and unstable weld states from successive on-line power spectra.

Complex signal processing and classification of noise corrupted data often require the combination of conventional digital and neural techniques.

The research and development of weld automation continues, with the application of ANN technology, toward the provision of a fully integrated welding system.

References

[1] Stroud,R.R, Harris, T.J, Taylor-Burge, K.L,.'Neural Networks in Automated Weld Control', TWI, Gateshead, UK, 1991.

[2] Chawla, K.S, Norrish, J, 'Real Time Quality Monitoring Using Analysis of the Arc Sound', TWI, Cambridge, UK, 1992.

[3] BRITE EuRam project - BREU 0415 BE-4006-90.

[4] International Institute of Welding, 'The Physics of Welding', Pergamon Press, 1984.

[5] Vu, V, *et al*, 'Time Encoded Matrices as Input Data to Artificial Neural Networks for Condition Monitoring Applications', COMADEM '91, Southampton, UK, July 1991.

[6] O'Brien, J.C, *et al*, 'Can Neural Nets Work in Condition Monitoring ?', COMADEM '92, Senlis, France, July 1992.

[7] Tanaka, T, *et al*, ' Trouble Forecasting by Multi-Neural Networks on Continuous Casting Process of Steel Production', Proc 1st ICANN, Espoo, Finland, June 1991.

[8] Harris, T.J, 'Neural Networks and their Application to Diagnostics and Control', COMADEM '92, Senlis, France, July 1992.

[9] Hecht - Nielson, R, 'Neurocomputing', Addison - Wesley Publishing Company, 1991.

[10] Dayhoff, J, 'Neural network Architectures. An Introduction.', Van Nostrand Reinhold, 1990.

[11] Kohonen, T, 'The Neural Phonetic Typewriter', Computer 21(3) 11-22, 1988.

[12] Kallio,K, *et al*, 'Classification of Lung Sounds by Self Organising Feature Maps', 1st ICANN, Espoo, Finland, 1991.

[13] Hyperception, 'Hypersignal Workstation', Advanced Signal Processor Environment, Revision 2.0, Feb 1991, Software and User Manual.

The Welding Institute - TWI

5th International Conference

" Computer Technology in Welding "

Paris, France, June 15 - 16, 1994

Paper 28

The Analysis of Airborne Acoustics of S.A.W. Using Neural Networks

J.R. McCardle	BSc	Brunel University
K. Taylor-Burge	BSc, MA	Brunel University
R.R. Stroud	MTech, PhD	Brunel University
T.J. Harris	BSc, PhD	Brunel University

SUMMARY

The analysis of acoustic emissions for machine health monitoring has made rapid advances in the last five years due to a revival of interest in the application of Artificial Neural Networks (ANNs). Complex signal analysis, which has often thwarted conventional statistical methods and expert systems, is now more possible with the introduction of 'neural' based computing methods.

Acoustic emissions from welding processes are well documented. In particular, it has been established that a manual welder is capable of making intrinsic decisions concerning electrode position based on process noise.

The analysis of time / amplitude signals and Fast Fourier Transforms (FFTs), within salient frequency bandwidths of the weld acoustic, has yielded erratic, unpredictable and noise polluted data. Extracting a meaningful interpretation from this data is computationally intensive when utilising standard statistical methods and leads to data explosions, especially when an 'on-line' corrective control signal is required.

An Artificial Neural Network is 'trained' on examples from acquired data and performs a robust signal recognition task rather than relying on a programmed set of data samples as in the case of an expert system. This technique enables the network to generalise and, as a consequence, allows the input data to be erratic, erroneous and even incomplete.

This research defines the development of a hybrid system, utilising high speed date capture and FFT computation for the signal pre-processing and a 'self organising' network paradigm to establish weld stability and real time corrective control of the process parameters.

The paper describes a successful application of a Neural Network hybrid system to determine weld stability in submerged arc welding (S.A.W) through the interpretation of airborne acoustics.

INTRODUCTION

The Neural Applications Group, within the Department of Design at Brunel University, UK, Directs its' research at the application of Artificial Neural Networks to 'real' industrial problems. The major focus is towards the welding industry, developing process control and NDT techniques. One area in particular has been the application of ANNs to a fully integrated control system for Submerged Arc Welding.

ANNs were principally developed as an attempt to replicate the topology and operation of biological neural systems. Advances have yielded models which 'train' or 'learn' output responses from given data. The consequences of this technique is an emulated biological system capable of dealing with erratic or even incomplete input data.

Observations of skilled manual welders has shown a subconscious tendency to change the angle of the electrode and length of arc by listening to adverse fluctuations in the process noise. This has resulted in much research into the analysis of airborne acoustic emissions (AEs) of welding processes.

Attempts have been made to interpret AEs utilising statistical and rule based techniques (1). The inflexibility of these techniques in dealing with noise polluted data, together with speed limitations, provided little success. Acoustic emissions from SAW requires the classification of large quantities of erratic data samples for any meaningful interpretation to be made (2). Many references can be found to the successful application of ANNs in signal processing (3) and condition monitoring (4,5,6).

In SAW, Logical Neural Networks (7) have been successfully employed to window and interpret noisy ultrasonic echoes from the weld head for seam tracking and weld penetration measurement in real time (8).

Although AEs from welding processes are potentially information rich, to limit the feasibility study into the ANN approach to signal classification, this research has focused on the detection of instabilities caused by non optimum voltage settings. Further studies are to be made into the identification of wayward variables during the welding process.

EXPERIMENTATION

The parameter setting for an optimum weld were chosen by weld profile dimensions obtained from bead on plate post weld cross sectional inspection. Mild Steel plate, 25mm thick, was used with a root penetration of 15mm. The travel speed, current (determined by the rate of feed of the sacrificial electrode) and voltage were then considered optimum.

A transducer element consisting of three omni-directional electret condenser microphones (ECMs) were mounted 300mm from the welding head. Each ECM exhibited suitable gains and frequency responses for infra, audible and ultra sound ranges up to 40 kHz. Suitably pre-amplified, signals obtained were subjected to 8 active bandpass and appropriate anti-aliasing filters before being analysed by standard DSP methods.

Previous investigation, utilising a TMS320C30 system board hosting Hypersignal Workstation software revealed salient frequencies during welding of between 15Hz and 10kHz (2). Frequency 'signatures' of ambient conditions and ancillary equipment such as the mains transformer, fume extractor, kinetic control system and wire feed motor etc., were identified from a 1024 point FFT analysis as shown in Fig 1.

The initial analysis provided the sampling frequency and resolution necessary to isolate salient bandwidths suitable for input to an ANN model. The TMS320C30 was set up to perform a 128 point Radix 2 FFT (yielding 64 real data points), sampling frequency of

20kHz with a 16 bit resolution.

Three welding runs were the initiated over a 1 metre length. Ten recordings were taken from each and monitored to gauge signal consistency. The wire feed rate (current) and travel speed were held constant while the voltage, primarily at optimum level, was changed to a low unstable level followed by a high unstable level as recorded in Table 1.

Further transient noise was removed from the FFT analysis by a statistical method, taking the r.m.s over 25 acquired FFT frames. The 64 real data points of each resulting FFT was treated as a 64 component vector and used as input to the ANN classifying model.

TABLE 1. Voltage, Current and Travel Speed Settings

	V	I	S
WELD 1 (optimum)	41.1 V	820 A	6.5mm/S
WELD 2 (low voltage unstable)	29.7 V	820 A	6.5mm/S
WELD 3 (high voltage unstable)	61.2 V	820 A	6.5mm/S

NEURAL NETWORK APPLICATION

Studies of biological neural systems have revealed that differing neuronal topologies as well as individual neurons perform different functions (9). The various characteristics of recognition and abilities in learning are replicated by ANN architectures and learning algorithms. The learning categories are generally of two types, Supervised or Unsupervised. Both these methods are utilised by two ANN paradigms which have emerged to become the most widely adopted methods in applications.

Back Propagation Networks (BPN)

BPNs provide a method of supervised learning . In essence the input data is given with a desired output, and the learning technique is one of re-inforcement also known as Hebbian learning (10). The architecture is generally a 3 layer network consisting of an input layer, a middle or 'hidden' layer and an output layer. Each layer is fully interconnected to the proceeding layer. The number of neurons in the input and output layers depends on the input pattern organisation and the intended output classification. The number of neurons in the hidden layer is still the subject of much debate, but the optimum number is dependent upon the complexity of the classification problem.

For this experimentation the input layer consists of 64 neurons, one for each component of the input vector, and an output layer of 3 neurons for the classification of each weld run voltage level. The number of hidden layer neurons was initially set at 12, as shown in Fig 2.

In the training stages, the input vector components are transferred through the connected neuronal transfer functions via the synaptic weights (Fig. 3). The resulting values given at the output layer are compared with the desired output. A function of the error is used to adjust the value of the synaptic weights thereby making a subsequently similar input generate an output closer to the desired value.

The success of the BPN model is dependent upon the distribution or the degree of correlation of the input data. Although ANNs have the inherent ability to deal with highly correlated data, the BPN model uses a gradient descent method (11) in its learning algorithm to establish an optimum boundary between similar data patterns. If the separation is complex, it can often lead the network to converge on a local rather than a global minimum resulting in a failure to train or an unreliable output response.

Self Organising Feature Map (SOM)

The SOM model utilises an unsupervised or competitive learning technique. This method replicates 'learning by experience' rather than by taught responses. The SOM consists of two layers, an input layer consisting of a number of neurons, suitable for the input data, fully interconnected to an output matrix or 'slab' of neurons. The number of neurons in the output slab depends on the required resolution of the desired output classification. (Fig. 4).

