CRYSTAL FILTER TUNING USING MACHINE LEARNING

DIMITRIS TSAPTSINOS

A thesis submitted in partial fulfilment of the requirements of the Council for National Academic Awards for the degree of Doctor of Philosophy

March 1992

Sheffield City Polytechnic in collaboration with Newmarket Microsystems

Acknowledgements

I am vastly indebted to many people who have helped and encouraged me, in various ways, during this research work. To all of them, I want to express my sincere gratitude.

First and foremost, my gratitude goes to Barrie Jervis for giving me the opportunity to become involved in the world of Artificial Intelligence, and for his continuous support and strengthening of my resolve in so many direct and indirect ways.

Ahmad Mirzai, for sharing his experiences with filter tuning, and for making me aware of the potential of neural networks.

Lawrence Mack, for the many hours spent with me in front of the measuring equipment, and for his programming assistance.

Rick Osborn who read earlier drafts of the thesis and contributed remarks, corrections and suggestions which substantially improved the thesis.

Stephen Wadsworth for arranging the collaboration with Newmarket Microsystems, and the National Advisory Board for financing the project.

The help of Anna Hart in the beginning of the research work concerning the ID3 implementation is truly appreciated.

A special thank you goes to the secretarial staff and to the school football team for making my stay at the School of Engineering Information Technology so enjoyable.

Finally, to my mother, Marjon and family goes my eternal gratitude for their love, support, and dedication.

Abstract

Manual tuning of electronic filters represents a time-consuming process which can benefit from some computer assistance. A prototype computer-based system for the tuning of crystal filters after manufacture was developed. This system solved the problem of crystal filter tuning in a novel way.

The system, called AEK (Applied Expert Knowledge), was developed using crystal filters and is a hybrid system with the following two functions:
(1) Required values of features are extracted from the filter waveform and passed to the expert system which determines the component to adjust and the direction to turn, or the end of the tuning.
(2) Sampled values of the waveform are extracted and passed to a neural network which determines how far to turn the component chosen in (1).

The prominent aspects were:
- Work using the protocol analysis elicitation technique indicated the need to separate the process into two sub-tasks (stopband and passband). Each sub-task was divided into three classification parts which determined (i) the continuation of the tuning process, (ii) the component and direction to turn, and (iii) the distance to turn respectively. Unfortunately, it was not possible to extract rules from the operator.
- Three learning techniques (ID3, Adaptive Combiners, Neural Networks) were used and compared as the means of automated knowledge elicitation. All three techniques used case knowledge in the form of examples. The investigations suggested the use of ID3 for the first two parts of each sub-task employing features with linguistic values. The number of linguistic values each feature has, was also derived.
- Neural networks were trained for the third part. It was necessary to have one network for each component/direction combination and to use examples from just one mal-adjusting process.
- Tests of the hybrid system for a number of cases indicated that it performed as well as a skilled operator, and that it can be used by novice operators but situations arose where there was either no knowledge or contradictory knowledge.

The prototype system was developed using one type of crystal filters but the generic construction procedure can be followed to build other systems for other types.

Table of Contents

Appendices

List of Tables

List of Figures

List of publications

1. Tsaptsinos D., Mirzai A.R., and Jervis B.W., *Comparison of machine learning paradigms in a classification task*, Proceedings of the Fifth International Conference of Artificial Intelligence in Engineering, Vol. 2, Computational Mechanics Publications, pp. 323-339, 1990.

2. Tsaptsinos D., Jervis B.W., and Mirzai A.R., *Learning by analytical methods or induction - a case study*, IEE Colloquium on Machine Learning, Digest No. 117, pp. 10/1-10/3, 1990.

3. Tsaptsinos D., Mirzai A.R., Jervis B.W., and Cowan C.F.N., *Comparison of knowledge elicitation techniques in the domain of electronic filter tuning*, IEE Proceedings-F, Vol. 137, No. 5, pp. 337-344, 1990.

4. Tsaptsinos D., Jervis B.W., Mirzai A.R., *Practical aspects of using an expert system-neural network hybrid system for tuning crystal filters*, IEE Second International Conference on Artificial Neural Networks, pp. 314-317, 1991.

Organisation of the thesis

The thesis is divided into two main parts. Part A consists of two chapters devoted to the concepts of electrical filters and artificial intelligence. The first chapter looks at crystal filters and the problems of post-assembly tuning. Furthermore an overview of expert system components and a review of knowledge elicitation and representation is given followed by a discussion in terms of the system constructed. The second chapter covers the procedure used at the collaborating establishment and the procedures proposed by other workers in the field. The chapter ends with a discussion about the motives for employing an expert system approach in the filter tuning domain.

Part B of the thesis has seven parts, which present a chronicle of the expert system and the neural networks development. Part B represents original work undertaken during the development of the AEK system. Chapter three presents the knowledge acquired during the first visit to the company and identifies the reasons for moving to the machine learning paradigm. Chapter four introduces a number of techniques which can be employed for the design of learning systems. These techniques include ID3, adaptive combiners and neural networks. Chapter five shows the adaption and comparison of the techniques presented in the previous chapter for the filter tuning application. Chapter six highlights the problems encountered when using ID3 and describes additional work undertaken to avoid the shortcomings of the technique. In the seventh and eighth chapter the induction of rules and the construction of the neural networks are presented respectively. Chapter nine presents the software and hardware employed together with instructions of how to use the AEK system. Chapter ten is entirely devoted to the application

of the rules and the networks to the tuning of a number of filters. This is followed by a detailed evaluation of their performance.

Finally, the thesis comes to a closure (chapter 11) with a discussion of the achievements, a critique of the expert and neural systems and an assessment of the software used.

1.1 Introduction

This chapter discusses general aspects of electrical filters with particular reference to the crystal filter used in the study. The requirement for filter tuning is justified in Section 1.2.5 which also introduces the methods employed.

Sections 1.3 to 1.4.5 provide an overview of artificial intelligence and expert systems. This is followed by an overview of the expert system constructed for this study in terms of knowledge representation and control. AEK (Applied Expert Knowledge) is the name given to the system and it is used throughout this thesis.

1.2 Introduction to electrical filters

An electrical wave filter, or just filter for ease of reference, is designed to receive a signal and to attenuate certain pre-defined frequency regions of the input signal while passing the rest of the frequency regions without changes. It is possible to classify filters in different ways[1]. In terms of the frequency spectrum, they may be grouped as audio, video, or radio-frequency and microwave filters. In terms of the circuit configuration of the basic elements, filters may be classified as ladder or lattice. Categorization in terms of the character of the elements used in them is also common, for example LC or RC filters. The most customary division is between analogue and digital filters which treat analogue and digital signals respectively. Analogue filters may be classified as passive or active. These constructions are similar except that the latter has an integral source of energy, usually an operational amplifier. Digital filters on the other hand utilise software, such as a subroutine on a

computer program, or as hardware, such as a circuit containing registers and multipliers.

1.2.1 Filter components

Electrical filters contain a variety of components[2] and it is the responsibility of the designer to select the appropriate components for any given task. Filter components come in two forms, namely active and passive. Active elements may amplify the signal power. By contrast passive elements do not contribute to signal energy; they can only absorb or transfer it. Capacitors and inductors are two common passive elements.

1.2.2 Magnitude responses and approximations

One way for studying any filter is to investigate the magnitude response of the output signal. The output signal is the product of the magnitudes of the input signal and the frequency response function of the filter. This means that if the magnitude of frequency response is equal to zero (or approximately equal to zero) for a certain frequency range, then the output signal will have a zero (or approximately zero) magnitude over this frequency band. This group of frequencies is called the stopband of the filter. Similarly, if the magnitude function is greater than zero and close to one for another frequency band, then this interval is called the passband of the filter. In addition, the band of frequencies between a passband and a stopband is defined as the transition band. Certain frequency bands are then transmitted while the rest are rejected. The design of each filter determines the regions, if any, where frequency is allowed to pass or not and provides yet another

taxonomy. They can be either lowpass, highpass, bandpass, or bandstop filters. Lowpass and highpass filters are, respectively, filters that transmit signals at frequencies below or above a defined cut-off frequency (ω_c) and attenuate those frequencies above or below the cut-off point (ω_c). Bandpass filters transmit all frequencies between defined upper (ω_2) and lower limits (ω_1), and attenuate frequencies outside those limits. Bandstop filters attenuate frequencies between upper (ω_2) and lower limits (ω_1) and transmit all other frequencies. These four basic types of frequency selective filters are illustrated in Figure 1. Of course, there are filters that do not belong to any of these four types but in most cases the magnitude specification of filters will fall into one of those categories. In practice, these characteristics are not attained with a finite number of components due to absorption, reflection or radiation, so a number of well known curves, which approximate the ideal responses within specified tolerances, are used. The common filter approximations are the Butterworth, Chebyshev, inverse Chebyshev, and elliptic[3] (Figure 2a).

1.2.3 Crystal filters

A crystal, physically, is a three dimensional pattern consisting of atoms, molecules, or ions[4]. A variety of classes of crystals exist of which about twenty exhibit the desired effect of piezoelectricity[4]. Piezoelectricity refers to the electric potential being generated whenever an external pressure is applied to the crystal. Crystals exhibit mechanical resonance which can be excited by the application of an AC signal. The size and shape of the crystal determine the frequency of the mechanical resonance which typically varies from 20 KHz to 50 MHz. Figure 2b shows the electrical equivalent circuit. L_1, C_1 and