During the training phase, the input vector is compared with the randomly initiated synaptic values of each output neuron. The output neuron possessing the least error in vector or Euclidean space is considered the winner and its synapses are updated proportionally to the error as dictated by the learning algorithm. Furthermore, each winning neuron has neighbours within the output slab. These neighbouring neurons are also updated in such a way as to encourage or inhibit its chances of winning when a subsequently similar vector is compared. This training continues until an acceptable state of equilibrium is reached.

In this way similar vector patterns are clustered within a multi-dimensional vector space or hyperspace. Depending upon the density function and the correlation of the input data, natural separations can occur. The degree of separation is measured in terms of error distances within hyperspace. Consequently this method requires a comprehensive understanding of the nature of the input data and necessitates the re-analysis of the clustered vector patterns.

In general, the problems associated with pattern recognition by means of ANNs is greatly relieved by careful signal pre-processing. Training data should be of a standard which highlights salient features. This not only reduces training times but increases the chances of achieving a fully optimised network (12).

Depending upon the number of training data sets and the size of the network, training can be a time consuming phase. The benefits are yielded from the network's inherent ability to generalise on previously unseen data patterns together with fast response times. The main limitation of this technique is memory allocation for computed vector analysis.

Network Implementation

The ANN models used for this research were software emulated. Commercially available software, in the form of the NT5000 from Neural Technologies was used for the BPN and custom compiled software for the SOM model. Although the networks are operated on a von Neumann derived computer architecture in which data is actually processed serially, the speed of operation of a trained system enables viable real time application (11). This is due to the simulated parallel processing of the input vectors.

RESULTS

Typical FFT frames introduced to the network models are shown in Figures 5, 6 and 7. Five hundred rms frames, sampled from a weld length of approximately 0.1 meters or 15 seconds, from each weld run was used as training data. Although pre-processed the FFTs are still erratic and certain characteristics are apparently similar in all three weld runs. This was evident from the results of the attempts to train the BPN model.

The primary training session of the BPN model yielded negative convergence after 15 minutes. The high correlation of the input data proved too complex for the initial system architecture to reach a satisfactory conclusion.

The architecture was subsequently amended by increasing the hidden layer neurons up to 24. Further training eventually provided a convergence after 25 minutes. The gradient descent method was assisted by the frequent injection of noise known as weight joggling. This technique is designed to assist the prevention of local rather than global minima detection.

Testing the network entailed the input of previously unseen data vectors in an attempt to correctly classify the three levels of weld stability. The success rate was gauged on a percentage correct basis. Results proved inconclusive with all three test sets yielding between 42% and 51% correct classification.

Further investigation is in progress for the application of optimisation techniques for BPNs.

The self organisation of the SOM model negates the need for a global convergence. The input data vectors are automatically clustered within the 64 dimensional hyperspace. The clusters are of similar input patterns and subsequent test vectors are associated to or within the trained cluster by means of a calculated error distance.

The network was initially trained on the signals obtained from the optimum weld run and data from the unstable welds used as test vectors. The high dimensionality of the data compounds the graphical representation of the clusters which necessitates the re-analysis of the patterns and calculated error distances. Analysis of the resulting errors showed that the high voltage unstable weld was regarded as a sub-set of the optimum weld cluster, whilst the low voltage unstable data was clearly segregated as shown in Fig. 8.

The second training run was initiated with the high voltage weld data and the subsequent errors from the optimum and low voltage test patterns assessed. Almost full separation was achieved with a minor union evident between the optimum and high voltage data

clusters as shown in Fig. 9.

The training times for the SOM amounted to approximately 15 minutes on a 486 / 66MHz PC machine. Response times to individual input vector classification was approximately 0.02 microseconds with total cluster comparison in 3.5 milliseconds.

The pattern separation is achieved with data representing the extremes of voltage level instability. If an appropriate corrective control signal is to be generated, a vector signal needs to be quantified from the individual or clustered test pattern errors. The classification of AEs with respect to intermediate discrete voltage settings requires a calibration technique capable of providing such a signal. As the voltage is adjusted from unstable to stable the generated data clusters will converge reducing the computed errors, however, data separation also reduces and cluster boundaries become unclear. Consequently, the detection of the *onset* of weld instability becomes more complex.

Solutions to this problem are currently under investigation.. Fuzzy logic and control (13) and further ANN models which exhibit predictive capabilities, such as Probablistic Networks and Spation Temporal Networks (14), are invisaged as providing possible remedies. Thes systems yield outputs relating to successively occurring data patterns and consequently emphasise cluster density functions.

CONCLUSIONS

The airborne acoustic emissions from submerged arc welding yields highly correlated erratic data which requires specific pre-processing to emphasise salient features.

Conventional DSP methods including FFT and averaging techniques are capable of revealing features, in real time, which are compatible with certain Artificial Neural Network models.

A Back Propagation Network proved inconclusive when attempting to classify data patterns derived from acoustic emissions associated with the voltage stability of submerged arc welding.

A Self Organising Feature Map successfully separated and classified low, high and optimum voltage settings from the acoustic emissions of submerged arc welding.

Complex and erratic signal interpretation often requires both conventional statistical and neural network techniques.

Research continues into neural network optimisation, corrective control signal generation and the development of a fully integrated welding control system.

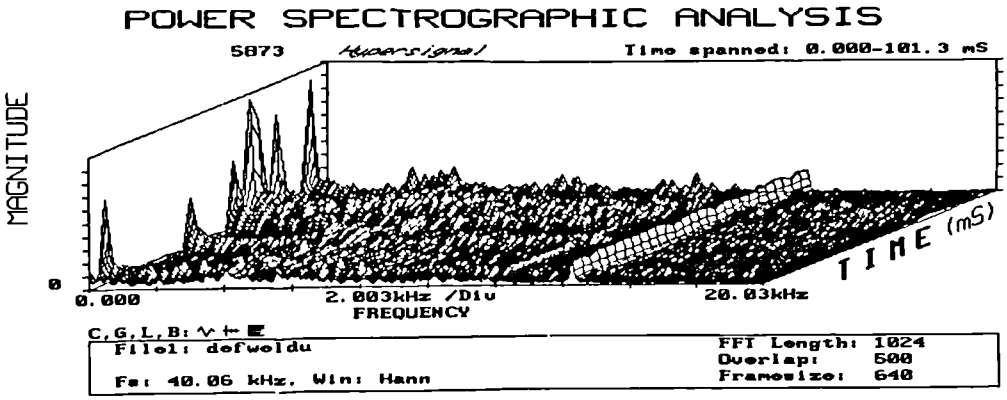


Figure 1.
1024 Point FFT of Typical optimum weld run.

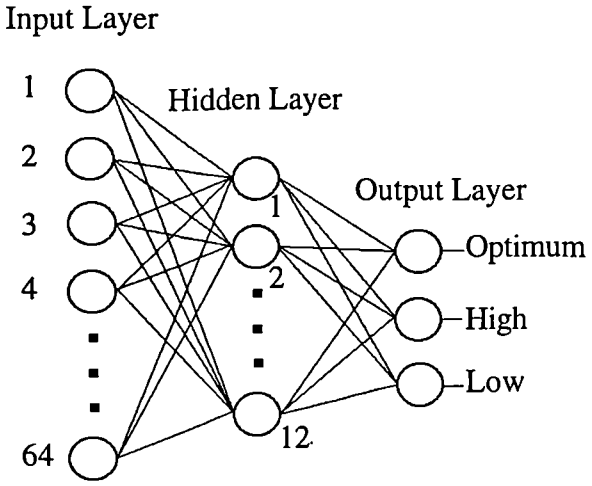


Figure 2.
Typical Back Propogation architecture

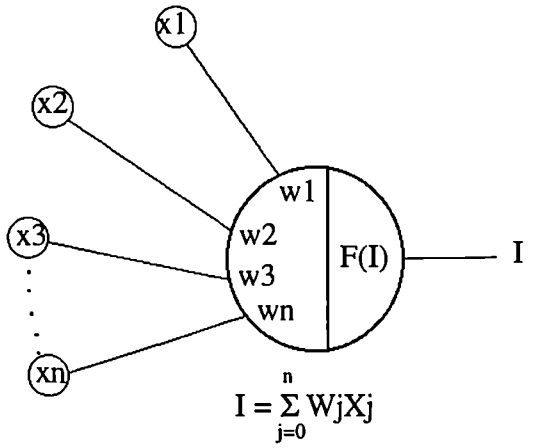


Figure 3.
Typical artificial neuron showing
transfer function.

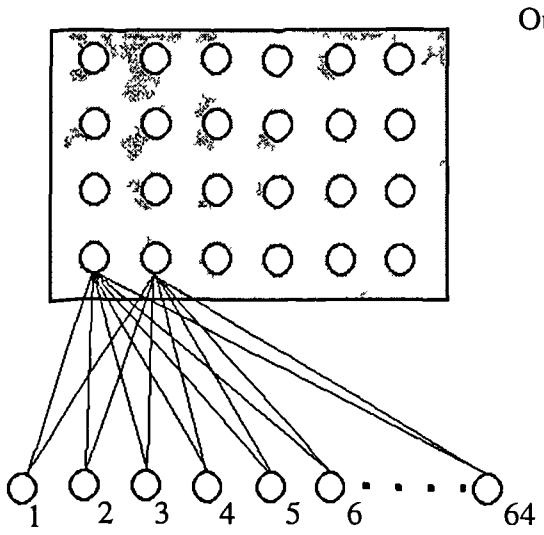


Figure 4.
Self Organising Feature Map architecture

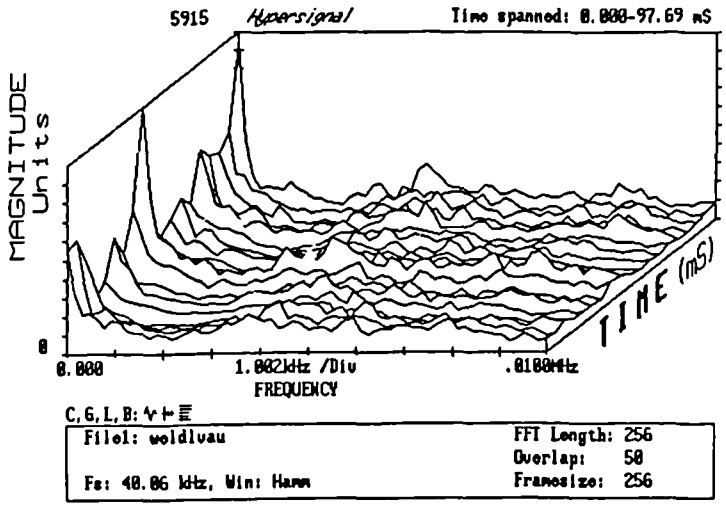


Figure 5.
128 Point FFT, low voltage signal

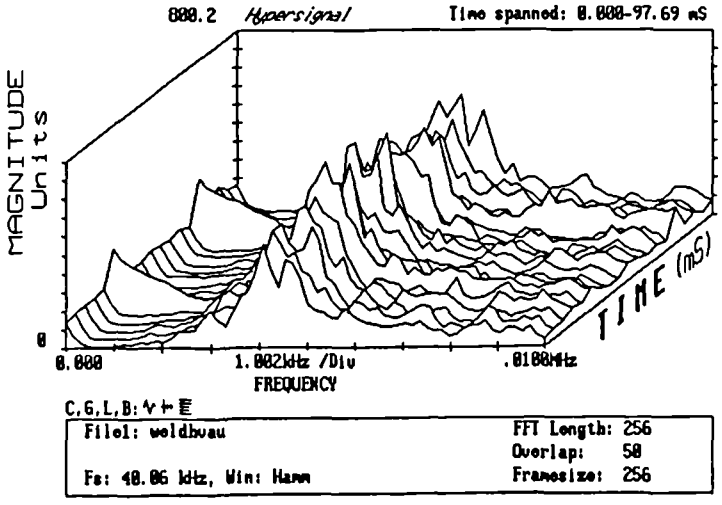


Figure 6.
128 Point FFT, high voltage signal

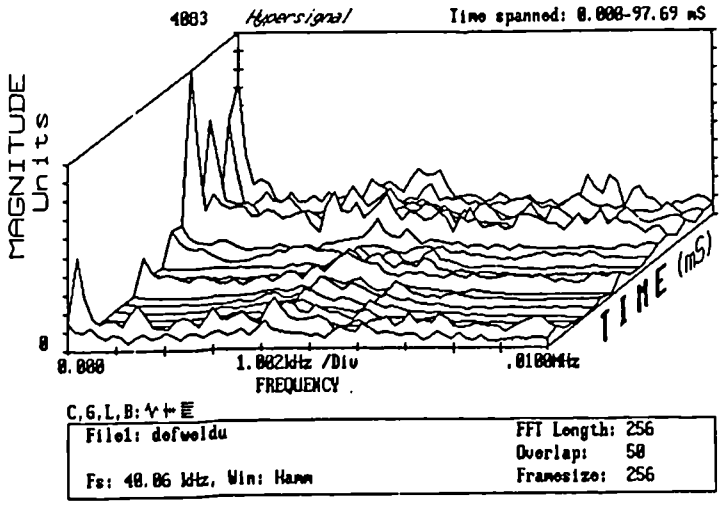


Figure 7.
128 Point FFT, optimum voltage signal

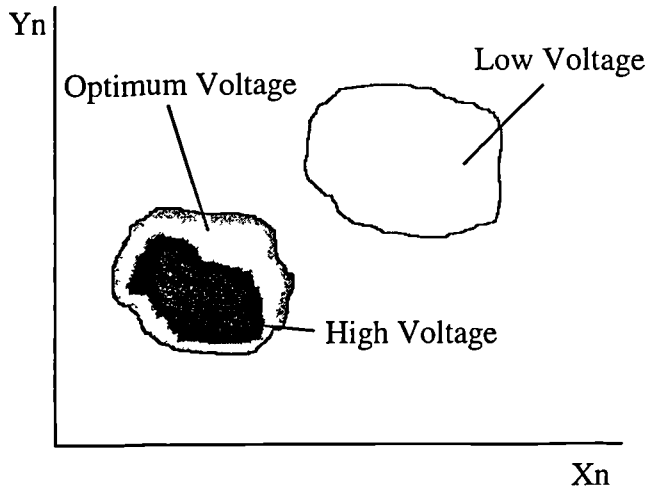


Figure 8.
Data cluster representation with high voltage
a subset of optimum voltage

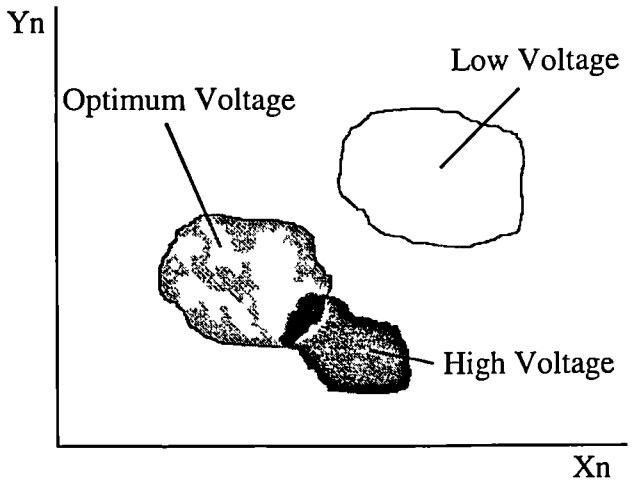


Figure 9.
Retraining with high voltage data causes
further separation with a minor union

REFERENCES

(1) Chawla, K.S, Norrish, J. "Real Time Quality Monitoring Using Analysis of the Arc Sound", Paper 17, TWI, Cambridge, UK, 1992.

(2) Taylor-Burge et al " The Real Time Analysis of Acoustic Weld Emission Using Neural Networks", P.60, Proceedings of JOM-6, Helsingor, Denmark, 1993.

(3) Hecht-Nielson, R. "Neurocomputing", Addison-Wesley Publishing Company, 1991.

(4) Vu, V. et al, "Time Encoded Matrices as Input Data to Artificial Neural Networks for Condition Monitoring Applications", COMADEM '91, Southampton, UK, 1991.

(5) O'Brien, J.C, et al, "Can Neural Nets Work in Condition Monitoring ?", COMADEM'92, Senlis, France, 1992.

(6) Kallio, K., et al, "Classification of Lung Sounds by Self Organising Feature Maps", 1st ICANN, Espoo, Finland, 1991.

(7) Aleksander, I., "Connectionism or Weightless Neurocomputing ?", 1st ICANN, Espoo, Finland, 1991.

(8) Stroud, R.R, et al, "Neural Networks in Automated Weld Control", TWI, Gateshead, UK, 1991.

(9) Ruch, T.C. "Neurophysiology", W.B. Saunders & Co., 1965.

(10) Dayhoff, J., "Neural Network Architectures: An Introduction", Van Nostrand Reinhold,1990.

(11) Eberhart, R.C. & Dobbins, R.W, "Neural Network PC Tools, A practicle Guide", Academic Press Inc, 1990.

(12) Harris, T.J, "Neural Networks and their Application to Diagnostics and Control", COMADEM'92, Senlis, France, 1992.

(13) Nadler, M., & Smith, E.P, "Pattern Recognition Engineering", John Wiley & Sons Inc,1992.

(14) Grossberg, S. "Embedding Fields: Underlying Philosophy, Mathematics,and Applications to Psychology, Physiology and Anatomy", Journal of Cybernetics, Volume 1, pp28-50, 1971.

The Welding Institute - TWI

6th International Conference

" Computer Technology in Welding "

Lanaken, Belgium, June 9 - 12, 1996

Paper 33

The Use of Arc Sound and On-Line Ultrasonic Signal Processing as a Monitoring Medium for Welding

J.R. McCardle	BSc	Dept. of Design, Brunel University, U.K.
S.S. Swallow	BSc	Dept. of Design, Brunel University, U.K.
R.R. Stroud	M.Tech, Ph.D	Dept. of Design, Brunel University, U.K.
K.L. Taylor-Burge	Bsc, MA	Dept. of Design, Brunel University, U.K.

SUMMARY

Monitoring the welding process on-line with ultrasound is problematic, but promises great rewards. A fast classifier is required to exploit the redundancy available in ultrasonic interrogation and ensure an adequate signal / noise ratio.

TARDIS is such a classifier, using logical neural network techniques and dedicated hardware. The classification performance of TARDIS alone is noisy, but exceptionally fast. This speed of operation can be used to offset the fuzziness of individual classifications, using higher order correlations.