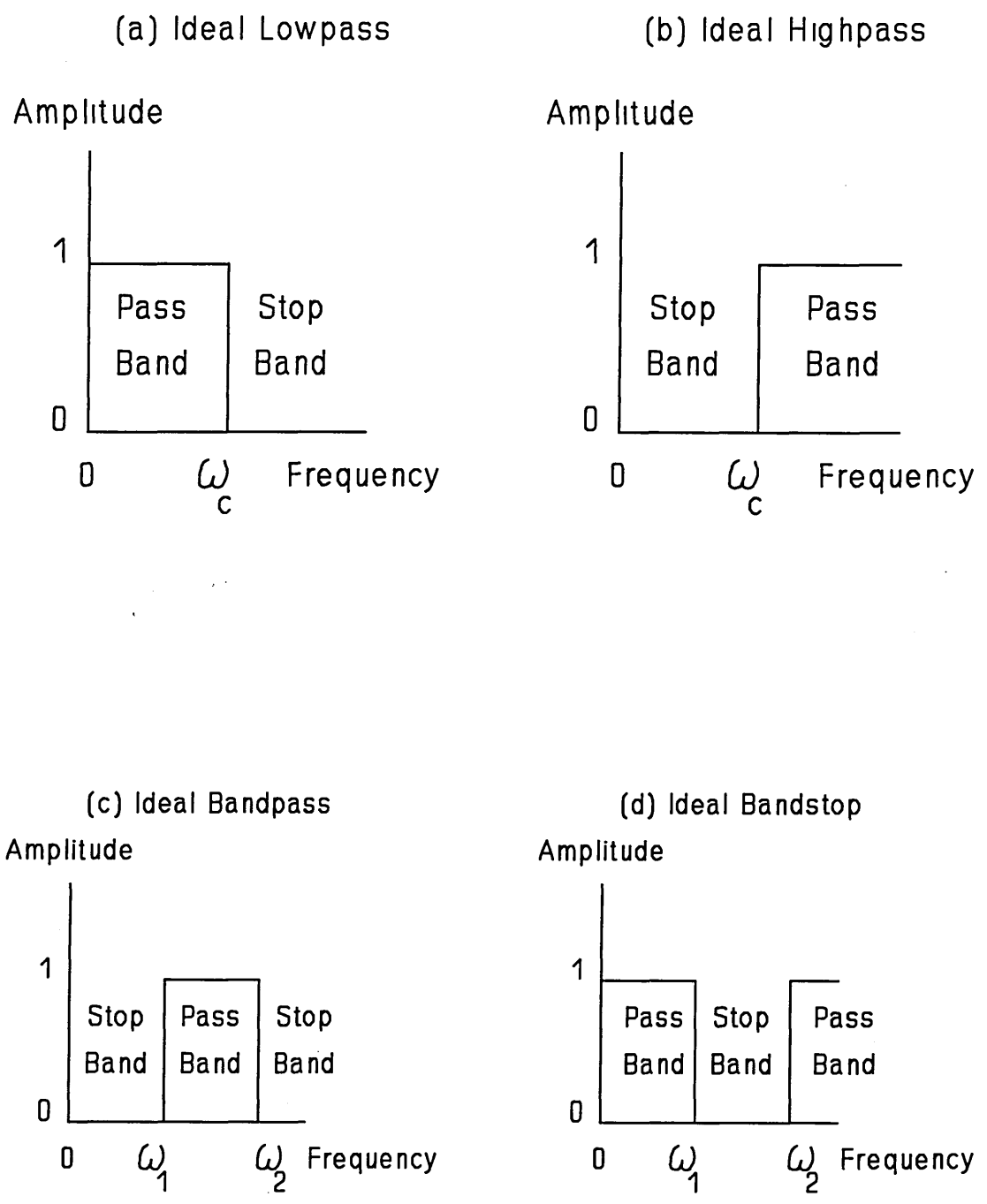


Figure 1: The ideal filter amplitude characteristics

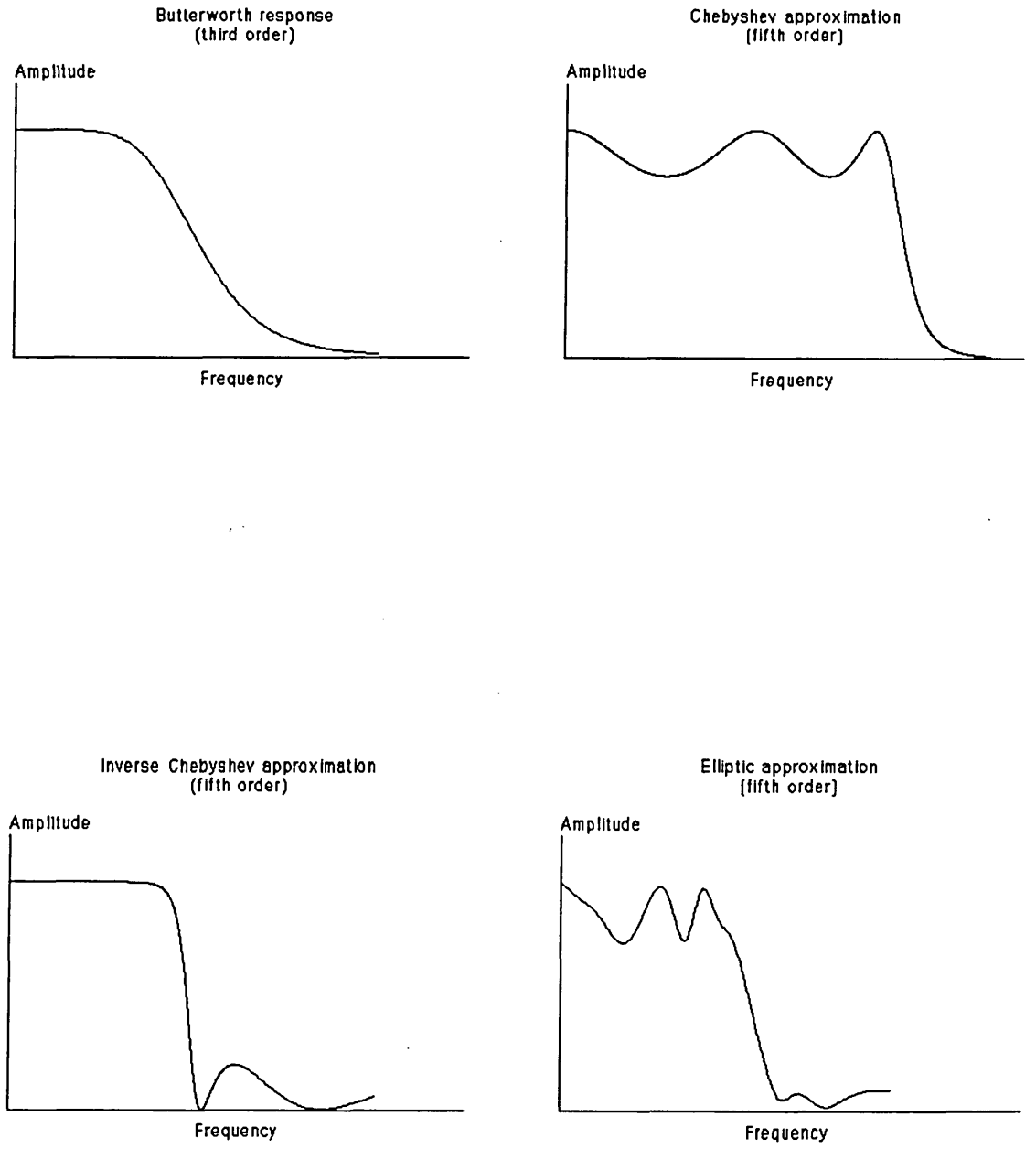


Figure 2a: Typical responses of low pass filters using approximations

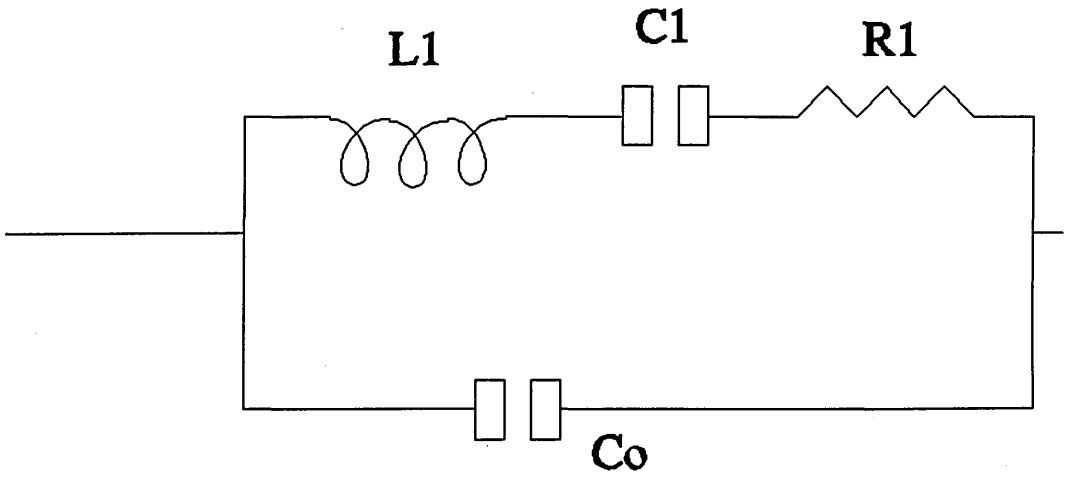
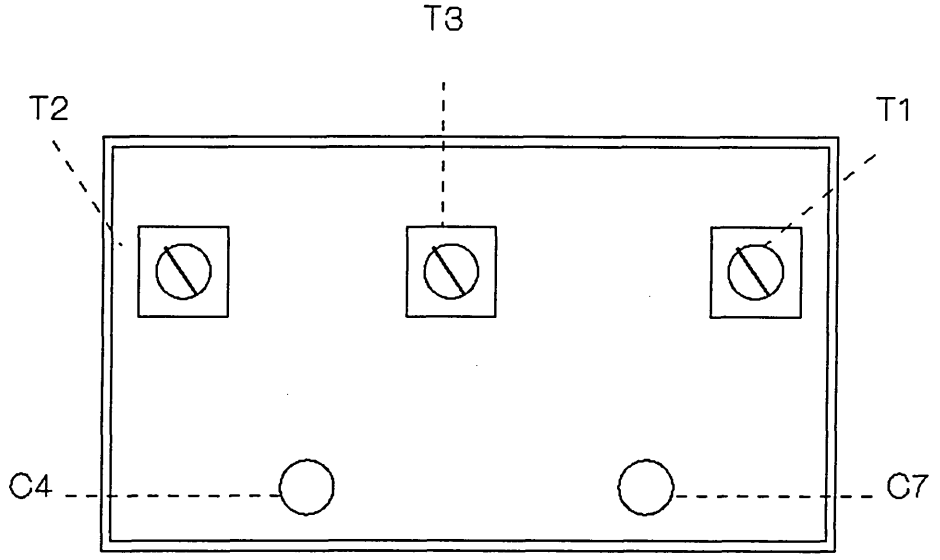


Figure 2b : Equivalent circuit of crystal resonator

R_1 are the motional parameters and C_o is known as the static capacitance and represents the effective capacitance of the crystal unit at frequencies far removed from resonance. The quality factor of the motional arm is extremely high, typical values are between 20000 to several hundred thousand, compared to other resonators, such as LC circuits. The quality factor and the unique combination of properties (stability with time and temperature, high quality factor, strength, inexpensive, small size, low insertion loss) make crystals attractive and provide flexibility for the practical design of filters with very narrow bandwidth. The term crystal filter is used to describe electrical filters incorporating crystal resonators. The principal crystal used in electrical filters, especially bandpass filters, is the quartz crystal. Theoretically, an electric circuit using inductors, capacitors and resistors can be constructed to simulate a crystal resonator but the problem lies with the practicality of obtaining the exact values for these components. Crystal filters can be either discrete or monolithic[5,6]. The former employ standard components plus a number of single crystal resonators. In comparison monolithic crystal filters provide a complete filter on a single quartz wafer with no supplementary parts.

1.2.4 The benchmark filter

The collaborating establishment produces about two hundred separate types of crystal filters. The filter code number 4716 was used for this study. This filter is a discrete 4-pole asymmetric bandpass crystal filter. Asymmetric refers to the passband region because of the steep skirt selectivity on one side of the passband and the reduced attenuation on the opposite side. Figure 3 displays the top view of the filter. The filter consists of two types of



<u>Key</u>

C_4, C_7 : trimmer capacitors

T_1, T_2, T_3 : inductors

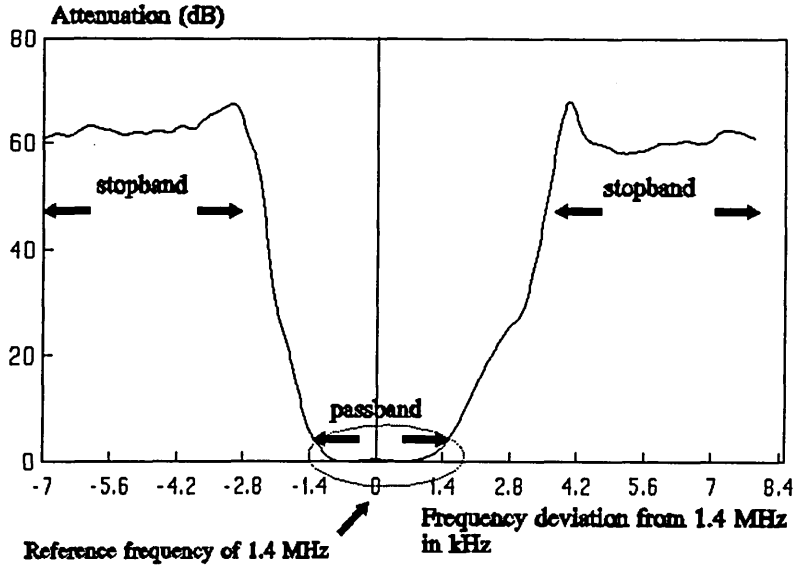
Figure 3: Top view of filter used in the study

adjustable components, namely trimmer capacitors (C_4, C_7) and inductors (T_1, T_2, T_3). The specification of the filter is summarised in Table 1. The selectivity requirements are divided into two general areas, namely the passband and the stopband response regions. Both regions are specified with reference to a nominal frequency which is the centre frequency (reference frequency). A typical filter response demonstrating the electrical specifications is shown in Figure 4.

1.2.5 The need for post assembly tuning

Filter engineers have tackled the tuning problem in two different ways. One approach takes place during the design stage and the other takes place after assembly. The post-assembly approach can be further categorised into two methods, namely functional and deterministic[7]. The latter method applies circuit modelling and includes techniques such as response sensitivity. This research concentrated on the former method. This is the traditional approach in which tuning is performed manually. The manual tuning procedure is described in Section 2.2.

In practice, the actual performance of an electrical filter differs from the specification. This is due to the inescapable effects of using real components which leads to apparently identical filters having slightly different responses. This becomes more transparent when, for example, the inductor component is considered. The use of inductors is a predominant cause of response deterioration, because obtaining exact values requires winding the component by hand. This results in inconvenience and further cost. Furthermore the method used in winding the coil, number of windings, spacing of turns, permeability of the core are all factors that contribute to the electrical



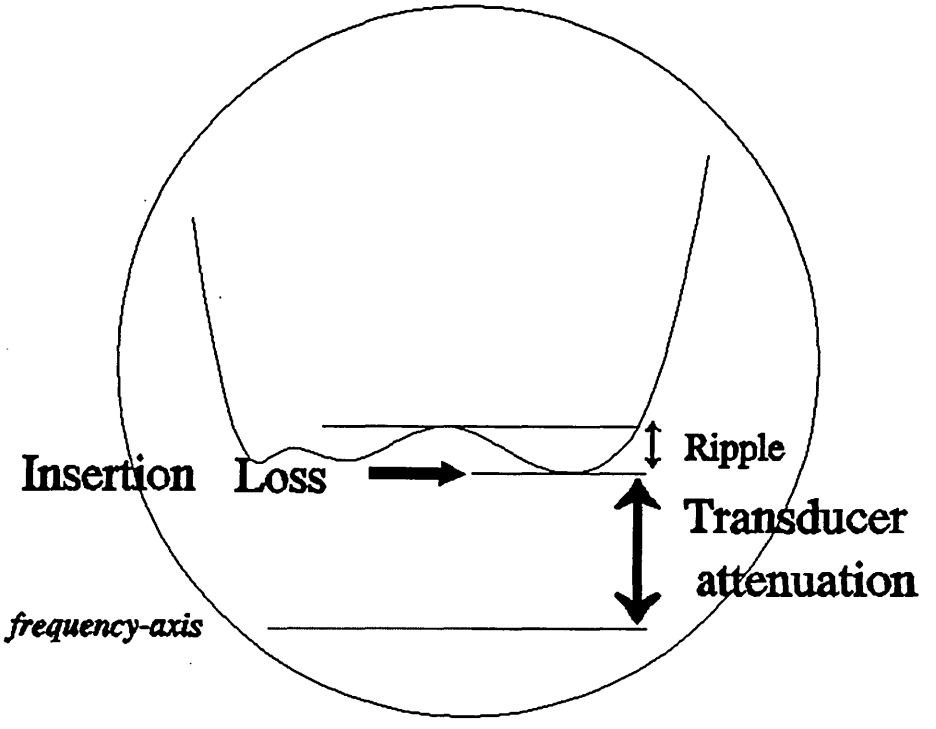


Figure 4: Typical Magnitude response showing the filter specification

Table 1 : Specification of used filter	
Filter type	Upper Side Band
Reference frequency	1.4 MHz
Passband width	+0.5 KHz to +2.5 KHz minimum at 4 dB
Stopband width	-0.7 KHz to +5.0 KHz maximum at 45 db
Passband ripple	3.0 dB maximum +800 Hz to +2.0 KHz
Transducer attenuation	5.0 dB maximum 0.5 dB minimum
Attenuation at 1.4 MHz	10 dB minimum
Ultimate attenuation	45 dB minimum to be maintained to ± 20 KHz
Termination impedance	1 kΩ ± 15 /// 75 ± 10 pF
Maximum input powers	1 mW
Operating temperature range	-10°C to +65°C
Maximum weight	75 gms

characteristics of an inductor.

1.3 Overview of Artificial Intelligence

In the literature of computer science the task of exploring and simulating human intelligence has been termed Artificial Intelligence. The objectives are twofold:

(i) the amplification of the user's capability in performing intelligent tasks, and

(ii) the understanding of the principles of intelligence.

One representative definition of Artificial Intelligence is given by Barr and Feigenbaum[8].

'Artificial intelligence is the part of computer science concerned with designing intelligent computer systems, that is systems that exhibit the characteristics we associate with intelligence in human behaviour - understanding language, reasoning, solving problems and so on.'(page 4)

Therefore Artificial Intelligence is based upon perceptions of human intelligence. Although we can recognize intelligence it is questionable that anyone could provide a definition covering all its aspects. The spread of the interpretation of the term intelligence has resulted in the discipline of Artificial Intelligence incorporating the fields of engineering, cognitive science, philosophy, psychology and linguistics. This generated applications and topics of research. Some examples of application areas are game playing, automated reasoning and theorem proving, natural language understanding, robotics, expert systems, machine learning and neural computing. The work performed for the tuning of electrical filters involved the expert systems, machine learning and neural computing branches. These areas are explained at the

appropriate sections of the thesis. The reader is referred to books by Winston[9], Charniak and McDermott[10], and Barr and Feigenbaum[11,12,13] for further background to the theory and pragmatics of Artificial Intelligence.

1.4 Expert Systems

The realization by the Artificial Intelligence community during the 1960's of the weakness of general purpose problem solvers led to the development of expert systems. Expert systems held the greatest promise for capturing intelligence and have received more attention than any other sub-discipline of Artificial Intelligence. The term knowledge-based systems is used interchangeably to avoid the mis-understandings and mis-interpretations of the word 'expert'. Irrespective of the adjective, each such system is designed to operate in one of a variety of narrow areas. The design involves attempts to model and codify the knowledge of human experts.

1.4.1 A review and classification of expert system projects

The number of expert systems reported in journals is rapidly increasing. But there are four examples that merit special attention due to the fact that they were the pioneering attempts. These systems are, the Dendral[14] system which infers the molecular structure of complex organic compounds from their chemical formulae and mass spectrograms, the Mycin[15] system which diagnoses blood infections and recommends the appropriate drug treatment, the Prospector[16] system which is designed to aid geologists in their search for ore deposits, and the R1(XCON)[17] system whose purpose is to configure VAX-11 computer systems. These systems are important. First they showed that

the new technology can work and secondly they provided models (of representation and inference) that other implementations followed. Since that era a wide variety of programs, not so much acclaimed, have been developed in many different fields, performing a range of diverse tasks. For a survey of recent applications, and a set of references, see Bramer[18], Reddy[19], Bremer[20] and IEEE Computer[21]. There are numerous ways to classify expert systems but the two that follow are probably the most important. One apparent practice is by their area of application (Mycin - medical, Prospector - geology). The other is by the tasks that they are called upon to perform[22] (Dendral - interpretation, Mycin - diagnosis).

1.4.2 The components of an expert system

The essential components of an expert system can be identified as :-

Knowledge-base module: this is the essential component of any system. It contains a representation in a variety of forms of knowledge elicited from a human expert (see Section 1.4.3).