The expert manual welder is capable of simultaneously monitoring visual and acoustic data and, coupled with a knowledge of the process and past experience, is able to attempt an optimum weld. Observations of skilled manual welders has shown a subconscious tendency to change the angle of the electrode and length of arc by listening to adverse fluctuations in the process noise in addition to visual assessment. This has resulted in much research into the analysis of airborne acoustic emissions (AEs) of welding processes. It is evident that to artificially copy these skills requires a fast, robust signal processing and pattern recognition technique similar the known architecture and operation of the brain.

The Department of Design, Brunel University, has been researching the possibilities of including the monitoring of airborne acoustic emissions as an additional correcting factor in automated weld process control. Salient relationships between acoustic emissions and process parameters using off-line statistical techniques has been established, however, real time application remains problematic due to the computational intensity of such methods.

Statistical approaches to the interpretation of arc sounds relies on the direct correlation observable between the acquired signal or its' various transforms and the monitored process parameters. The method is a time consuming and often mathematically gruelling . Artificial neural networks (ANNs) provide an alternative. By the construction of different architectures and the application of various learning algorithms ANNs can provide a noise tolerant adaptive knowledge acquisition system.

The work discussed in this paper illustrates the methods of signal preprocessing and utilisation of artificial neural networks to interpret arc sounds. Techniques are used to filter and compress high dimensional erratic data patterns to form classifiable representations of the process state. Real time scenarios are discussed together with commercially viable hardware solutions.

TARDIS: A NEURAL SIGNAL PROCESSOR FOR WELD ULTRASOUND MONITORING

On-line ultrasonic monitoring of welding

Commercial seam tracking seems dominated by optical systems. CCD camera and spot triangulation technologies have proven accurate, cheap, rugged and compact and offer the potential of making joint metrology or even weldpool qualitative measurements (2). Unfortunately, the bulk of this research endeavour is of no use in monitoring submerged-arc welding. The flux covering complicates seam tracking and precludes the measurement of weldpool features.

On-line ultrasonic interrogation of the weldpool area 'Figure 1' offers direct measurement of penetration, is a front face technique, circumvents weldpool obscuration and is a well understood, much implemented technology in the sister field of NDT.

Transit time is the A-scan feature that is ultimately required. From this, the distance to the reflecting solid/liquid interface (for penetration monitoring) or unwelded plate edge (for seam tracking) can be determined. Identifying the correct echo is necessary for extraction of the transit time, so the echo's amplitude and shape, or profile, must be considered.

The over-riding problem when processing weld ultrasound is contamination of this transient echoes with noise. Echo amplitudes perhaps suffer most. Echoes are attenuated by a combination of HAZ scattering, molten interface scattering and variations in coupling efficiency. All of these effects are dynamic and non-deterministic, with the added complication that attenuated energy often recurs as noise.

A variety of noise sources act upon received transit times, called delay noise. There is an offset due to high temperature velocity reduction and a possible non-deterministic delay occurs dependant upon the incident angle to a weldpool's dendritic interface.

The associated shapes of the echoes are similarly erratic. The elementary wavelets of differing phase that make up a pulse can be considered to each undergo attenuation and delay noise as described above. In fact, there exists no single "ideal" echo (7) because there exists no ideal reflecting surface. Robust characterisation of the correct echoes is the technique's greatest problem (4).

However, the high repeat rates of ultrasonic monitoring are an advantage. Each complete echo-return offers an independent measure, with typically thousands generated per second. Updating a welding rig at, say, 20 Hz presents a fiftyfold data redundancy between available measurements and required outputs. Reducing this redundancy with some form of rapid on-line signal averaging is essential for this processing problem (3). It allows the very randomness of the various noise sources to be exploited.

Unfortunately, the pulsewidths and fundamental frequencies of NDT ultrasound demand very fast digitisation, causing an avalanche of data. Straightforward superposition of consecutive echo returns is problematic. Digital hardware to perform the task would be enormous, most general purpose computers too slow and analogue techniques using CCD or SAW devices too extravagant (8).

The usual approach towards large and unwieldy raw data sets is feature extraction, which implies some form of classification. The solution to the weld ultrasound problem thus becomes some very fast, on-line classification technology. TARDIS is such a classifier.

TARDIS

The time / amplitude rapid discriminator, or TARDIS, is a digital pattern correlator designed specifically for the rapid on-line classification required by ultrasound weld monitoring. It is based on n-tuple classification techniques (1) realised in fast bit-slice type dedicated hardware. The architecture is shown in Figure 2.

The TARDIS preprocessor's operation is, briefly, as follows.

1. The ADC generates a new sample. The ADC is on-line, and constantly inspects the ultrasound pressure with zero lag, every 100 ns.

2. Every stored sample value in the array of window registers is shifted to the right by one register. The rightmost, eldest sample is lost and the leftmost register loads the new sample from the ADC.

3. Each register presents its stored sample to its associated logical node. These are bipolar PROM devices, preprogrammed with templates of the sequence to be recognised 'Figure 3'. The 8-bit binary code that represents the sample value is used to address a single location in the PROM.

4. The PROM outputs the data, a "1" or a "0", that it has stored at the selected address. A "1" signifies that this sample value, in this particular position in the window, could be part of the waveform sequence to be recognised. These data values are stored intermediately in pipeline registers, for speed of operation.

5. Three further nodes, that are also fast PROMs, one of which is combined with the state machine, form a summation pyramid, to total the number of affirmative logical node outputs. This total is known as the response level for the logical neural network (which is what the system amounts to). The response is a measure of the similarity of a particular waveform to the preprogrammed waveforms.

6. The state machine and window counter extract the highest response and its timebase position from a string of samples. Thus a most likely timebase position for an echo is extracted from hundreds of samples of raw data. The compression ratio is around 500 to 1. Remember, this data has been classified on-line and could be used to facilitate closed loop control.

TARDIS performance

Figure 4 shows the classification behaviour of the TARDIS system (solid line) as it encounters a plate edge echo (shaded line). The response characteristic is not particularly sharp. This is not unusual for n-tuple schemes, which tend to find themselves applied as

quite "rough-and-ready" but very fast classifiers (6).

If one considers the TARDIS pattern space, the independence of each pattern dimension leads to the formation of cruciform clusters 'Figure 5a'. This is more flexible than linear discriminant methods (Bayesian, FIR matched filter, cross-correlator) 'Figure 5b' but is less so than a minimum-distance (MD), K-nearest neighbour (K-NN) or backpropagation neural network 'Figure 5c'.

In fact, an inspection of the system's performance over complete echo-returns Figure 6' reveals how "rough" the classification performance is. The bogus classification peaks, incidentally, may be stray reflections from the seam, interface noise or occasionally the default result of the total loss of any recognisable signal.

Over many echo returns, of the many varying signal / noise qualities characteristic of tracking a hot weldplate with a moving probe, the classification performance of TARDIS is visibly fragile 'Figure 7'. A single plate edge position reading taken in isolation may well be wildly inaccurate.

In previous work (5), using a similar logical neural network emulated on a general purpose computer, these inaccurate echo returns had to be caught and rejected. This proved to be a huge burden on computing resources and partly responsible for the low eventual classification rate of 4 Hz. The corollary of such a low classification rate is that many potentially useful echo-returns are lost, even as a poor one is being processed.

Using, TARDIS, however, the feature extraction compresses the data and makes it easier to manipulate, for averaging or other further processing. Figure 8 shows the PDF of 80 classification results. Ignoring zeroes (no classification), a modal average easily pinpoints the correct seam position.

The four stage pipelining and bipolar technology allows the network to classify at 10 MHz. Every new window arrangement can be classified in the time it takes a new one to be generated by the ADC. The network throughput is 1.28 Gbits/s. This classification rate is close to supercomputer performance. Figure 9 provides performance measures for some other classification techniques on various computing platforms. The values are for a single classification on an input window of 16 8-bit integer values, assuming digitisation at 10 MHz and an echo duration of around 1.5 us. None approach the raw classification speed of TARDIS.

A current drawback is the primitiveness of the feature extraction state machine. This is necessarily a simple device 'Figure 10', due to the speed at which it must operate. It merely stores a highest response, comparing it with every input summed response and keeping the higher value. These limited and inflexible computing abilities cause many drawbacks.

For one, it is highly susceptible to glitches caused by noise (which are not uncommon in such a large, fast and power hungry system), with no available error checking. Also, the situation of multiple highest responses could be dealt with in a more satisfactory fashion. At present, only the earliest highest response is recorded. Embedding a fast microcontroller in the place of the state machine would solve both of these problems, also simplifying the communication to a host weld control system.

CHARACTERISATION OF ARC SOUNDS

Manual welders

It has been established that skilled manual welders are able to monitor arc sound and alter electrode position in response to changes in audible process noise. The Design Department at Brunel University has been researching the possibilities of including this as an additional correcting factor in automated weld process control. Salient relationships between acoustic emissions and process current using off-line statistical techniques has been established, however, real time application remains problematic due to the computational intensity of such methods.

Investigations of acoustics in MIG using various flux cored wires has proved that by numerically integrating the sound data and plotting a spectrum of the result, close similarities occur with the voltage and current data with a 1 ms delay (9). Further manipulation with moving averages, has highlighted the pulse or dip modes of the process, however, limited success has been found in assessing the spray current mode. This research concludes with a suggested correlation between arc current and integrated acoustic signals and substantiates the use of acoustics for quality assessment of the welding process. However, a faster and more robust method is recommended to utilise this parameter for on-line analysis and control.

In GTA welding Kaskinen et al (10) have provided relationships in acoustic emissions and voltage / current parameters in pulse mode. In this mode, the pulsed power input to the weld creates a corresponding volume change to the arc plasma which, in turn, generates an alternating sound pressure level which is audible. The sound intensity proved to be proportional to the power input to the process, from which voltage and current values were derived. The voltage level provides information concerning the arc length and the research concludes with feasible arc length controller utilising acoustic information.

Fast Fourier Transforms were applied, with limited success, to CO_2 laser welding (11). Acoustic data was acquired via a microphone suitably amplified and processed in real time with a PC hosted digital signal processing (DSP) card to generate an FFT between 4 Hz and 100 kHz. Further processing reduced the frequency range to isolate salient bandwidths. The acoustic emission energy was derived from integrating the FFT spectrum over all frequencies and reproducible correlations found with laser power, welding speed and lens focal position.

The evidence for research into the use of acoustic emission analysis in submerged arc welding is limited. This can be attributed to three main reasons. The lack of visual information on which to validate the results. For example, expensive and often complex apparatus such as X-Ray and infra-red thermography is required to examine weld pool dynamics. Secondly, sound pressures are suppressed by the shielding flux which, at the same time, is responsible for creating sporadic transients as gases escape from the layer of molten slag. Furthermore, material transfer mechanisms
which are known to be partly responsible for changes in audible noise in welding processes (12) are limited to mainly slag protected transfer.

The effort evident in the area of acoustic emission analysis for welding processes,

focuses on the correlations with the fundamental process parameters. The use of acoustics as a valid aid to weld monitoring is debatable in some instances as the methods used to interpret them have revealed already known or more easily acquired metrics. However, the general conclusions call for more robust methods of signal processing and interpretation to yield more consistent and reliable measures of the process variables.

Consequently, the analysis of acoustic weld emissions using artificial neural network techniques, including Kohonen Self-organising feature map and Back Propagation methods, has been pursued.

Experimentation

Preliminary experimental investigations resulted in an optimised set up where an omni-directional electret condenser microphone (ECM) was mounted 300mm from the welding head in close proximity to suitably shielded pre-amplification. Typical time/amplitude sound pressure levels at varying voltage settings are illustrated in Figure 11.

The signals were then subjected to an active bandpass filter within the audible range and data manipulation achieved with the TMS320C30 system board hosting Hypersignal Workstation software. The transient capture sampling frequency was set to 20kHz and an on-line 128 point Radix 2 FFT instigated 'Figure12'. Initially weld runs were carried out with optimum parameter settings which were qualified by post weld inspection. The acquired time variant FFT frames were considered as 64 component input vectors for the ANN model. The data from the ideal weld was used as training data for a self organising neural network utilising a Kohonen learning algorithm (13) as illustrated by Figure 13.

This two layered network of fully connected neurons self organised a topological map, from a random starting point, a display of the natural relationships of the patterns used in the training data sets. This is achieved by the competitive activation of an output neuron dictated by the learning algorithm. The output neurons are arranged in a matrix so that each has theoretical neighbours. The activation threshold of neighbouring neurons are adjusted proportionally to the distance in vector space from the winning neuron. In this way each neuron is inhibited or encouraged to activate when presented with the next training set. The training continues until a state of equilibrium occurs or an acceptable level of accuracy is reached.

Instigating subsequent weld runs with various weld process parameter settings and analysing the activation levels of neurons in the output layer, differing patterns emerged indicative of the process state. Figure 13 show resultant three dimensional contour mappings of the network output layer highlighting the responses to changes in the process parameters of voltage, current and travel speed.

In this way the very erratic time variant FFT frames are compressed into an interpretable three dimensional representation of the process state. The generalising capabilities of the network enable further sets of FFT frames to be analysed in a more robust way. The number of output nodes of the self organising map relates to the resolution of the activation map as well as the robustness of the compression technique as each node represents a generalised version of an FFT frame. Too few nodes creates an overgeneralised representation whilst too many results in a slower response time. The optimum number has yet to be established, however, it is evident that the number necessary is dependent upon the welding process and configuration

being instigated.

The resultant three dimensional representations can be classified by use of further neural network models. In this instance both Kohonen Feature Maps and Backpropagation (14) methods were used. In the case of the back propagation model, a network was constructed to monitor the voltage. The activation map was used as the input with the corresponding voltage level as the desired output, the approach is shown in figure 14. The results of this model 'Figure 16' show a definite trend towards the actual monitored voltage with increased accuracy gained by lengthening the training times and extending the training set.

Figure 15 shows the approach for the feature map classifier. The activation maps were presented to the unsupervised learning model which grouped similar maps into regions on the output layer. These regions were then labelled by presenting known classified activation maps to the network. Recall was tested with unknown activation maps with encouraging results, however, more data is required to train a usable network in comparison to the backpropagation method. Furthermore, this network possessed a slower recall time.

Hardware Considerations

The application of neural networks entails a substantial amount of signal preprocessing to ensure network compatibility. The computational time for network recall is small in comparison to the necessary on-line DSP. Figure 17 shows the system setup. Data was initially collected on a PC hosted TMS320C30 DSP board. The off line training of the networks was carried out on a 486/66 PC and tested on this platform with prerecorded data. The final trained networks were embedded on a TMS320C32 50MHz DSP board hosted by a PC. The TMS series CPUs are tailored towards signal processing applications which rely on floating point and Multiply Accumulate Cycle (MAC) computation (15). Executing a neural network in recall is usually very calculation intensive. In the case of backpropagation methods each node requires the repetitive calculation of an inner product term to establish the nodal value or weight (16). The TMS320C32 is capable of a MAC rate of 20MHz which enables very fast real time recall for this application.

Conclusions

The research presented in this paper suggest a method to interpret the erratic signals obtained from airborne acoustics from the submerged arc welding process. The use of neural networks for audible sound analysis in SAW is possible at real time rates although the information so far gained relates only to already known metrics. Future work will entail the use of these techniques to assess the stability of transfer mechanisms within the process.

REFERENCES

1 BOILLOT, J.P. AND COTE, J.L. " A modern adaptive robotic welding system". In Proc.4th Int. Conf. on Computer Technology in Welding, June 1992, Cam bridge, UK.

2 STROUD, R.R.," Problems and observations whilst dynamically monitoring molten weld pools using ultrasound", British Journal of NDT. Vol. 31, No.1. 1989.

3 FENN, R.," Ultrasonic monitoring and control during arc welding", Welding Journal. Sept. 1985.

4 DUFFILL, C., HAYWOOD, B.C.G., SCRUBY, C.B. AND STARES, I.J., "On-line assessment of weld quality using ultrasonics", United Kingdom Atomic EnergyAuthority Report AERE R 13401, July, 1989.

5 SWALLOW, S.S., " TARDIS". Thesis submitted for the degree of Doctor of Philosophy.Dept. of Design, Brunel University. UK.

6 BLEDSOE, W.W. AND BROWNING, I., " Pattern recognition and reading by machine." In Proc. Eastern Joint Computing Conf. 1959.

7 SMITH, G., " Fast image preprocessing with Ntupling.", Proc. Int. Conf. on Artifi cial Neural Networks. Sorrento, Italy. May 1994.

8 HARRIS, T.J., STROUD, R.R. AND TAYLOR-BURGE, K. , " Ultrasonic weld control using artificial intelligence (neural networks).", Proc. JOM-5. May 1991. Helsingor, Demark.

9 CHAWLA, K., " Objective on line assessment of the performance of flux cored wires by real time computer based monitoring.", Ph.D Thesis, Cranfield Institute of Technology, UK, 1993.

10 KASKINEN, P. et al, "Acoustic Arc Length Control.", Advances in Welding Technology and Science, TWR'86, pp 763-765.

11 MAO, Y.L et al, " Real-Time Fast Fourier Transform Analysis of Acoustic Emission During CO_2 Laser Welding of Materials.", Journal of Laser Applications, Vol 5, No 2 & 3, Autumn 1993.

12 ERDMANN-JESNITZER, et al, "Acoustic Investigations of the Welding Arc.", Report Document 212- 86-66, Institute of Metals, Hannover.

13 DAYHOFF, J.E., "Neural Network Architectures, an introduction", Van Nostrand Reinhold, 1991.

14 RUMELHART, D.E., MCCLELLAND, J.L., "Parallel Distributed Processing", Vols 1&2, MIT Press, 1986.

15 TMS320C3X Users Guide, Texas Instruments, 1994

16 LEE, D.G. Jr. "Hardware Implimentations" in, " Neural Network PC Tools, a practical guide", Eberhart, R.C, Dobbins R.W, Academic Press Inc, 1990.