Inference engine module: the inference engine utilises the contents of the knowledge base in conjunction with the data given by the user in order to achieve a conclusion.

Working memory module: this is where the user's responses and the system's conclusions for each session are temporarily stored.

Explanation module: this is an important aspect of an expert system. Answers from a computer are rarely accepted unquestioningly. This is particularly true for responses from an expert system. Any system must be able to explain how it reached its conclusions and why it has not reached a particular result.

Justification module: using this module the system provides the user with justification(s) of why some piece of information is required.

User interface module: the user of an expert system asks questions, enters data, examines the reasoning etc. The input-output interface, using menus or restricted language, enables the user to communicate with the system in a simple and uncomplicated way.

Through the years systems have appeared which include additional modules. For example, learning modules, knowledge acquisition modules and refinement modules. Each one of the above constitutes a research topic on its own.

1.4.3 The nature and representation of knowledge

Whereas from a philosophical point the concept of knowledge is highly ambiguous and debatable, expert system builders (to be referred to as knowledge engineers) treat knowledge from a narrower point of view. This way the knowledge is easier to model and understand, but remains diverse including rules, facts, truths, reasons, defaults and heuristics. The knowledge engineer needs some technique for capturing what is known about the application. The technique should provide expressive adequacy and notational efficacy[23]. Knowledge representation is very much under constant research and several schemes have been suggested in the literature. The four most widely used in current expert systems are production rules[24], semantic nets[25], frames[26], and logic[27]. Obviously, no single method can represent all kinds of knowledge and although some kinds of knowledge can be represented in many

ways some other kinds of knowledge, such as time, cannot be captured.

1.4.4 Controlling the knowledge

Much of the power of an expert system comes from the knowledge embedded in it. In addition, the way the system infers conclusions is of equal importance. The knowledge engineer has to consider how to implement the control, ie. what to do next, and the search, ie. how to find some information. These decisions rely on the classification of the task[28], and on the amount of information known beforehand about the problem space. Various problem-solving methods have been described in the literature[29].

1.4.5 Knowledge acquisition and elicitation

The terms knowledge acquisition and knowledge elicitation are often confused. The knowledge acquisition process is defined as the combined activity of eliciting, analyzing, interpreting, representing, administering and utilising the knowledge of human experts. Clearly, knowledge elicitation only address the *elicitation* aspect of the task. The primary activity during elicitation is to capture knowledge from experts through a series of sessions. A large number of elicitation techniques have been proposed as suitable and as a result of a literature review, the following techniques were identified:

Structured interview	Questionnaires
Interruption	Retrospective comment analysis
Behaviourial observation	Informal interview
Protocol analysis	Multidimensional scaling
Concept sorting	Repertory grid
Cluster analysis	Socratic dialogue
Forward scenario	Conceptual clustering

It is important to realise that generally none of these techniques can surface on its own but a mixture will probably obtain the required results. The reason

for this being that knowledge has many forms and each technique can only attempt to extract a subset. For example the protocol analysis technique, described below, works very badly for domains which are best represented declaratively but a rich amount of procedural knowledge can arise.

1.4.5.1 Protocol analysis

Protocol analysis (or process tracing, or verbal reporting) was first described by Newell and Simon[30] and, in recent years, by Ericsson and Simon[31]. The expert is given a typical problem to be solved and before the session begins s/he is requested to verbalize whatever s/he is thinking. The session is audio and/or video-taped and the protocol is transcribed and analyzed at a later stage. During the session the builder participates only when the expert seems to be idle by asking probing questions such as *what are you thinking at this moment?*. The technique minimizes the builder - expert interaction resulting in economising the expert's time. Although there are some problems associated with this technique, protocol analysis seems to be useful at the start of a project. Problems can be encountered due to the fact that not all individuals find it easy to verbalize and perform simultaneously and also most people can think more quickly than they can talk. In both cases knowledge might be lost. Additionally, protocol analysis can provide us with extensive information of how the knowledge is used but not about its full range. Finally, analyzing protocols is time consuming and difficult. Various authors have described types of analysis to apply to the same raw data in order to become familiar with it, to understand the reasoning involved and to facilitate the representation of the knowledge. A brief review follows. Waldron[32] provides a framework for classifying decision alternatives in terms

of alternatives, attributes, aspects and attractiveness. He also classified naturally occurring rules into dominant, lexicographic rules. Bainbridge[33] offers three analytic approaches to be applied to the transcript. Explicit content, implicit content and groups and sequences of phrases. She has used those approaches in analyzing verbal protocols from a process control task. Kuipers and Kassirer[34] analyzed a verbatim transcript taken from a second year student in three stages: Referring phrase analysis, assertional analysis and script analysis.

There exists a considerable overlapping on each author's ideas and proposals. This is something to be expected since knowledge elicitation is a new discipline but the terminology leaves something to be desired. Different people use the same terms to mean different things. A lot of research is under way in order to compare the various elicitation techniques so a builder can rate each technique's suitability under various circumstances. A review of knowledge acquisition evaluation research can be found in the article by Dhaliwal *et al*[35].

1.4.6 Expert systems and conventional programs

One might wonder what makes expert systems different from conventional ones. One might remark that in some sense, any computer program is expert at something. A payroll program incorporates knowledge about accountancy, but it is not included in the expert class. The reason being that the numbers generated by the payroll program might differ depending on the inputs, but they are always generated in the same way. Creating conventional programs

involves the definition, from the beginning, of the data, its nature and the process involved. The process consists of the presentation, in the proper order, of the correct set of procedures and control structures. The conventional approach typifies program-driven processing where what happens next at any particular point is pre-determined. Hence, conventional programs rely on algorithms which contain a step-by-step description of the procedures to be followed. These algorithms guarantee that the right conclusion will be reached when the correct data have been entered or that new knowledge from old can be inferred but the inference order is known. Expert systems differ from conventional software systems in that they are able to reason about data and draw conclusions employing heuristic rules. These are rules that have been formed through practical experience and they are employed to solve problems. Heuristic rules do not require perfect data and are not guaranteed to succeed but the proposed solutions are derived with varying degrees of certainty. The route to a conclusion varies according to the input data but the difference with conventional programs is that the inference order is not preset by the programmer. The inference order is determined by the success or otherwise of the branches of the rules. Heuristic rules are useful for situations where it is not possible to construct an algorithm. Another difference is that with conventional programming the knowledge and the processing procedures are tangled and spread throughout the entire program. In an expert system, however, knowledge is concentrated in one module and another separate module directs the inferencing. The separation means that one can make at least some changes to either module without necessarily having to alter the other. These differences led to the usage of different type

of programming languages employed. Traditional programming involves the use of imperative languages, whereas on the other hand declarative languages are employed for an expert system construction. Additionally, expert systems can reason using incomplete data and can generate explanations and justifications, even during execution of their actions. Once again these facilities are provided by separate modules.

1.5 An overview of the AEK expert system part

The knowledge engineer has at his disposal a number of tools to aid the construction of an expert system. These tools fall into four major categories. Programming languages, shells, development environments and domain specific tools. As described by Waterman[36] and Harmon *et. al*[37] there are a variety of expert system tools.

The AEK system was constructed using a commercially available expert system shell, namely Xi-Plus. Shells provide an alternative to programming languages since the knowledge engineer does not have to create the entire system from scratch. Shells like Xi-Plus provide an editor, the user interface, the inference engine and the explanation facilities. On the other hand the majority of such shells constrain the construction process due to the lack of a number of representation and searching schemes. This way the knowledge engineer might try to represent the whole of the area of knowledge using a single representation formalism. If the need arises development environments can provide the solution. These environments are equipped with more sophisticated editors, graphical interfaces and numerous representation methods. Gevarter[38] presents evaluation criteria for selecting a commercial tool for performing a particular task.

The Xi-Plus system was used because it was readily available. Its utilisation was continued because of the suitability of the architecture and control features for the task.

1.5.1 The AEK expert system part architecture

The system implements the most common form of architecture in expert systems, namely the rule-based architecture. The components of the system are the ones described in Section 1.4.2.

1.5.2 Representing knowledge in AEK (expert system part)

The knowledge is represented using rules, facts and defaults. Facts are statements which are true under all conditions. Defaults are values used in the absence of other information. Rules, or production rules[39], are small chunks of knowledge expressed in the form of *if..then* statements. The left hand side (IF) represents the antecedent or conditional part. The right hand side (THEN) represents the conclusion or action part. A number of rules collectively define a *modularized know-how system*[40]. A list of the benefits and drawbacks using production rules is given by Hayes-Roth[40]. The rationality for selecting rule-based presentation becomes apparent when examining the following three factors.

The wording of the expert: When dealing with experts, it is important to try to select the approach that is most natural to them. In our case, during protocol analysis (see Section 3.3), it became apparent that the

expert was expressing his problem solving techniques in terms of situation-action rules in order to show empirical associations between attributes. (Appendix 1 contains a protocol transcript).

The nature of the task: The tuning of the filter is accomplished by classifying the appropriate action to be taken from a pre-specified list of possibilities. Production rules can only represent what is called 'shallow' or 'low' knowledge[41] but they present a natural framework for classification tasks[42].

The use of an induction tool: Elicitation of knowledge was performed using an induction tool (see Chapter 7). The outcome was a decision tree which was transformed to a set of rules.

1.5.3 Control in the AEK expert system part

The shell comes with predefined control structures but the user can implement some of his/her own. When a user of AEK requests the classification of a given magnitude response the system operates in the backward chaining mode (i.e. tell me how to classify). The order of looking at the rules is lexical order viz. when scanning rules it will first look at rule 1, and then rule 2 etc. The order that the rules are recorded is then critical. Since the rules were generated from a decision tree, the system performs a depth-first search. When it searches, it inspects each rule to see if the left hand conditions are true. This is achieved by either reading the working memory or by asking questions or by generating further subgoals. In any case, the system continues to the next rule until all rules have been inspected (if this is not desirable the user can instruct the system to stop at the first true rule). Theoretically all rules that can execute must be placed in a conflict

set and one of the rules is selected[43]. Using Xi-Plus the system displays all options and the user has to make the decision. The selected rule then executes. This is what is known as the match, select and execute cycle. The system provides forward chaining (ie. what can you tell me when this data is true) as well. Additionally, meta-rules are available in order to reduce the search space. Other control facilities are the checking of outstanding queries, of a completed goal and the initiation of the evaluation of rules.

References

1. Zverev A., *Introduction to filters*, Electro-Technology, Vol. 73, pp. 61-90, 1964.

2. Sangwine S.J., *Electronic components and technology : engineering applications*, Van Nostrand Reinhold, 1987.

3. Van Valkenburg M.E., *Analog filter design*, Holt-Saunders International editions, 1982.

4. Bottom V., *Introduction to quartz crystal unit design*, Van Nostrand Reinhold, 1982.

5. Kinsman R.G., *Crystal filters : design, manufacture and application*, John Wiley and Sons, 1987.

6. Salt D., *Hy-Q handbook of quartz crystal devices*, Van Nostrand Reinhold, 1987.

7. Bowron P., and Stephenson F.W., *Active filters for communication and instrumentation*, McGraw-Hill, 1979.

8. Barr A., and Feigenbaum E.A., *The handbook of Artificial Intelligence*, Vol. 1, Morgan Kaufmann, 1981.

9. Winston P.H., *Artificial Intelligence*, Addison-Wesley, 1984.

10. Charniak E., and McDermott D., *Introduction to Artificial Intelligence*, Addison-Wesley, 1985.

11. Barr A., and Feigenbaum E.A., *The handbook of Artificial Intelligence*, Vol. 2, Morgan Kaufmann, 1982.

12. Cohen P.R., and Feigenbaum E.A., *The handbook of Artificial Intelligence*, Vol. 3, Morgan Kaufmann, 1982.

13. Barr A., and Feigenbaum E.A., *The handbook of Artificial Intelligence*, Vol. 4, Morgan Kaufmann, 1989.

14. Buchanan B.G., and Feigenbaum E.A., *Dendral and Meta-Dendral : their applications dimension*, Artificial Intelligence, Vol. 11, No. 1, pp. 5-24, 1978.

15. Shortliffe E.H., Axline S.G., Buchanan B.G., Merigan T.C., and Cohey S.N., *An AI program to advise physicians regarding antimicrobial therapy*, Computers and biomedical research, Vol. 6, pp. 544-560, 1973.

16. Gaschnig J., *Prospector : an expert system for mineral exploration*, Machine intelligence, Infotech state of the art report, Vol. 9, No. 3, 1981.

17. McDermott J., *R1 : a rule-based configurer of computer systems*, Artificial Intelligence, Vol. 19, pp. 39-88, 1982.

18. Bramer M.A., *A survey and critical review of expert systems research*, In: Introductory readings in expert systems (Ed. Michie D.), Gordon and Breach Science publishers, 1982.

19. Reddy R., *Foundations and grand challenges of Artificial intelligence*, AI magazine, Vol. 9, No. 4, pp. 9-21, 1988.

20. Bremer X., *Expert systems in business : a British perspective*, Journal Expert Systems, May, pp. 104-112, 1988.

21. IEEE Computer, Vol. 19, No. 7, 1986.

22. Hayes-Roth F., Waterman D.A., and Lenat D.B. (Eds.), *Building expert systems*, Addison-Wesley, 1983.

23. Woods W.A., *Important issues in knowledge representation*, Proceedings of the IEEE, Vol. 74, No. 10, pp. 1322-1334, 1986.

24. Hayes-Roth F., and Waterman D.A., *Principles of pattern-directed inference systems*, Academic press, 1978.

25. Brachman R., *On the epistemological status of semantic networks*, In: Associative networks, representation and use of knowledge by computer (Ed. Findler N.), Academic press, 1979.

26. Fikes R., and Kehler T., *The role of frame based representation in reasoning*, Communications of the ACM, Vol. 28, No. 9, pp. 904-920, 1985.

27. Israel D.J., *The role of logic in knowledge representation*, IEEE Computer, Oct, pp. 37-41, 1983.

28. Reichgelt H., and Van Harmelen F., *Criteria for choosing representation languages and control regimes for expert systems*, The knowledge engineer review, Dec, No. 4, pp. 2-17, 1986.

29. Nilsson N.J., *Problem solving methods in AI*, McGraw-Hill, 1971.

30. Newell A., and Simon H.A., *Human problem solving*, Prentice Hall, 1972.

31. Ericsson K.A., and Simon H.A., *Protocol analysis : verbal reports as data*, MIT press, 1984.

32. Waldron V.R., *Process training as a method for initial knowledge acquisition*, Proceedings of the second conference on artificial applications, Dec, pp. 661-665, 1985.

33. Bainbridge L., *Verbal reports as evidence of the process operator's knowledge*, In: Fuzzy reasoning and its applications (Eds. Mamdani E.H., and Gaines B.R.), Academic press, 1981.

34. Kuipers B., and Kassirer J.P., *Knowledge acquisition by analysis of verbatim protocols*, In: Knowledge acquisition for expert systems - a practical handbook (Ed. Kidd A.L.), Plenum press, 1987.

35. Dhaliwal J.S., and Benbasat I., *A framework for the comparative evaluation of knowledge acquisition tools and techniques*, Knowledge acquisition, Vol. 2, No. 2, pp. 145-166, 1990.

36. Waterman D.A., *A guide to expert systems*, Addison-Wesley, 1985.

37. Harmon P., Mans R., and Morrissey W., *Expert systems : tools and applications*, John Wiley and Sons, 1988.

38. Gevarter W.B., *The nature and evaluation of commercial expert systems building tools*, IEEE Computer, May, pp. 24-37, 1987.

39. Newell A., and Simon H.A., *Human problem solving*, Prentice Hall, 1972.

40. Hayer-Roth F., *Rule based systems*, Communications of the ACM, Vol. 28, No. 9, pp. 921-932, 1985.

41. Michie D., *High road and low road programs*, AI magazine, Vol. 3, pp. 21-22, 1982.

42. Clancy W.J., *Classification problem solving*, Proceedings of national conference on AAAI, pp. 49-55, 1984.

43. McDermott J., and Forgy C., *Production system conflict resolution strategies*, In: Pattern directed inference systems (Eds. Hayes-Roth F., and Waterman D.A.), Academic Press, 1978.

2.1 Introduction

Chapter 2 provides an introduction to the manual procedure currently in use (Section 2.2). Section 2.3 describes previous work in the field of electronic filter tuning. Three approaches are described in total. The heuristic and the machine learning approaches were selected since their overall methodology is close to the one followed in this work whereas the third approach (sensitivity-based approach) represents conventional techniques. Finally, Section 2.4 discusses the motives for implementing the expert-neural (Hybrid) approach by identifying the strengths and weakness of the previous approaches and the areas where the hybrid system can perform (or compliment) better. It was hoped that the hybrid would eliminate repetitive and time consuming calculations, provide a better system-human interface and enable a complete automation of the tuning task.

2.2 Manual tuning procedure

Manual tuning can be thought of as a human real-time optimisation which attempts to reduce the total and individual errors in the features of interest, with as few steps as possible. Error is defined as the difference between the required and the obtained performance.

There does not appear to be a general theory of the practical tuning of filters. Through an initial training and with acquired experience the operator is transformed into a skilled operator. An experienced operator then effectively generates an heuristic algorithm for tuning a particular type of filter. Knowledge about which components are appropriate for adjustment for tuning and which to be ignored, the order of the specification checking etc. is

referred to as heuristic. Heuristic algorithms are different to conventional algorithms in the sense that they do not guarantee success or a solution. They can fail at certain times, but often they work. The difficult part, as will become obvious later on, is to extract the algorithm. The operators appear to be unaware of it.

Despite the variations between operators, which can be found in detail, the general pattern is the same. An operator checks the performance of the filter (e.g. magnitude response). From experience coupled with the feedback provided by the response measurement system he or she decides what corrective action, if any, is to be taken. The action being the adjustment of an appropriate tunable component. These steps are then repeated as many times as necessary until the performance satisfies the requirements. Then the response is checked at a set of frequencies and further corrective actions, if required, are carried out. Effectively, the operators act as signal interpreters and the interpretation is not based on any theory but is essentially a synthesis of a strong capability for pattern recognition linked with knowledge accumulated from past experiences.

2.3 Work in the electronic filter tuning field

Although manual tuning is successful the advantages of providing computerised assistance to an operator have been recognised before. This section introduces and contrasts the work of others in the field. The reasons behind the motivation for using the expert system technology are also discussed. Rather than introducing a catalogue of all techniques, this chapter will highlight on three proposed methods, namely the work described by Nazemi and Fidler[1], Mirzai[2], and Crofts and Jervis[3]. The first two projects are

most directly relevant to this work due to the involvement of experts, in Nazemi *et. al.* case, and the machine learning approach, in the case of Mirzai. A discussion of the three techniques will hopefully help to understand AEK's contribution to the field.

2.3.1 Filter tuning using a microprocessor based heuristic algorithm

Nazemi and Fidler[1] realized the need for the automatic tuning of filters and proposed a method which took into consideration the operator's knowledge. The development of the heuristic method involved three phases. The first phase involved the selection of the tuning components and the frequency points. To facilitate the selection, sensitivity analysis was employed as a starting point. Secondly, the error and stopping criteria were defined in order to have some means of stopping the tuning process. Finally, the heuristic tuning algorithm was developed. This involved the creation of an information storage data table (ISDT). The table included information on which component to adjust and the direction of adjustment at every test frequency point. This information was dependent on the polarity of the error. This ISDT was stored in the memory of a microprocessor controlled system which tested the filters after each adjustment and then adjusted them again, and so on, until they were tuned. What is interesting and of particular relevance to our work is the method used to generate the table. In general, the algorithms, since each type of filter has a different one, were developed by tuning the filter manually a number of times. The pattern of tuning and the pattern of adjustments were combined and their examination resulted in the creation

of the algorithms. In particular, for a second order Sallen and Key lowpass filter Nazemi[4] reported the creation of the algorithm as follows:

> "By performing the tuning manually many times, the best approach was recorded and from that an ISDT was formed." [Chapter 5, page 142]

The goals of the testing of the heuristic algorithms were as follows:

(a) Can the heuristic algorithm be used on its own, and

(b) can the heuristic algorithm be used as a front-end of another technique. If so, are there any benefits in doing so.

The heuristic algorithms were tested on a number of hardware circuits and compared to a pattern search optimisation technique devised by Hooke and Jeeves[5]. The criterion of comparison was the number of measurements carried out by each method. One conclusion was that the heuristic algorithm can be operated on its own but usually resulted in a coarse tuning. An important observation was the substantial improvement in the number of measurements. When used as a front-end no more than eighty-eight (88) measurements were required although total reliance on the Hooke and Jeeves method required a minimum of five hundred (500).

2.3.2 Alignment of filters using a Machine Learning System

Mirzai[2] proposed a machine learning system (MLS) for tuning waveguide filters. The MLS was originally developed for fault diagnosis of telecommunications systems, in particular microwave digital radios[6]. The approach is based on linear adaptive combiner algorithms and more information is given in Chapter 4. Here, only an outline of the MLS will be

given. The overall system is used in two modes, namely: the training mode and the use mode. In the training mode the adaptive combiner was used for fine tuning only. The coarse tuning was performed manually. In order for the algorithm to learn how to perform the fine tuning the following steps were taken:

(i) The reference characteristic was selected. This was the S_{11} polar plot (Figure 5a). This was chosen because it enabled tuning of the group delay of the filter as well as its amplitude response. S_{11}, where S stands for scattering, looks at the division of the output by the input in frequency domain at all the frequencies of interest. The scattering parameter using a network analyzer system enabled the measurement of both the magnitude and phase information and the plotting of the data on a polar display. The measurement of the scattering parameter can be illustrated better using network parameter theory. Figure 5b which displays a flow graph of a two port network will be used. Nodes a and b are the incident and reflected nodes respectively. When an incident wave enters the device at node a of port 1, part of it will be returned through the S_{11} path and b_1 reflection node. Part of the wave will be reflected through the a_2 node as well. This can be expressed as:

$$b_1 = a_1 * S_{11} + a_2 * S_{12}$$

If the device is not connected to port 2 (i.e. by terminating port 2 with i t s characteristic impendance) then the equation becomes:

$$b_1 = a_1 * S_{11}$$

Therefore $S_{11} = b_1 / a_1$ given that $a_2 = 0$.

Other scattering parameters can be measured in a similar fashion. These generalized parameters can be measured easier than other traditionally used

parameters especially for frequencies above 100 MHz. Additionally their conversion is quite simple.

(ii) A set of prominent features were extracted from the reference characteristic in order to have some means of assessing the sensitivity of the adjustable components on the polar plot. In total sixteen (16) features were selected. These included the area of the loops, the geometric mean of the plot etc (Figure 5a).

(iii) The adjustable components to be used were selected - in total six (6).

(iv) The value of each feature for a fine-tuned filter was recorded.

(v) Further examples were generated by simply mal-adjusting one adjustable component at a time. This was implemented for both directions.

(vi) The examples were fed to the algorithm and a number of combiner weights were calculated. These weights represented the knowledge in the form of mathematical relationships.

In the use mode the system was simply provided with the feature set of a coarsely tuned filter. This initiated the production of a graphical display of the adjustment levels of each component. The component which generated the maximum error at each iteration was adjusted. This process was repeated until the response of the filter was within the specifications set by the reference filter. One coarsely tuned filter was found to meet the specification within twenty (20) adjustments. Unfortunately, the initial amount of mal-adjustment has not been reported.

2.3.3 Sensitivity-based filter tuning

This section introduces the work by Crofts and Jervis[3] which is based on sensitivity analysis. The concept of sensitivity involves the identification of

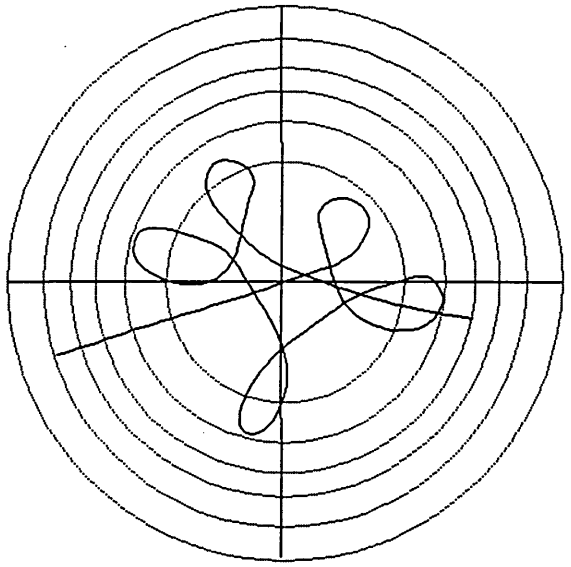


Figure 5a : A typical S_{11} polar plot

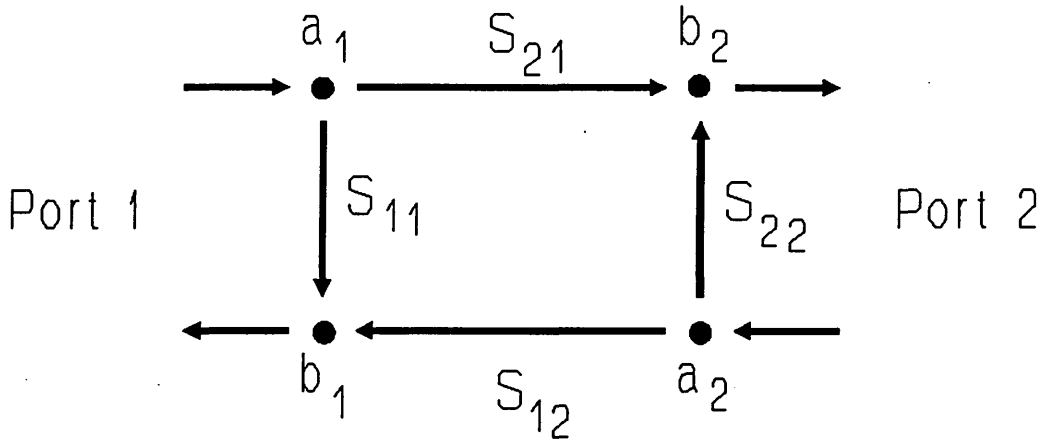


Figure 5b : Flow graph of a two port network

the relationship between variations in a particular function F and the variable parameters of that function. In the case of circuit analysis the voltage transfer function H(s) was used and the adjustable components X_k represented the variable parameters. Two tuning algorithms by Antreich[7] *et. al.* and Jobe[8] were compared using simulated and actual tuning of two differently designed low-pass, 7th order, elliptic filters (a 4.5 MHz and a 100 kHz filter). Only the magnitude response was considered and the work of Crofts and Jervis[9] involved the identification of which adjustable component (one of three inductors) dominated the sensitivity of the magnitude response at some selected frequencies. An outline of the tuning procedure is given below:

(1) Calculate the network response using (H(s)) the nominal component values at six selected frequencies.

(2) Perform the sensitivity analysis by incrementing each component in turn by a known value (± 2.5 for the 4.5 MHz filter, ± 4 for the 100 kHz filter) from its nominal value.

(3) Calculate the adjusted network response at the six selected frequencies and compare with the specification. Their difference (Δ H(s)), termed object function, at each selected test frequency was found by simple subtraction. The magnitude sensitivity was calculated using the following formula:

$$S_{X_k}^{H(s)} = \frac{X_k}{|H(s)|} \frac{\partial |H(s)|}{\partial X_k}$$

If the specification was satisfied then step (5) was performed, otherwise step (4).

(4) The object function combined with the results from step (2) indicated

which component(s) were in error. Using Antreich's[7] *et. al.* method the adjustments, for a circuit with two adjustable components analyzed at two frequencies, were given by:

$$\Delta_x = \left\{ \begin{array}{l} \dfrac{E_1}{S_{11} - \dfrac{S_{12} \cdot S_{21}}{S_{22}}} + \dfrac{E_2}{S_{21} - \dfrac{S_{11} \cdot S_{22}}{S_{12}}} \\ \dfrac{E_1}{S_{12} - \dfrac{S_{11} \cdot S_{22}}{S_{21}}} + \dfrac{E_2}{S_{22} - \dfrac{S_{12} \cdot S_{21}}{S_{11}}} \end{array} \right\}$$

Using Jobe's method the adjustments were given by:

$$\left\{ \begin{array}{l} R_1 = \dfrac{E_1}{S_{11} + \dfrac{S_{21} \cdot S_{21}}{S_{11}}} + \dfrac{E_2}{S_{21} + \dfrac{S_{11} \cdot S_{11}}{S_{21}}} \\ R_2 = \dfrac{E_1}{S_{12} + \dfrac{S_{22} \cdot S_{22}}{S_{12}}} + \dfrac{E_2}{S_{22} + \dfrac{S_{12} \cdot S_{12}}{S_{22}}} \end{array} \right\}$$

The required component adjustments were then given. The tuning procedure was repeated from step (2).

(5) The tuning procedure was terminated.

The tuning results with the computer simulations and the actual tuning showed that[9]:

- Both tuning algorithms were capable of tuning the filters.
- The Antreich *et. al.* method was more efficient than the Jobe method (simulation results).
- In the case of the actual tuning of the 4.52 MHz filter no difference could be found between the performance of the two methods.
- The actual tuning of the 100kHz filter showed that there was a poor match between the practical tuning and the computer simulation but tuning was

achieved in most cases.

2.4 Motivations for using a hybrid system

The motives for employing a hybrid system (expert system, neural network) in the filter tuning domain can be categorised into three broad areas, namely technical, business and science.

Technical considerations

The desirability of applying expert systems in terms of a comparison with other approaches and general task properties were considered. The expert system approach could be used for comparison with other techniques in terms of measurements required and time taken. However, such comparisons are not feasible since the various authors describe their work using different filter types. An investigation could be carried out where the same filters will be used, but unfortunately this is work which may never be performed. The question is then best answered by considering how well those previous approaches fulfil the requirement of a system which exhibits certain essential and desirable features. Such essential features are: the reporting of which tunable component to adjust, in which direction and by how far. The desirable features are: generality, explanation of reasoning and easy human interaction. Discussing briefly those approaches, one can assert that both Mirzai and Crofts provide excellent information about the essential features. The drawbacks are the need for repetitive and time consuming calculations (especially Crofts), lack of generality and basic system-human interface. The latter indicates that the systems cannot possibly be used as tutors. Further, Crofts work is deterministic and corresponds to an inexperienced operator, viz. it starts from scratch in every case and does not take into account the

expertise of an operator. On the other hand Nazemi and Fidler use the operator's knowledge but the elicitation method leaves loopholes. For example the "best" approach of the day does not guarantee it will always be the best. This is acknowledged by Nazemi and Fidler who conclude that such heuristics must be generated automatically - something our work contributed towards. At the same time the work of Nazemi and Fidler does not provide information about distance (i.e. how far to turn).