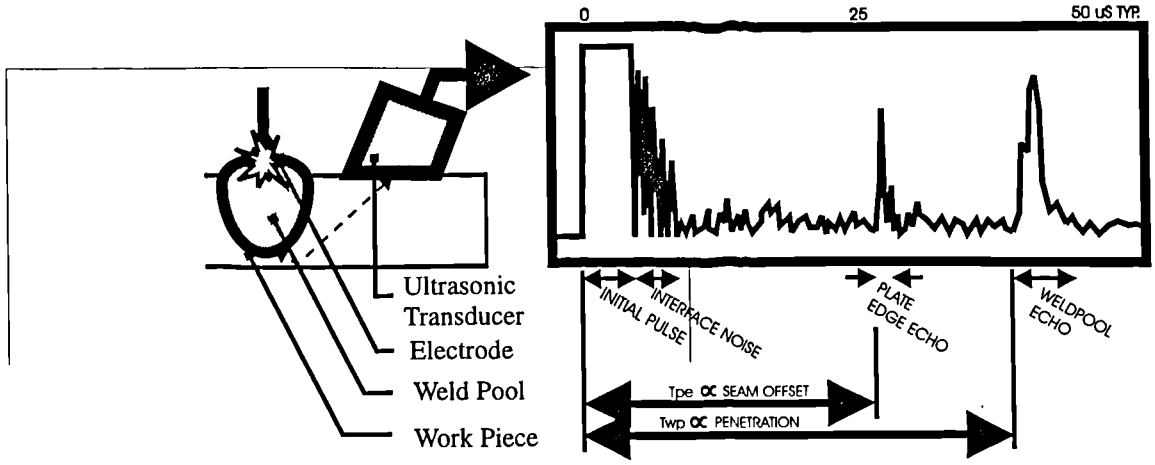


FIGURE 1. ON-LINE ULTRASONIC MONITORING

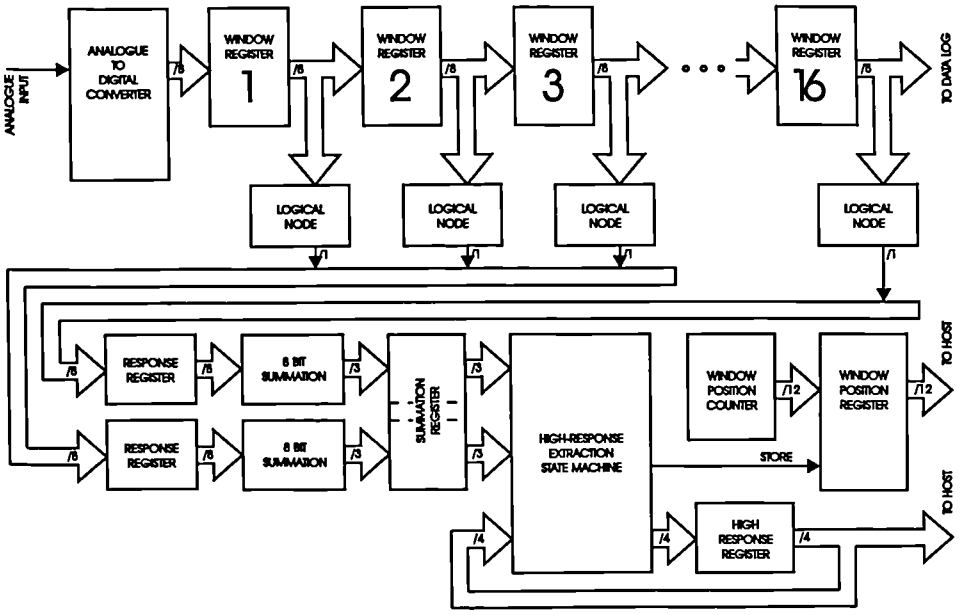


FIGURE 2. TARDIS HARDWARE ARCHITECTURE

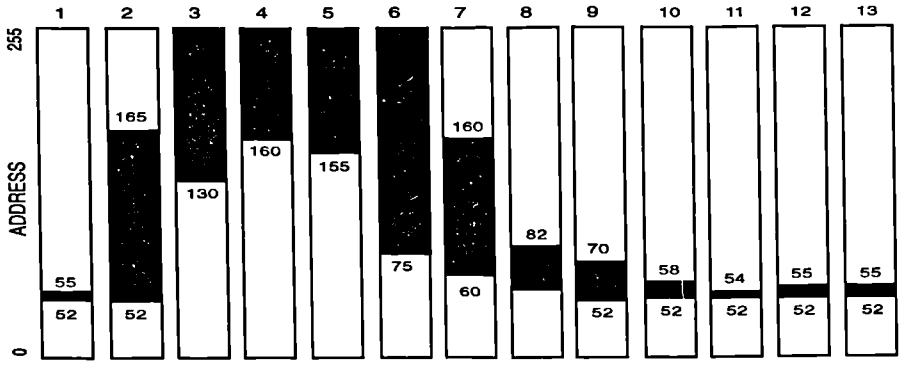


FIGURE 3.PLATE EDGE RAM-NODE TEMPLATES

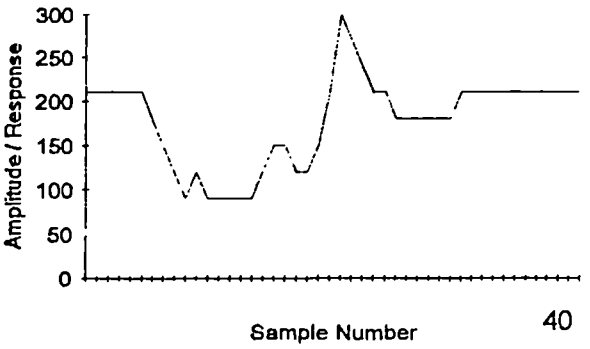


FIGURE 4. PLATE EDGE CLASSIFICATION

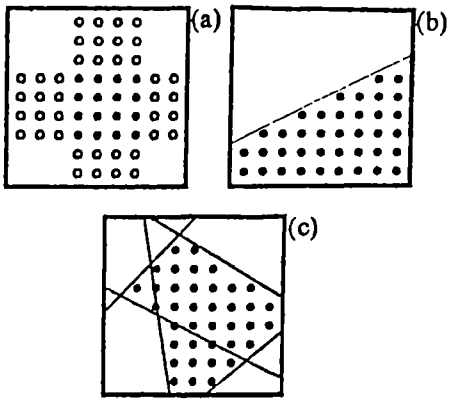


FIGURE 5.
PATTERN SPACES
(a) TARDIS (b) LDF (c) PIECEWISE

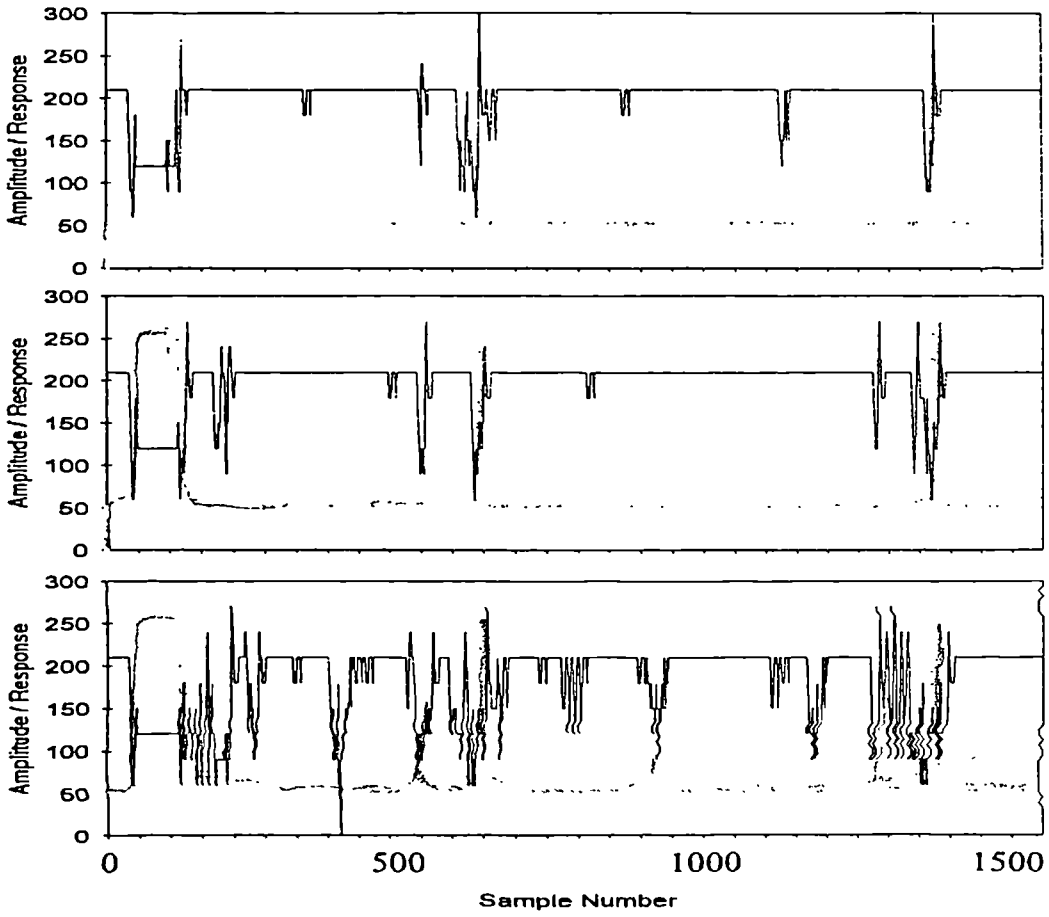


FIGURE 6. TARDIS CLASSIFICATION
OVER COMPLETE ECHO RETURNS

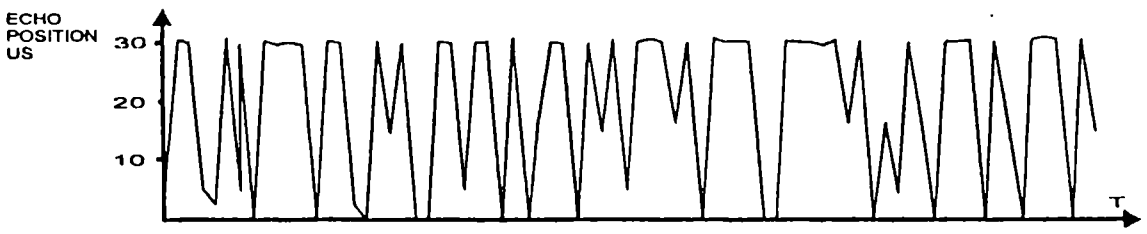