The general task properties that have to be satisfied when selecting an expert system application are numerous[10,11,12]. For example, there must exist recognised experts who are probably better than novices in performing the task. The task must be well bounded, must require the use of reasoning and not just numeric processing, and must be neither too easy nor too difficult. The filter tuning task satisfies these expectations.

Business considerations

The other major aspect is the value of the system to the business. At the present time, manual tuning has some drawbacks. It is time consuming, represents a large proportion of the total filter production cost, and can be described as uninteresting and uncreative. An expert system could free the operator to undertake work more satisfying to him or her and be more productive for the manufacturer.

Knowledge considerations

The filter tuning task is different to, say, the familiar domain of medicine. In the medical field one deals with a highly qualified expert, with several years of practice, able to reason for the decisions taken and performing in static time. By contrast the operator in the tuning process is not highly qualified,

not always able to reason and operates in real time with a constantly changing environment. The numerical nature of the knowledge and the problems of eliciting the knowledge resulted in needing a further tool, i.e., creating a hybrid expert system-neural network system. Our goal was then to develop a hybrid system to provide the operator with all the essential features using an appropriate display.

References

1. Nazemi J., and Fidler J.K., *Filter tuning using a microprocessor based heuristic algorithm*, Proceedings of the 1985 European conference on circuit theory and design, pp. 101-104, 1985.

2. Mirzai A.R., *Waveguide filter alignment*, In: Artificial Intelligence - Concepts and applications in Engineering (Ed. A.R. Mirzai), Chapman and Hall, 1990.

3. Crofts M., and Jervis B.W., *Sensitivity-based computer-aided tuning of elliptic filters for optimum magnitude vs frequency response*, Research report, Department of Electrical and Electronics, Sheffield City Polytechnic, England, 1987.

4. Nazemi J., *Microprocessor-based filter tuning system*, PhD thesis, University of Essex, 1984.

5. Hooke R., and Jeeves T.A., *Direct search solution of numerical and statistical problems*, Journal of ACM, Vol. 8, pp. 212-229, 1961.

6. Brown K.E., Cowan C.F.N., Crawford T.M., and Grant P.M., *Knowledge-based techniques for fault detection in digital microwave radio communication equipment*, IEE Journal on selected areas in communications (Special issue on knowledge-based systems for communications), Vol. 6, No. 5, pp. 819-827,

1988.

7. Antreich K., Gleissner E., and Muller G., *Computer aided tuning of electrical circuits*, Nachrichtentechnische Zeitschrift, Vol. 28, No. 6, pp. 200-206, 1975.

8. Jobe G.G., *Computer aided adjustment of electrical filters*, MPhil thesis, Newcastle-upon-Tyne Polytechnic, England, 1979.

9. Crofts M., and Jervis B.W., *A comparison of computer-aided tuning algorithms applied to the amplitude response of passive analogue filters*, IEE Proceedings on Circuits, Devices and Systems, Vol. 138, No. 3, pp. 363-371, 1991.

10. Zack B.A., *Selecting an application for knowledge-based system development*, Proceedings of the third international expert system conference, pp. 257-269, 1987.

11. Prepau D.S., *Selection of an appropriate domain for an expert system*, AI Magazine, Vol. 7, No. 2, pp. 26-30, 1985.

12. Laufmann S.C., DeVaney D.M., and Whitinh M.A., *A methodology for evaluating potential KBS applications*, IEEE Expert, Vol. 5, No. 6, pp. 43-61, 1990.

Part B

Knowledge-base Construction

Chronicle Elicitation

Chapter Three

Initial Knowledge Elicitation

3.1 Introduction

Chapter 3 reports on the work and the results obtained during the first visit to Newmarket Microsystems. The results included the selection of the expert operator and the type of filter to be employed. Additionally, protocol analysis was identified as a suitable starting knowledge elicitation technique mainly because of the verbal on-line format of the technique. Section 3.3 presents the protocol analysis implementation and the subsequent analysis of the transcripts. The main analysis result was the identification of the overall filter tuning procedure. Furthermore, the analysis of the transcripts indicated the need for an alternative elicitation technique due to the apparent lack of theory behind the selection of a particular tunable component the direction and how far to turn it (Section 3.3.2).

3.2 The first visit

The first stage of any knowledge engineering project must always be the familiarization of the knowledge engineer with the domain. In addition, various general but important questions have to be answered before the task commences. For that reason the objective of the first visit to the collaborating establishment was to obtain background information beneficial for domain acquaintance[1]. The following activities were carried out:

√ Identification of benchmark filter

√ Identification of expert operator

√ Identification of sources of reference

√ Identification of the role of the system

√ Identification of any parenthetical knowledge

√ Elicitation of concepts

√ Definition of the problem areas

√ Identification of appropriate knowledge elicitation technique.

3.2.1 Identification of benchmark filter

One of the first tasks was to select a suitable filter. This filter had to satisfy two requirements. Firstly, the tuning of such a filter had to be more or less representative of the task. Secondly, the tuning process had to be neither too trivial, because the effort of developing an expert system might outweigh the potential benefits, nor too difficult. The filter had to be somewhere in the middle of the complexity scale. A factor which probably determines how easy or difficult the tuning of a filter will be is the number of adjustable components. Another factor derives from how trivial or complex the required specification is. The degree of complexity depends, for example, on the requirement of examining the phase response or on the number of frequency ranges to be checked. The collaborating establishment manufactured more than 200 types of crystal filters. With the help of an operator the whole spectrum was segregated into three categories . From each category one filter type was identified. The filter type from the medium category was elected to be the benchmark filter.

3.2.2 Identification of expert operator

The choice of whom to use as expert is critical. Without an expert, there cannot be a system, unless the knowledge engineer is also the expert. At Newmarket, there exist various people who have competence in tuning filters. These people differ in age, experience and qualifications. Most operators fit

one category: those people with few years experience on the job and unqualified. Our expert was chosen because of his vast experience in designing and tuning filters (over 25 years), his willingness and enthusiasm about the project and his articulateness.

3.2.3 Identification of sources of reference

Sources of reference are often sufficient to introduce the knowledge engineer to the domain. Unfortunately, despite the plethora of books about filters and their design, there is no book on how to tune filters. A reason for this might be that filters are manufactured for a particular client's specification, resulting in hundreds of different designs. A formal theory or methodology has not surfaced. What was made available was information for the benchmark filter. That information included a schema of the filter, the specification that it had to satisfy and a graphical representation of the magnitude response.

3.2.4 Identification of the role of the system

An expert system can act in a number of different roles[2]. For example as an assistant - performing a sub-task of the process, or as a critic - reviewing the decision of the expert and providing comments. The role a proposed system takes depends on the user. Is it going to be used by an expert or a novice? It also depends on the degree to which the problem can be automated. Another factor is the company's wish, which in a commercial world is probably the most important one. By discussing the subject with the expert and senior staff, it was decided that the system could take the role of the consultant. That way the system offers an opinion which the user does not have to

comply with.

3.2.5 Identification of any parentheti- cal knowledge

The term 'parenthetical' is borrowed from Freiling *et. al.*[3] who define it in the following manner:

> "...knowledge about how the task being performed relates to other tasks and the operational environment in which the task is being performed."

Another term that can be used is associated knowledge. There is not a methodical way to obtain this kind of knowledge but it comes out during casual conversations. A guided tour of the filter tuning production line was made during the visit. The answers to questions such as *what happens when the task is completed* were obtained during the tour. Filters were tuned by trained persons. In situations where the task could not be completed the filter was passed to a more experienced person. He could either tune it or reject it because there was something fundamentally wrong. When the filter characteristics were tuned to within the specification the filter was packaged into a metal box and sealed. Then it was distributed to the client. A new person is trained in-house by a senior operator and it can take up to three months to reach a satisfactory level of competence. Initially the training involves monolithic filters and later on other types. This indicated that some overall generality might exist. One must collect such information because it can affect the design and the role of the expert system. For example, the specification could be supplied automatically by the system eliminating the job of searching for the correct specification.

3.2.6 Elicitation of concepts

Prior to the visit a letter was prepared (Appendix 2) which was presented to the expert. The purpose of the letter was to collect those concepts influencing the decision process. At that time it was unclear what those concepts were. The expert was asked to tune a filter and at the same time to record those concepts on a piece of paper. The expert faced difficulties with the term 'concept'. His answers took the form of description of the task instead of only the concepts, which can be found hidden in the text.

3.2.7 Definition of the problem areas

When the operator decides that the characteristics of the magnitude response of a filter are not within the desired specifications, he must choose which section of the response to adjust first, which tunable component to use, in which direction to turn it and by how far. He also has to determine which action is to be taken in order to correct a wrong choice. One minor problem is that the operator wastes time searching for the specification of each filter.

3.2.8 Identification of an appropriate knowledge elicitation technique

The technique chosen for the filter tuning project was protocol analysis (see Section 1.4.5.1). It was considered appropriate to video-tape the sessions for the following reasons. In the filter domain the expert interprets, plans and executes tasks by visually inspecting the display unit of the measurement set. The set displays the magnitude response of the filter. By adjusting the set, the expert can inspect the full response or part of the response. It was felt

that the expert would have found it difficult, or even impossible, to describe the response in a verbal off-line format. The choice, then was between behavioral observation and protocol analysis. Since the knowledge engineer was unfamiliar with the domain terminology, protocol analysis, where the expert refers to the task process using the terminology, was preferred to the behavioral observation. Protocol analysis was selected for the beginning of the analysis process. Protocol analysis had to be complemented with other techniques (e.g. structured interviews) which were thought to be more useful in a 'more clarification' mode. The reader must appreciate that knowledge elicitation is at a very early stage of development, where general principles have not emerged and only a combination of techniques can provide fruitful results. The combination will vary from project to project. The expert and senior management did not oppose the idea of using a video recorder so a second visit was arranged.

3.3 Protocol analysis implementation

The tuning of the chosen type of filter was video-taped twice. The expert was instructed to 'think-aloud' about the process and to refer not only to his mental skills but also to his manual skills. Manual skills means those needed to operate the measuring set. Mental refers to the reasons behind each action taken, such as why to turn component X instead of Y. At the end of the recordings the video tape was played back and notes were taken. Those notes were concerned with:

(a) ambiguous statements

e.g. "...arrange these peaks into a more reasonable place."

(b) cross-reference of the expert's decision taking. That involved

watching the two video takes of the process and comparing them.

(c) recording probing questions for further use. Questions such as 'why did you take that action' for those situations where the operator did not provide any explanations.

(d) transcribing and analyzing the verbatim account. That involved watching and listening to the tape and writing on to paper everything that the expert was saying.

Some general observations are as follows:

√ the expert did not find it difficult to verbalize his manual skills nor, in some circumstance, to explain his reasoning but there was a steady decrease of the level of details from the first recording to the last one.

√ The expert was able to describe the tuning process for whichever component he was tuning at a particular time but when there was more than one candidate component he did not provide a theory for which one to select.

3.3.1 Analysis of the transcripts

It was realised early on that the transcription process is time consuming . When both video takes were transcribed they were entered into document files of the Wordstar wordprocessing package. Packages as such can be very useful as support tools to browse and edit the text. Prints of the transcripts can be found in Appendix 1. The files include a reproduction of the protocols in a complete fashion, and no attempt to filter the contents of the protocol was made. Each transcript was broken into short lines, in such a way that each line contained a phrase which could stand in its own right. Having individual lines did not provide any additional knowledge but made the

Table 2 : Tunable components associated with each feature	
Feature	Component
Ripple	T_1, T_2, T_3
Passband width	T_3 (maybe T_1, T_2)
Attenuation	C_4, C_7
Stopband width	C_4, C_7
Ultimate attenuation	C_4, C_7
Insertion loss	reject

transcript easier to read, understand and analyze. Where it was possible the format was: Do this - Why - Because (action - justification - explanation). Appendix 3 includes the phrase transcriptions. Lines which did not belong to this format were either general comments or operational comments. The benefits of the transcription analysis were as follows:

√ Identification of order for specification checking ie. what features and in what order were checked. If during checking one feature needs re-adjustment, the expert attempts to fix it and he starts re-checking from the beginning.

√ A set of possible tunable components associated with each feature identified above was also recognised (Table 2).

√ The classes of activity the operator engaged in were identified. The operator had knowledge about the measuring set, useful in order to have the most appropriate display at each time (Operational). He had knowledge of how to interpret a response and identify those regions, if any, that need adjustment (Interpretational). Additionally, he had knowledge of which region, or part of, to tune first, what to follow etc. (Planning), knowledge of how to proceed in order to make a final check (Inspection), and knowledge of how to recognize an achieved state (Recognition).

By identifying the various activities, one can concentrate and tackle a particular activity at a time (i.e. modularity).

√ The objects were recognized and classified. By objects is meant the most primitive lexical entries that the expert uses to express domain knowledge. Objects usually take the form of a noun or a compound

noun. Such objects in the verbatim transcript are: "coils", "capacitors", "frequency", "anticlockwise" etc. The outcomes of the object identification process were twofold. Firstly, the knowledge engineer became familiar with the domain terminology, resulting in the production of a lexicon[1]. A sample can be seen in Appendix 4. Each definition is from the IEEE standard dictionary of electrical and electronic terms[4]. Secondly, synonyms were identified which helped to reduce misunderstandings. For example screw-in and clockwise mean the same action.

√ Casual statements, with a lot of information, were identified. Such statements were as such:

- it is used to adjust the passband
- capacitors are used to adjust the stopband
- the right capacitor is the best bet to adjust the return levels.

√ The expert's tuning process was identified. That is, a general overview of how the expert proceeds. The expert's process can be split into three main stages. Set-up the measuring set, qualitative tuning and quantitative tuning. Stage one, is simply the setting-up of the measuring set using, for example, the reference frequency. The first stage is not of concern since it is mechanical in nature and is the same for any type of filter, except of course, that different reference values are used. By qualitative tuning, is meant that stage in which the expert uses visual information to decide if tuning is required. He also employs visual information when he adjusts a particular component to determine if the correct action has been taken. Quantitative tuning can be thought of as the specification

checking. The expert uses not only visual information but numerical values (obtained from the meter) to determine if more tuning is required.

3.3.2 The need for an alternative elicitation technique

Let us concentrate on the second stage. Stage two, can be broken down into two further sub-stages. Tuning of the stopband and tuning of the passband. It was also discovered that the expert always attempts to tune the stopband region first. Additionally it was found that the trimmer capacitors are the only adjustable components to be used for the stopband tuning. The inductors are used for the passband tuning. Another observation was that having successfully tuned component X, then when he moved to the next component he tuned in the same direction as he did with X. The problems arose when the expert was unable to provide any explanations of either why he selected a component X instead of Y, or why a certain direction was chosen. It seemed that the expert either made those decisions by chance or that something triggered his decision which he was not able to express. Also, he did not express by how far to turn. The expert actually kept turning until a particular shape of the response was reached. The rules governing what constitutes a satisfactory shape could not be expressed. This situation was worse in those circumstances where the expert had moved away from the 'optimum' state. His subsequent action was to turn the component the opposite direction until the 'optimum' state was re-achieved. To overcome the problem of which component to use at certain response states, which direction

to turn and by how far to rotate, the possibility of automatically acquiring and updating the rules was considered.

3.4 Conclusions

During the first visit to Newmarket Microsystems the 4716-type of crystal filter was selected as the benchmark filter and it was decided that the computerized system should act as an advisor.