FIGURE 7. CLASSIFIED PLATE-EDGE ECHO LOCATION OVER SUCCESSIVE RUNS

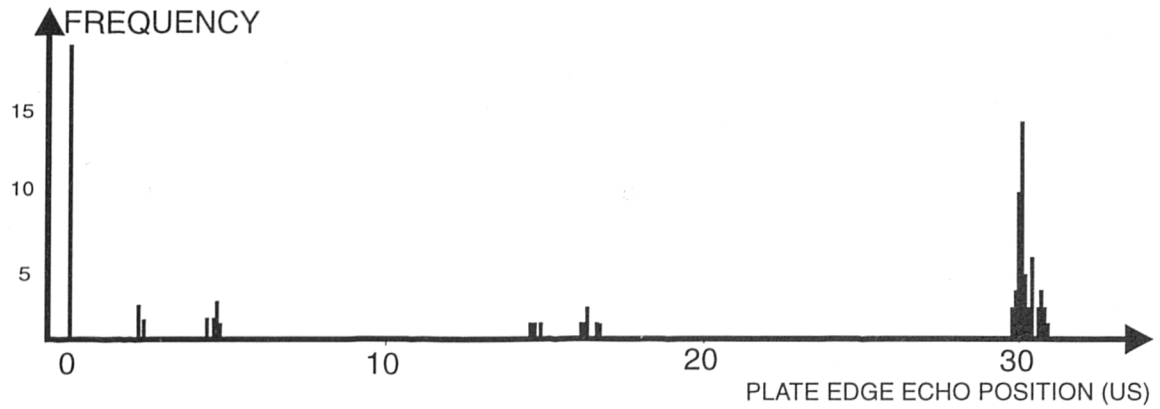


FIGURE 8. PDF OF PLATE EDGE ECHO POSITION

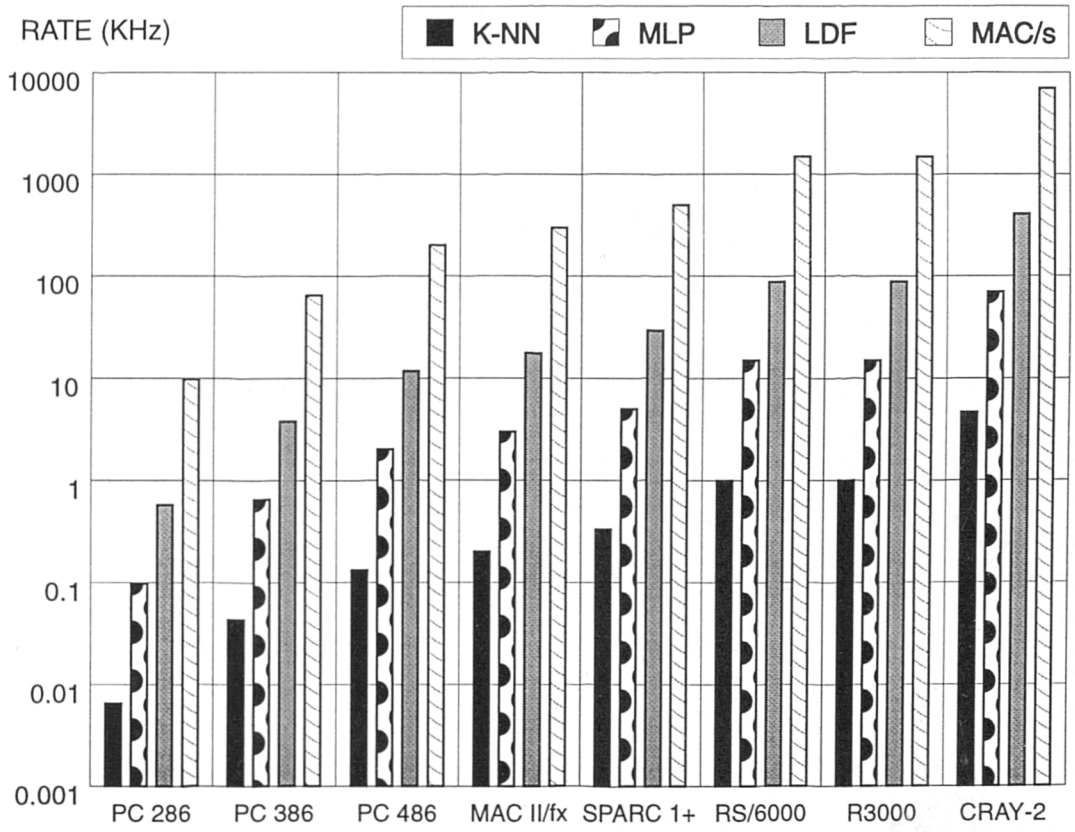


FIGURE 9. PERFORMANCE OF MAC CLASSIFIERS ON GENERAL PURPOSE PLATFORMS

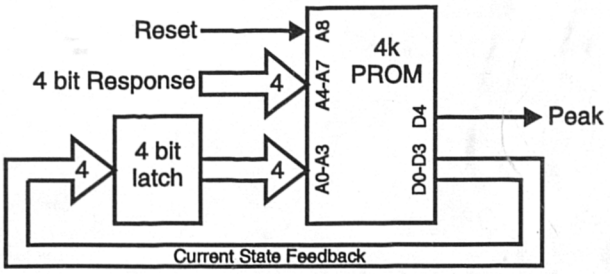


FIGURE 10. HIGH RESPONSE EXTACTION STATE MACHINE

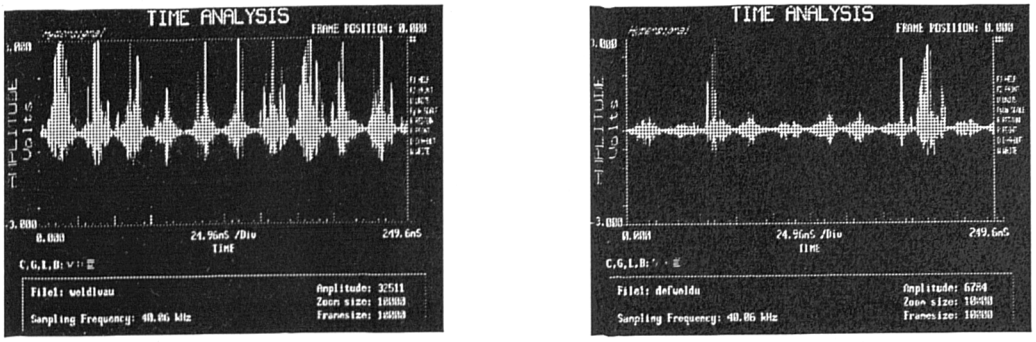


Figure 11. Typical Time/Amplitude Signals, Sound Pressure Level

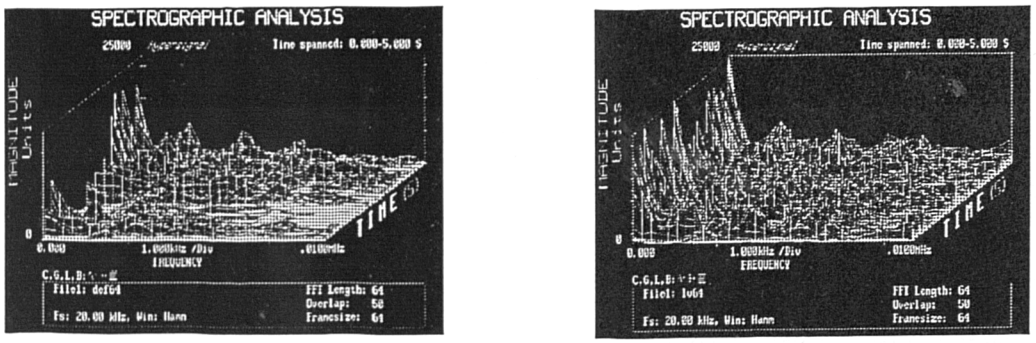


Figure 12. 6th Order Time Variant FFT