It was decided to apply protocol analysis as the first step for acquiring knowledge. Following the implementation and analysis of the protocol transcripts it was clear that machine learning algorithms as the means for automatic knowledge elicitation must be investigated.

References

1. Grover M.D., *A pragmatic knowledge acquisition methodology*, Proceedings of the 8th IJCAI, Vol. 1, pp. 436-438, 1983.

2. Hayes-Roth F., Waterman D.A., and Lenat D.B. (Eds.), *Building expert systems*, Addison-Wesley, 1983.

3. Freiling M.S., Alexander J.H., Messing S.L., Remfus S., and Shulman S.S., *Steps towards automating expert system development*, Proceedings of the International Conference on Cybernetics and Society, pp. 988-993, 1985.

4. Jay F. (Chief Ed.), *IEEE Standard Dictionary of Electrical and Electronics Terms*, The Institute of Electrical and Electronics Engineers, 1984.

Chapter Four

Machine Learning Principles and Techniques

What we have to learn to do, we learn by doing

Aristotle

4.1 Introduction

As already mentioned in Chapter 3 an alternative knowledge elicitation approach had to be considered since the applicable classification rules were not clearly known.

Recently systems which are capable of automatically identifying and synthesizing the knowledge of an expert have proved of interest. Machine learning systems is the commonly used term to describe such systems. The concept of machine learning and in particularly learning through the use of examples is the subject of Section 4.2. In Sections 4.3 to 4.5 three systems (ID3, Adaptive Combiners, Neural Networks) are described. The algorithm of each system is given in detail and the main limitations and proposed modifications are highlighted.

4.2 Introduction to Machine Learning

The power of an expert system depends on the knowledge incorporated into the system. Knowledge must first be elicited and subsequently represented and refined. The task of elicitation has been labelled as the *bottleneck*[1] of the construction process of such systems. One role of machine learning is to assist during the elicitation process and to bypass the *bottleneck*. Additionally, expert systems perform in a deductive format[2], i.e. the conclusions always depend on the knowledge supplied. The presence of an incorrect conclusion can generally only be corrected by the builder's interference and not by the system itself. Systems that learn improve the quality of their performance with time without being reprogrammed. An improvement of a performance can be manifested by a faster response or a higher proportion of correct

decisions or both. Three major research paradigms can be identified: neural modelling and decision-theoretic techniques; symbolic concept acquisition (SCA); and knowledge-intensive, domain-specific learning[2]. Each paradigm is based upon the same principle, namely that of inferring conclusions given *a priori* knowledge, and differs from the others only in the amount of information required and in the way the knowledge is represented and modified. A number of learning strategies have been documented[2] but in the work reported here techniques which learn from data composed of a number of independent examples have been implemented. Each example is described in terms of a number of attribute values, together with an additional attribute, known as the class, which allocates the examples to a particular category (supervised learning). A number of different techniques have been reported in the literature, e.g. neural networks[3], genetic algorithms[4], and the AQ (Aurora) family[5,6,7] of algorithms. The techniques chosen were ID3, adaptive combiners and three-layer neural networks and these are outlined briefly in the following sections. The reasons for choosing these three techniques were more practical than theoretical. Extensive previous work using adaptive combiners in the field of tuning of waveguide filters[8], in addition to the availability of a commercial package implementing ID3, were the main factors behind the decision. Therefore, results obtained with ID3 and adaptive combiners can be compared and any benefits of using one technique rather than the other can be identified. Neural networks were chosen because of their ability to model non-linearities (a shortcoming of the adaptive combiners). It has also been reported that ID3 is faster, in terms of induction, than AQ11[9] or a genetic algorithm[10] with the same performance

rate.

4.3 The Iterative Dichotomiser Three (ID3) Algorithm

One learning strategy is induction. Induction means reasoning from specific cases to general principles. A subdomain of induction is concept learning from examples. This involves the generation of rules (or any other kind of presentation) which best classify the examples with which the system was presented. Best refers to the accuracy factor when tested with previously unseen examples and the comprehensibility of the rules. Comprehensibility of the rules is critical since it determines how effortlessly the knowledge can be understood and consequently conveyed to people in order for them to appraise, critise and use. In this section ID3, an example of an inductive inference system, is described. Prior to the presentation of the actual algorithm it is worth noticing the following points:

(i) The algorithm does not use any other domain specific knowledge beyond that of the training examples themselves.

(ii) The algorithm applies to a variety of application areas, viz. it is a general purpose algorithm.

(iii) The original algorithm looks at the entire set of training examples before forming the rules. This is usually referred to as a single learning stage. Further offsprings of the algorithm bypass this requirement, this is known as *windowing*.

(iv) The rules which ID3 learns are represented as decision trees. A decision tree embodies the relationships between the

attributes and the classes. Each node of the tree represents an attribute and each branch corresponds to a possible value the attribute can take. Each terminal (leaf) node represents a class prediction to be assigned.

The ID3 algorithm was developed by Quinlan[11] in 1979 and is a descendant of Hunt *et al*[12]'s concept learning system. A diagrammatic description of the algorithm is shown in Figure 6. The decision tree is grown in stages. First the algorithm looks to see if all examples belong to the same class. If they are the label 'null' (or something equivalent) appears. Otherwise, the algorithm selects the most informative attribute and either forms subsets equal in number to the number of values the attribute takes (i.e. creates the branches of the decision tree) or forms a binary split (cutoff point) when the attribute holds numerical values (e.g. >5, <=5). For each subset the algorithm checks whether all the examples are of the same category. If they are then the algorithm labels that subset with the name of the class (ie. creates a leaf of the decision tree) and partitioning stops for that subset (labelling rule); alternatively the algorithm creates further, smallest subsets. The algorithm stops when no more subsets can be created, i.e. the tree has been grown meaning all leaves and internal nodes have been defined and all examples have been considered (termination rule). It is worth noticing that the algorithm may label a leaf as 'empty' or 'clash'. *Empty* appears when there are no examples that can be used for that particular branch. *Clash* emerges when there are two (or more) examples covering that specific branch but their classes are distinct. The key principle underpinning the algorithm lies in the selection of the most informative attribute and is based on Shannon's classic

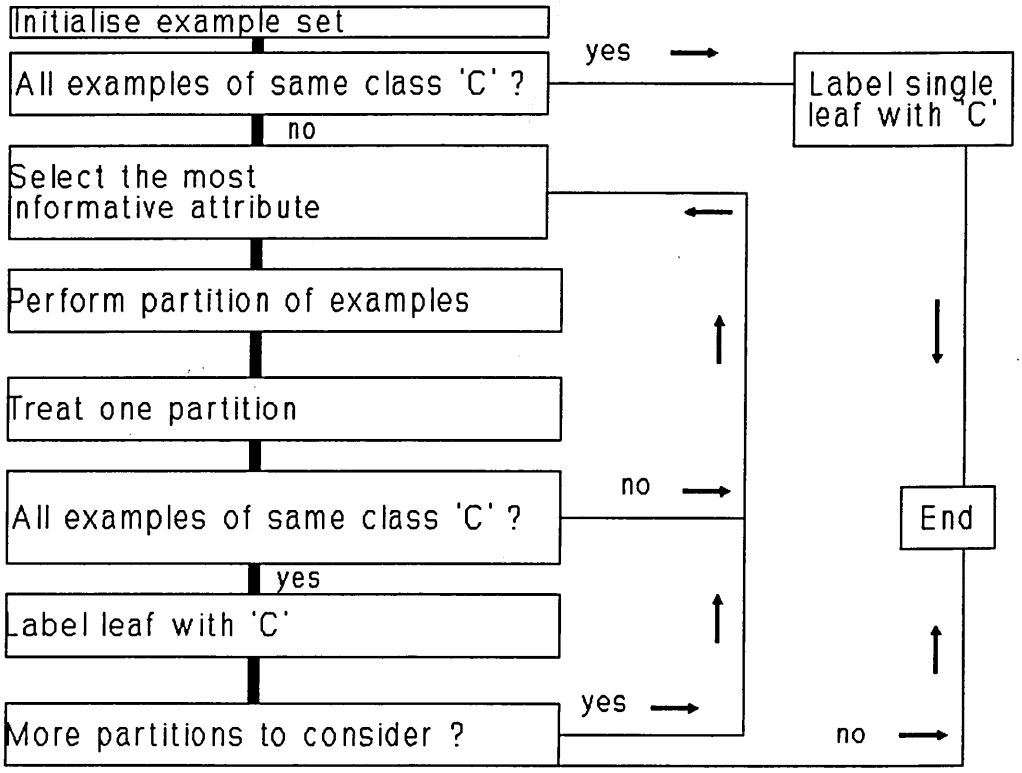


Figure 6: The ID3 algorithm

work in information theory[13]. The most informative attribute, at a certain instance, is the one that maximises the information gain (G) which is calculated by:

$$G\ (of\ attribute\ X_i) = I - E_i, \quad i=1..total\ number\ of\ attributes$$

where **I** is the expected information of the whole training set and **E** is the expected information of the whole training set when only attribute X_i considered. Both values can be expressed as:

$$I(y,n) = -\frac{y}{y+n}\log_2\left(\frac{y}{y+n}\right) - \frac{n}{y+n}\log_2\left(\frac{n}{y+n}\right)$$

$$E(X_i) = \sum_{i=1}^{u} \frac{y_i+n_i}{y+n} I(y_i,n_i)$$

where

u : denotes the number of values attribute X can take

y_i : denotes the number of examples that have the i_{th} attribute value at the column defined by attribute X, and belong to class y

n_i: denotes the number of examples that have the i_{th} attribute value at the column defined by attribute X, and belong to class n.

The algorithm has been used on a variety of tasks, in the standard or a modified form[14,15], with some success[16,17,18]. It has also been compared to different approaches and its performance has been shown to be comparable[19,20]. The use of ID3 for real world applications uncovered various deficiencies in the basic mechanism of the algorithm. For example, studies by Kononenko *et al*[21] have highlighted the deficiency of favouritism towards attributes with a large number of values. Chapter 6 describes further shortcomings as experienced during this research work. Despite the

imperfections, it seems that ID3 is a valuable aid for knowledge elicitation.

4.4 Adaptive Combiners

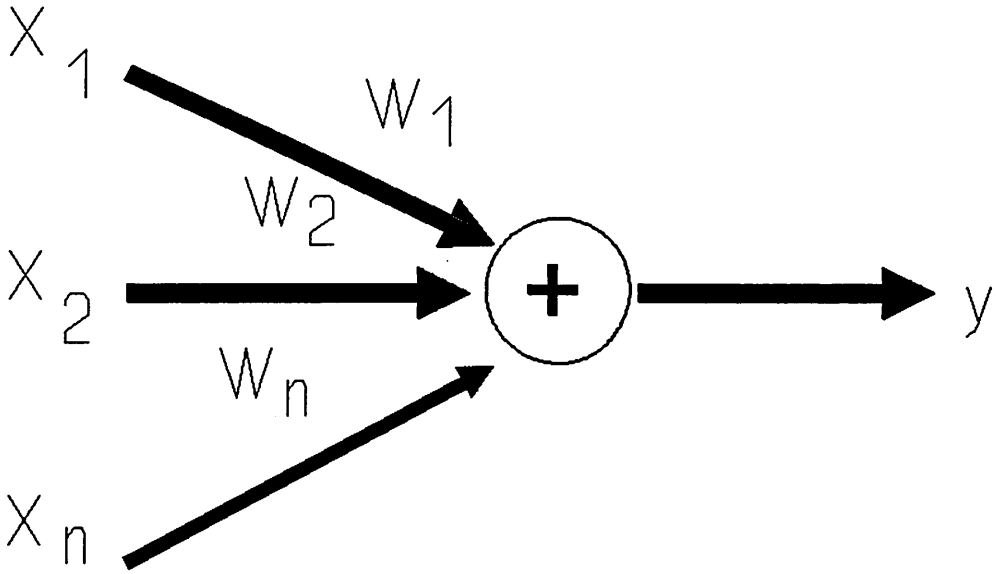
In recent years one class of adaptive architectures, linear combiners, has been used for the design of intelligent systems[22]. These are systems where traditional elicitation techniques fail to provide any rules since the underlying relationships are not known and many of the variables are continuous in nature. Figure 7a illustrates a simple combiner structure. Given knowledge about a particular problem in the form of input attributes it is possible to represent them in vector form as shown below,

$$\underline{x} = [x_1, x_2 \ldots x_n]^T$$

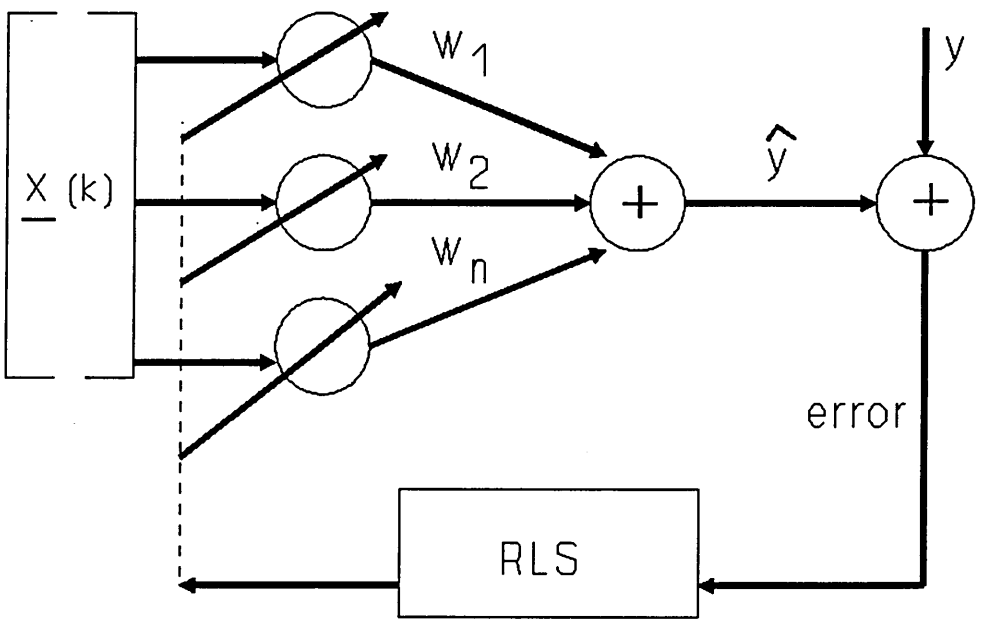
where **n** represents the number of attributes and **T** denotes the matrix transpose operation. Additionally the class y_l is also provided. It is desirable to estimate the weight vector shown below,

$$\underline{w} = [w_1, w_2, \ldots, w_n]$$

in such a way that, when the system is presented with a new set of attribute values, it can predict the correct outcome. In other words, we wish to represent the knowledge relating the attributes to the classes as the weight vector in the combiner. The adaptive combiner structure described here can be thought of as a one layer connectionist network. Adaptive combiners, like neural networks, fall within the first learning criterion as presented by Michie[23]. This criterion states that when a system uses sample data to generate an updated basis for improved performance on subsequent data then learning occurs but the emphasis is on the performance of the system and other aspects of intelligence, such as explanation of reasoning, are neglected.



(a) A simple combiner structure



(b) Adaptive combiner

Figure 7: Architecture of the linear adaptive combiner

The knowledge within an adaptive combiner, or a neural network, is represented by a mathematical function and distributed to a set of weights. Weights act as parameters of the mathematical function, but have no meaning by themselves which makes it rather difficult to assign credit or blame to an individual weight. Adaptive combiners ignore the reasoning characteristic an intelligent system must have and concentrate on the performance. The recursive least squares algorithm is employed for the estimation of the weight vector.

Figure 7b illustrates an adaptive linear combiner where $\underline{x}^T(k)$ is the present set of attribute values, $\underline{w}(k)$ is the weight vector and $\hat{y}(k)$ is the estimated combiner output. From Figure 7b, the estimated output is,

$$\hat{y}(k) = \underline{x}^T(k)\underline{w}(k)$$

The error can be expressed in terms of the desired class value, y(k), and the estimated output, $\hat{y}(k)$ as follows,

$$e(k) = y(k) - \hat{y}(k)$$

The RLS algorithm is used to adjust the weights in order to minimise the mean squared error. It has been shown[24] that the optimal weights, $\underline{W}_{opt}$ are given by the Wiener solution,

$$\underline{W}_{opt} = \underline{\phi}_{xx}^{-1}\ \underline{\phi}_{xy}$$

where $\underline{\phi}_{xx}$ is the auto-correlation function of $\underline{x}$ and $\underline{\phi}_{xy}$ is the cross-correlation function of $\underline{x}$ and y. In the RLS algorithm[25], the present weights, $\underline{w}(k)$ may be expressed in terms of the previous weights by,

$$\underline{w}(k) = \underline{w}(k-1) + \underline{r}_{xx}^{-1}(k)\underline{x}(k)e(k)$$

where $\underline{r}_{xx}$ is an estimate of $\underline{\phi}_{xx}$ given by,

$$\underline{r}_{xx} = \sum_{n=0}^{k} \underline{x}(n)\underline{x}^T(n)$$

$\underline{r}_{xx}^{-1}(k)$ can be expressed in terms of a standard matrix identity by,

$$\underline{r}_{xx}^{-1}(k) = \underline{r}_{xx}^{-1}(k-1) - \frac{\underline{r}_{xx}^{-1}(k-1)\underline{x}(k)\underline{x}^T(k)\underline{r}_{xx}^{-1}(k-1)}{1+\underline{x}^T(k)\underline{r}_{xx}^{-1}(k-1)\underline{x}(k)}$$

This form of RLS has an infinite memory. In other words, the weights are functions of all the training examples. It is useful to introduce a forgetting factor into the algorithm in order to give greater importance to the recent training examples than the old ones. One way of accomplishing this would be to apply a time varying exponential window to the recursions. In this case the above equation is modified to,

$$\underline{r}_{xx}^{-1}(k) = \frac{1}{\lambda}\left(\underline{r}_{xx}^{-1}(k-1) - \frac{\underline{r}_{xx}^{-1}(k-1)\underline{x}(k)\underline{x}^T(k)\underline{r}_{xx}^{-1}(k-1)}{\lambda+\underline{x}^T(k)\underline{r}_{xx}^{-1}(k-1)\underline{x}(k)}\right)$$

where $0 < \lambda < 1$ and usually lies in the range $0.9 < \lambda < 1$. It was mentioned above that adaptive combiners can be thought of as a subset of connectionism. The main difference is the fact that the combiners are linear structures and cannot be directly applied to non-linear systems. However, the non-linearity can be treated by manipulating the attributes, i.e. by using second or third order attributes depending on the degree of non-linearity.

4.5 Neural Networks

Following a period of inactivity neural networks (or alternatively *neural*

computing, connectionism, parallel distributed processing) research was revived resulting in the development of various types of systems. A historical overview of neural research can be found in Pollack[26] and an excellent survey of the different systems in Lippmann[3]. Whereas the symbolic approach is based on an explicit rule set in order to understand a problem, neural networks research targets *hard problems* (i.e. the ones that eliciting rules is hard) so difficult to model that way. One can argue[27], that the two approaches can compliment each other rather than cancel each other out. For instance, for a natural language processing task, parsing sentences may be done by symbolic systems and interpretation may involve neural nets.

Neural networks research has been inspired by the way the human brain operates but the neural network models are not or even try to be exact replicas. Simply, certain similarities exist in terms of the features, the connectivity arrangements and the operation. It is the selection of the connectivity and operation employed that characterises, to a large extent, the type of neural model being used. Although, models differ in detail, each one contains the same basic features. A discussion of these common features and their relation to the popular Back-Propagation architecture is given below.

Any neural model contains a number of **processing units** (or nodes or elements). In the Back-Propagation architecture three types of units exist: input (sensory) units, hidden (associative) units and output (response) units. Each type of unit exists in a **layer**. Back-Propagation networks contain one input layer, one output layer and one or two hidden layers. A single unit can represent a small feature and their distribution over the whole network provides a meaningful entity. The role of the hidden units is to translate the

input patterns into output patterns. The function of each unit is to receive **inputs** (from sensors or other units) and to spread an **output** (to other units or to external agents). Unit inputs and output may be discrete, for example {0,1} or {-1,0,1} or alternatively they may be continuous undertaking values in the interval [0,1] or [-1,+1]. Using the Back-Propagation model a unit can receive a number of inputs but it can only produce one output which can be distributed to more than one unit. The output of a unit is generated by collecting, combining and transforming the inputs. Each unit has associated with it a **combining function**, a **transfer function** and a set of **weights** (See Figure 8). The weights define the influence of an input, the combining function combines the inputs and the weights and the outcome is passed to the transfer function which determines the output. The most common combining function, and the one used in this work, is the **summation function** which calculates a weighted sum of all the inputs:

$$\sum_{i}^{N} W_{ij} * I_i$$

where W_{ij} is the weight between unit i of layer (S-1) and unit j of layer (S) and I_i is the input from unit i. Other combining functions include the maximum function, the minimum function, the majority function and the product function. A number of transfer functions are available (Figure 9), for example, the step, tangent, linear and **sigmoid** (logistic) functions. With the linear transfer function the outcome of the combining function is distributed without alteration whereas with the sigmoid the outcome is transformed to a value between 0 and 1 (i.e. a high and a low saturation limit).

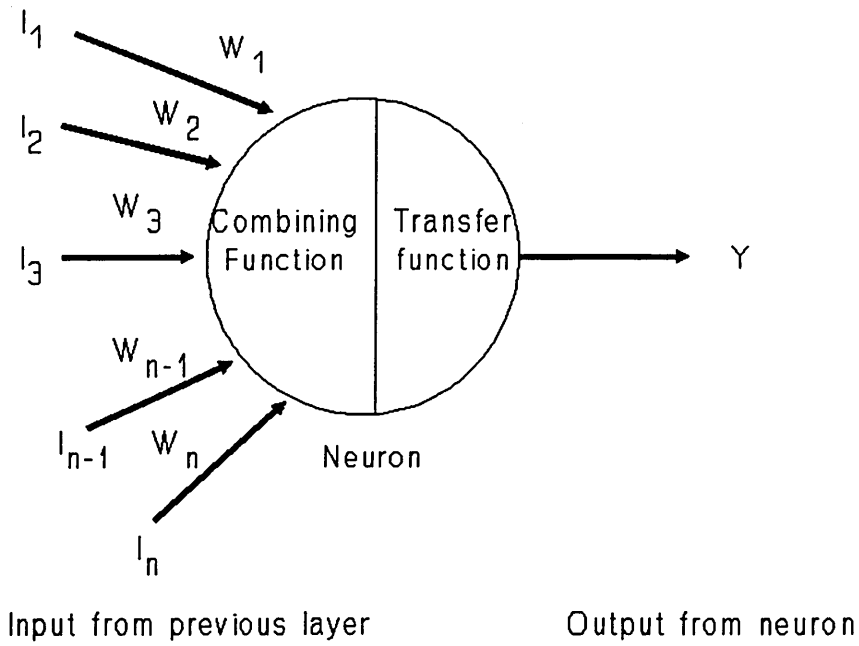


Figure 8: Calculation of the output of a neuron

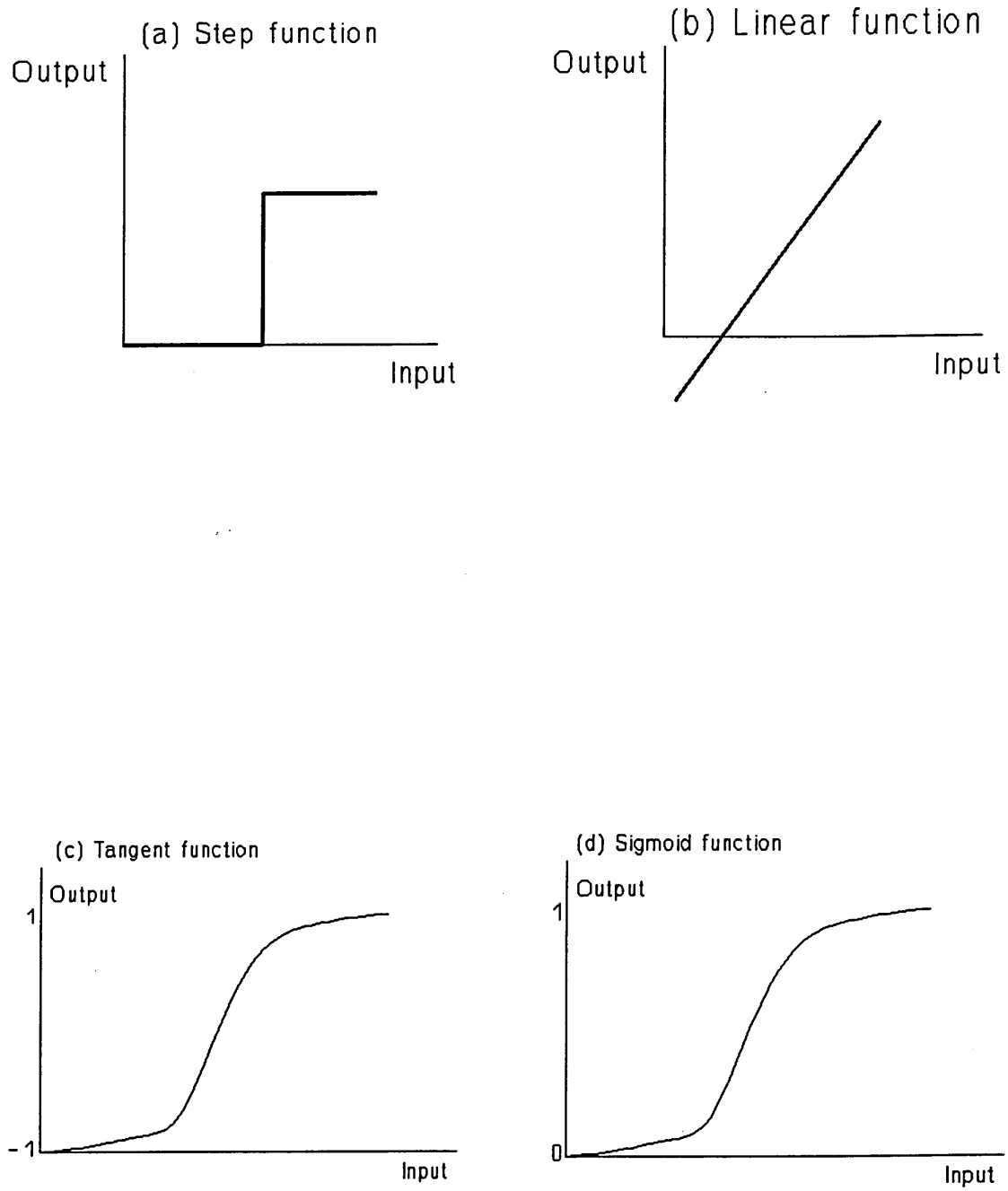


Figure 9: Sample transfer functions

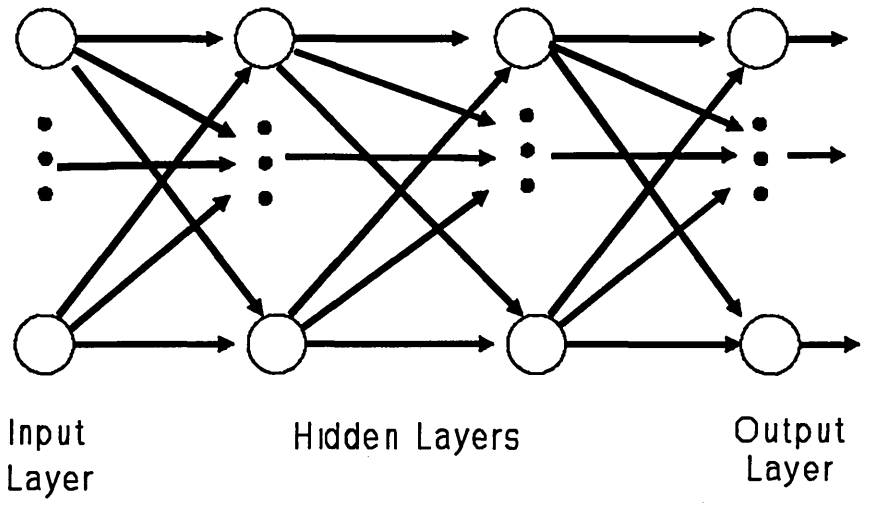


Figure 10: General architecture of a 3-layered feedforward neural network

$$O_j = \frac{1}{1 - e^{\sum_{i}^{allinputs} (W_{ij} I_i + \theta_j)}}$$

where O_j is the output of a unit and θ_j is the weight from a unit that is always on (i.e. holds the values of one). This is referred to as the **bias** and is used in order to offset the origin of the transfer function. The **pattern of connectivity** determines those units that the outcome is passed on. The outcome can be passed to units in the preceding, the following or even the same layer. With the Back-Propagation architecture connections are permitted only between successive layers (feed-forward). Additionally it is fully connected which means that all units of a preceding layer are connected to all units of the succeeding layer (Figure 10). Connectivity, once established cannot change. Having established the basic architecture of a neural model, it is important to understand how one can use the net for learning. The overall objective is the formation of a set of *optimum* weights in order to minimize the global error. For complex problems it is rather difficult to pre-set the weights.Therefore they have to be generated using a **learning method**. Three types of learning exist: unsupervised, supervised and reinforcement. **Supervised learning** is the one used with Back-Propagation. This way the net is presented with inputs and also with the desired output. Each learning method implements a number of algorithms which determine the way the weights change. These algorithms are known as **learning rules**. Back-Propagation networks employ the **error propagation rule**[28] (or generalised delta rule). This rule bypasses the *credit assignment* problem (i.e. which unit is to blame for an incorrect output) by distributing blame to all units. The term **global error** was mentioned previously without actually

specifying it. The global error is defined as half the sum of the squares of all the local errors and is given by:

$$E = \frac{1}{2} * \sum_{k} ((d_k - o_k)^2)$$

where the subscript k indexes all examples of the training set and

$$d_k - o_k$$

is the **local error** which is defined as the difference between the actual output o_k and the desired output d_k. A gradient descent rule, using the knowledge of the local errors, determines how to increment or decrement a current set of weights:

$$\Delta W_{ij}^{[s]} = lcoef * e_j^{[s]} * X_i^{[s-1]}$$

where *lcoef* is a learning coefficient which determines the rate of learning. Since there is no exact knowledge of what a desired output of a hidden unit should be the local error of a hidden unit is calculated using:

$$e_j^{[s]} = X_j^{[s]}(1.0 - X_j^{[s]}) \sum_{k} e_k^{[s+1]} W_{kj}^{[s+1]}$$

where k is over all nodes in the layers above node j. Now a summary of the standard back-propagation learning can be given.