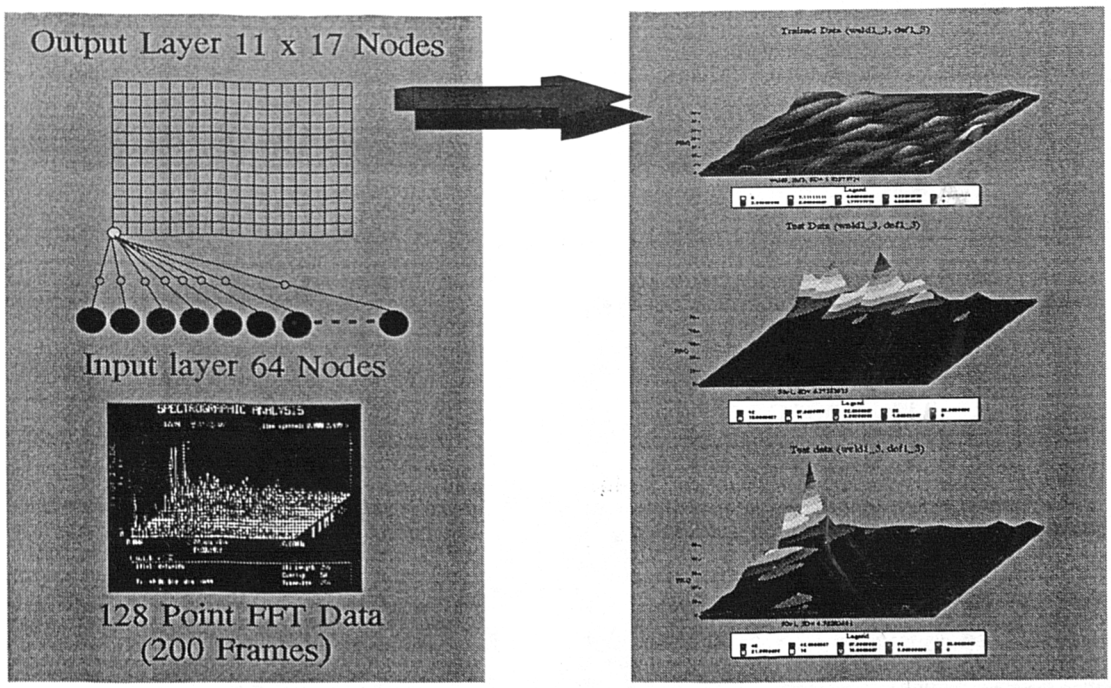


Figure 13. Feature Map Training and Recall Activations

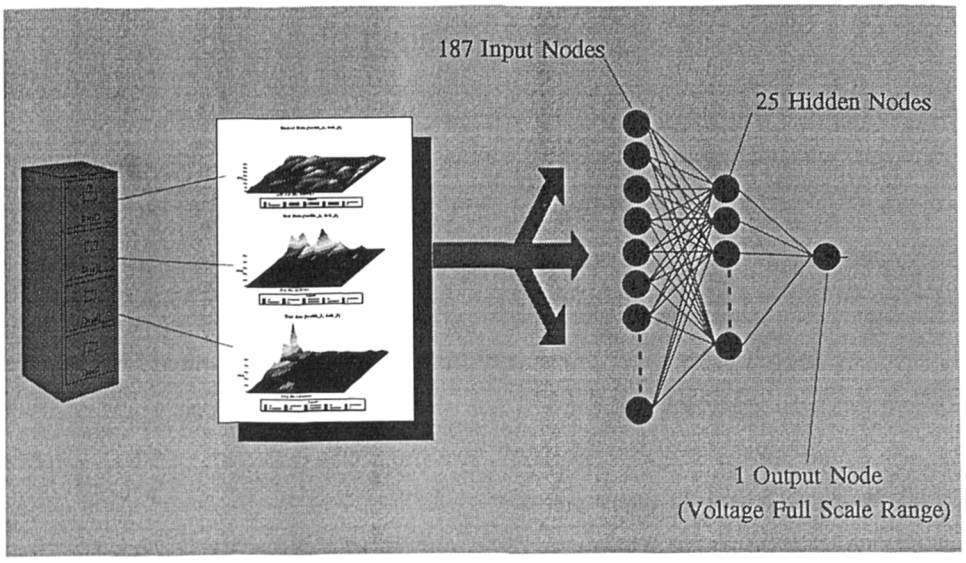


Figure 14. Backpropagation Activation Map Classifier

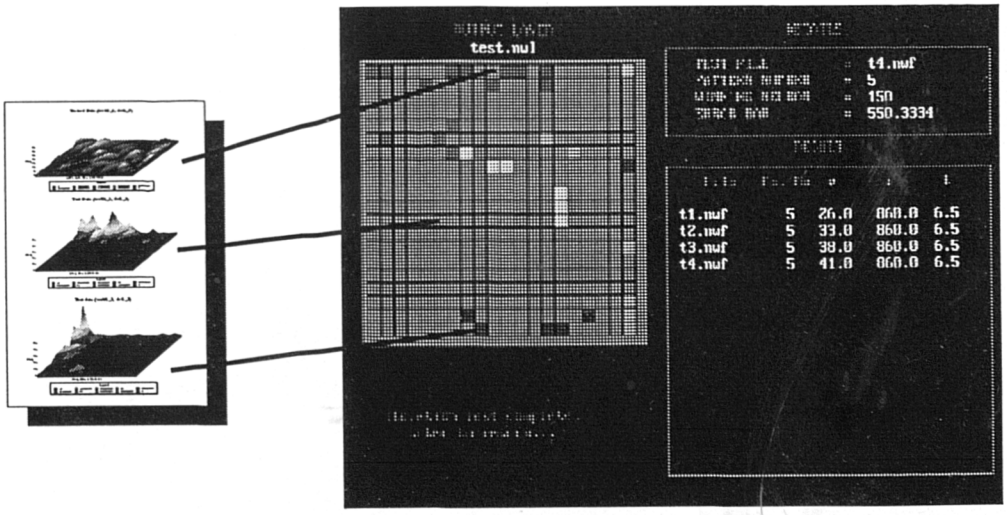


Figure 15. Labelled Feature Map Classifier

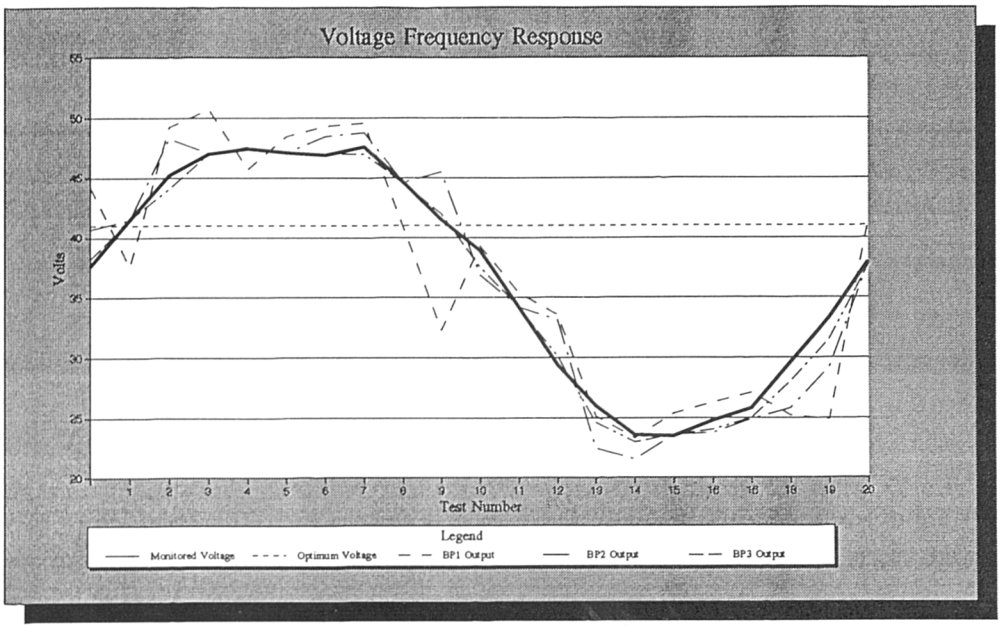


Figure 16. Response from Backpropagation Model to Monitored Voltage
(Accuracy improvement gained with increased training set)

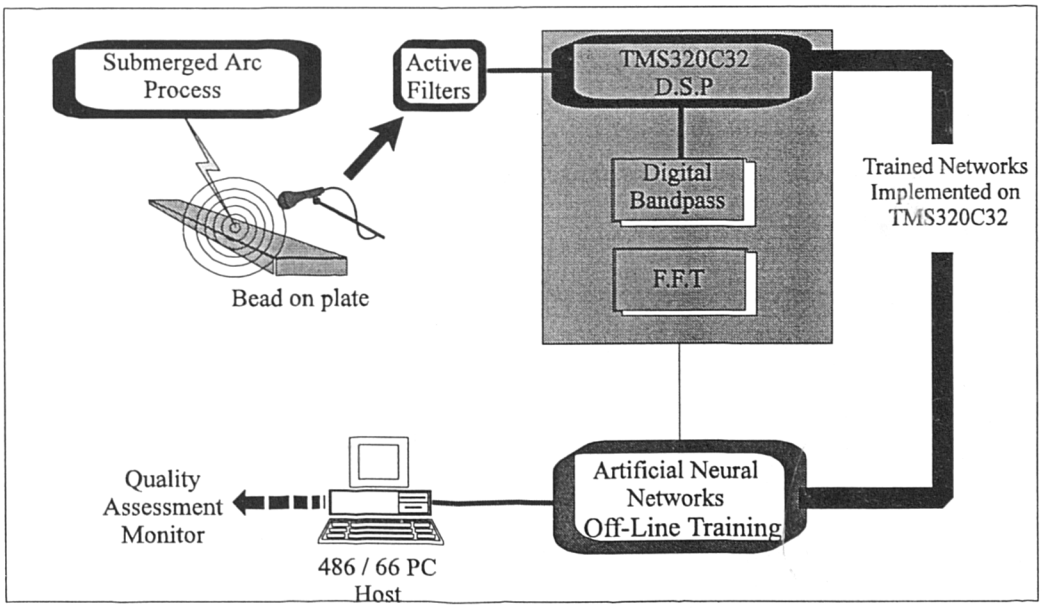


Figure 17. System Setup and Hardware Requirements