(i) Present inputs to the input layer.

(ii) Calculate the output of each unit.

If a unit is in the input layer no transformation takes place but sometimes scaling might be necessary. Otherwise the sigmoid function is employed.

(iii) Calculate the local error for each unit in the output layer. Then calculate the required changes to the weights and update all corresponding previous

weights.

(iv) Calculate the local error for each unit in the layers below the output layer. Then calculate the required changes to the weights and update all corresponding previous weights.

(v) Repeat until the desired global error has been achieved.

The error-propagation learning rule has been used successfully in numerous applications but it has to be realized that it can also fail. Failure can arise due to non-convergence. Rumelhart *et al*[28] have reported that a neural network failed sometimes to converge during learning of the exclusive or task. The convergence process sometimes gets trapped in a local minimum and the network cannot produce the desired response. Additionally, there are many parameters that have to be pre-set without any prior knowledge of their probable values. For example, the required number of hidden units or the value of the learning rate. Despite that back-propagation is not error free it has been very popular and this led to the invention of several improvements to the standard algorithm. For example, weight decay and the addition of the momentum term. With weight decay the value of each weight of the network is reduced after each run (all input patterns or one input pattern) therefore only often repeated patterns are learned. The momentum term takes into account the previous weight changes effectively filtering out large variations of the error surface. The gradient descent rule becomes

$$\Delta W_{ij}^{[s]} = lcoef * e_j^{[s]} * x_i^{[s-1]} + mom * \Delta W_{ij}^{[s-1]}$$

where *mom* is the momentum constant that determines the effect of past weight changes.

Neural nets are mainly developed on conventional serial computers. The

software for the neural net simulation can be written using programming languages like Pascal or C or another option is to use a spreadsheet. Another way is to purchase neural network demonstration systems which accompany books. For example, the books by Aleksander and Morton[30] or McClelland and Rumelhart[31] include software which can be used as a tutorial of the book or as a stand alone. Alternatively one can purchase commercially available neural network programs (or shells) such as NeuralWorks (Recognition Research), BrainMaker (California Scientific Software) and NeuroShell (Ward Systems Group). These shells allow the users to experiment with a number of network architectures and the values of the various parameters, they offer built-in input/output facilities (e.g. they can import data from spreadsheets or databases), and they provide various statistics (e.g. the change of a particular node). Because it can take several hours or days to train a large network (large in terms of connections), it is beneficial to use a serial computer with add-on accelerator boards capable of performing fast arithmetic operations and a large storage memory.

References

1. Gammack J.G., and Young R.M., *Psychological techniques for eliciting expert knowledge*, In: Research and development of expert systems (Ed. E.S Bramer), Cambridge University Press, 1985.

2. Ryszard S.M., Garbonell J.G., and Mitchell T.M., *Machine learning : An Artificial Intelligence approach*, Vol. 2, Morgan Kaufmann Publishers, 1986.

3. Lippmann R.P., *An introduction to computing with neural nets*, IEEE ASSP Magazine, April, pp. 4-27, 1987.

4. Goldberg D.E., *Genetic algorithms in search, optimisation and machine*

learning, Addison-Wesley, 1989.

5. Michalski R.S., and Negri P.G., *An experiment on inductive learning in chess end games*, In: Machine Intelligence (Eds E.W Elcock, and D. Michie), Willey, 1977.

6. Michalski R.S., and Larson J.B., *Selection of most representative training examples and incremental generation of Vl1 hypotheses : the underlying methodology and description of programs ESEL and AQ11*, Report 867, University of Illinois.

7. Michalski R.S., Mozetic I., Hong J., Lavrac X., *The multi-purpose incremental learning system AQ15 and its testing application to three medical domains*, Proceedings of the AAAI, pp. 1041-1049, 1986.

8. Mirzai A.R., Cowan C.F.N., and Crawford T.M., *Intelligent alignment of waveguide filters using a machine learning approach*, IEEE Transactions, Vol. 37, No. 1, pp. 166-173, 1989.

9. Newstead M.A., and Pettipher R., *Knowledge acquisition for expert systems*, Electrical communication, Vol. 60, No. 2, pp. 115-121, 1986.

10. Quinlan J.R., *An empirical comparison of genetic and decision tree classifiers*, Proceedings of the Fifth International Conference on Machine Learning, Morgan Kaufmann, pp. 135-141, 1988.

11. Quinlan J.R., *Discovering rules from large collection of examples : a case study*, In: Expert systems in the microelectronic age (Ed. D. Michie), Edinburgh University Press, 1979.

12. Hunt E.B., Marin J., and Stone P.T., *Experiments in induction*, Academic Press, 1966.

13. Shannon C.E., and Weaver W., *The mathematical theory of*

communication, University of Illinois Press, 1972.

14. Utgoff P.E., *ID5: An incremental ID3*, Proceedings of the Fifth International Conference on Machine Learning, Morgan Kaufmann, pp. 107-120, 1988.

15. Schlimmer J.C., and Fisher D., *A case study of incremental concept induction*, Proceedings of the Fifth National Conference on Artificial Intelligence, Morgan Kaufmann, pp. 496-501, 1986.

16. Cestnic B., Kononenko I., and Bratko I., *Assistant 86: A knowledge elicitation tool for sophisticated users*, Proceedings of EWSL 87 : Second European Working Session on Learning, Sigma Press, pp. 31-45, 1987.

17. Shepherd B.A., *An appraisal of a decision tree approach to image classification*, Proceedings of the Eighth International Joint Conference on Artificial Intelligence, Vol. 1, pp. 473-475, 1983.

18. Quinlan J.R., *Inductive knowledge acquisition: a case study*, In: Applications of expert systems (Ed. J.R. Quinlan), Turing Institute Press/Addisson Wesley, 1987.

19. Mingers J., *Rule induction with statistical data - a comparison with multiple regression*, Journal of Operational Research society, Vol. 38, No. 4, pp. 347-351, 1987.

20. Ward J.B., Green S.M., and Allan A., *Machine induction in mass spectroscopy*, IEE Colloquium on Application of Knowledge-Based Systems, No. 7, pp. 1-4, 1987.

21. Kononenko I., Bratko I., and Roskar E., *Experiments in automatic learning of medical diagnostic rules*, Technical report, Jozaf Stefan Institute, Ljubljana, Yugoslavia, 1984.

22. Brown K.E., Cowan C.F.N., Crawford T.M., and Grant P.M., *Knowledge-based techniques for fault detection in digital microwave radio communication equipment*, IEEE Journal on selected areas in communications, Vol. 6, No. 5, pp. 819-827, 1988.

23. Michie D., *Machine learning in the next five years*, Proceedings of the Third European Working Session on Learning, Pitman, pp. 107-122, 1988.

24. Wiener N., *Extrapolation, interpolation and smoothing of stationary time series*, Wiley, 1949.

25. Cowan C.F.N., and Grant P.M., *Adaptive filters*, Prentice-Hall, 1985.

26. Pollack J.B., *Connectionism: past, present and future*, Artificial Intelligence review, Vol. 3, No. 1, pp. 3-20, 1989.

27. Rich E., *Expert systems and neural networks can work together*, IEEE Expert, Vol. 5, No. 5, pp. 5-7, 1990.

28. Rumelhart D.E., Hinton G.E., and Williams R.J., *Learning representations by back-propagating errors*, Nature, Vol. 132, pp. 533-536, 1986.

29. Rumelhart D.E., Hinton G.E., and Williams R.J., *Parallel Distributed Processing*, Vol. 1, MIT Press, 1986.

30. Aleksander I., and Morton H., *An introduction to neural computing*, Chapman and Hall, 1990.

31. McCleland J.L., and Rumelhart D.E., *Explorations in parallel distributed processing*, MIT Press, 1988.

Chapter Five

Comparison of Machine Learning Techniques

Eureka, Eureka

Archimedes

5.1 Introduction

The previous chapter introduced the three paradigms (ID3, adaptive combiners, neural networks) which have been used and compared as knowledge elicitation tools in this work. As a result of the protocol analysis the tuning of the filter was divided into two primary tasks. Namely, the tasks of tuning the stopband and passband regions. Additionally, it was established that only the two trimmer capacitors were used for the stopband region. In this chapter results are presented only for this region but the conclusions apply to both regions.

Section 5.2 explains the term example, the nature of the examples used initially and the way that the examples were collected. Section 5.3 details the initial work using ID3 which resulted in the division of the stopband tuning in three knowledge bases (searches). Section 5.4 reports on the comparison of the three learning algorithms (ID3, Adaptive Combiners, Neural Networks) for each of the three searches. The experiments were performed in order to select the classifier that provided good performance with limited training data, and to explore the tradeoffs in terms of training and testing time. The performance merits of the systems are highlighted together with their drawbacks. Suggestions for improving the performance of each technique are also detailed. The problems in applying the learning systems to the alignment of crystal filters are reported in Section 5.4.7. The comparison led to the proposal of employing ID3 for the construction of rules for the first two searches and the need for further work for the third search (Section 5.5)

5.2 Selection of attributes and generation of examples

The three techniques function according to a similar principle. They require a set of examples, referred to as the learning set. Each example is described in terms of attributes, with each attribute in turn specified by a value, together with a class identifier. The purpose of the techniques is to determine the relationships between the attributes which then can be used for classification of other examples. Prior to the generation of the learning set the most appropriate attributes were selected. Attributes are the parameters the operator uses to extract and interpret information from the response characteristic of the filter. Six relevant attributes were identified as having strong significance. These were:

(i) Locations of sharp positive peaks of the waveform (Figure 11, identified as p1, p2, p3, p4, measured in MHz units - horizontal axis).

(ii) Relative magnitudes of sharp negative peaks of the waveform (Figure 11, identified as r1, r2, measured in dBs units - vertical axis).

The attribute selection was based on the transcripts derived form the protocol analysis. The operator's reasoning was revealed by sentences such as "...arrange these peaks into a more reasonable place" and "...pull that peak out of the screen". Further discussions with the operator supported the choice. The second step was to obtain a set of examples. Since a database of examples was not readily available the expert operator was requested to tune a number of filters. Prior to each action taken by the operator the attribute values were recorded manually, together with the decision taken each time.

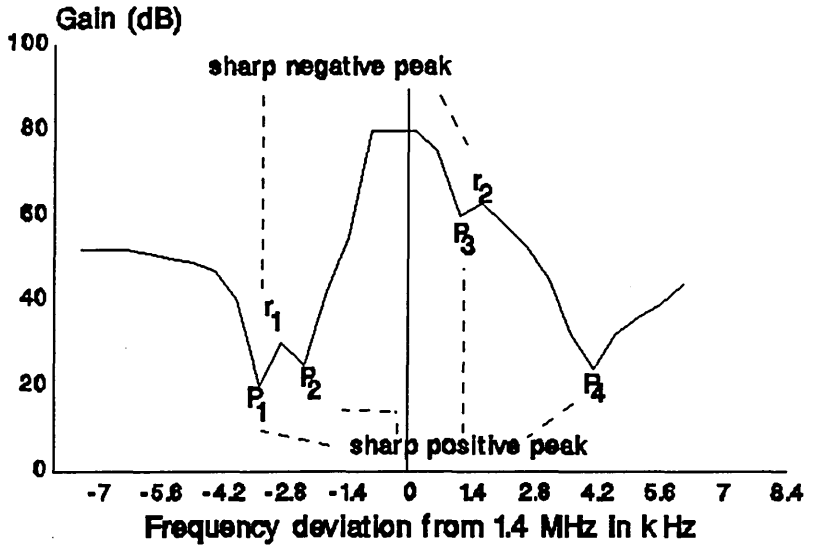


Figure 11: Normalised magnitude response showing the attributes used for the tuning of the stopband. The reference frequency of 1.4 MHz is denoted by zero at the frequency-axis.

The decision being the component, direction and distance used or an indication that no further tuning was required. This approach has been labelled as the tune procedure.

5.2.1 Levels of classification

A typical tuning process for one filter took the following form

p1	p2	p3	p4	r1	r2	decision
1.39934	1.39986	1.40273	1.40310	55	46	C4a0.50
1.39921	1.39969	1.40269	1.40568	62	26	C7a2.00
1.39777	1.39945	1.40520	1.40880	60	68	C4c0.25 (U)
1.39788	1.39954	1.40448	1.40800	56	60	C4a0.50
1.39690	1.39915	1.40638	1.40640	66	66	end

The examples can be interpreted as: " turn the C_4 component anticlockwise, half a turn (first example) and no further tuning is required (last example) when the attributes have the given values". Three observations need to be discussed at this stage. Firstly, each filter's tuning process leads to a number of examples. For the process above this means four examples. Each example is considered on its own without taking into account what happened before or after. This is known as *instance-to-class induction*[1]. Secondly, one has to realise that the decision taken by the expert at each step is not the only option. Other options could have been followed which probably would had resulted in fewer or more subsequent decisions being necessary. This is mainly the case for the 'how far to turn' part and to a lesser degree for the other two parts. An infinite number of actions can be taken. This leads to the problem that for each filter, myriad routes lead to a tuned position. Sometimes though the expert operator realised that a given action was dramatically wrong, as in example three (3) above. These examples were not used in the learning set. Thirdly, it is clear that the first

example points to four decision levels. The operator recognized that adjustment is needed and then he considered which component to adjust and in which direction and by how far.

5.3 Initial empirical results with ID3

Using the tune procedure twelve (12) filters were tuned resulting in forty-seven (47) examples. Thirty-six (36) of these examples were generated using the expert operator and used as the learning set. The rest of the examples (11) were generated using another operator and were used as the testing set. The purpose of testing was to investigate the benefits, if any, of dividing the stopband sub-task into a number of search spaces. The configuration of each search space, ie. what level of classification to represent, was also examined. Four knowledge bases were created employing the learning set and tested on the remaining examples. Each knowledge base was developed feeding the same examples to ID3 but in a different configuration (Table 3). For example, referring to Table 3, configuration 1 had just one search. Each example of the training set could then take one of two classes, either *end-of-process* or *component / direction / distance*. A testing criterion was the number of correct or nearly correct answers given by the system when examples from the training set were used. Another criterion was the number of rules created. The testing results are displayed in Table 4. Some general observations now follow:

(i) All configurations except one had similar number of successes for the component part.

(ii) Irrespective of the configuration there was total success for the direction part.

Table 3 : Search configurations

Configuration	Search1	Search2	Search3	Search4
1	end, component direction distance			
2	end, component direction	distance		
3	end, carry-on	component	direction	distance
4	end, carry-on	component direction	distance	

Table 4 : Testing results using the four configurations

Config.	Number of rules	Correct			End	Carry
		Compon.	Direct.	Dist.		
1	18	1/3	8/8	1/8	5/8	-
2	22	2/3	8/8	1/8	6/8	-
3	27	3/3	8/8	1/8	1/8	8/8
4	25	3/3	8/8	1/8	6/8	8/8

(iii) Except for one instance all system recommendations for the distance to turn were dissimilar to the operator's actions. This is a problem we encountered later on as well.

(iv) The introduction of the *carry-on* class resulted in a better recognition of the state of a tuned response (i.e. correct *end-of-process* for configurations 3 and 4).

The results demonstrated that it is beneficial to introduce search spaces and the 'best' configuration was the one which contained three search spaces:

(i) search space one: to carry-on or to end the tuning process.

(ii) search space two: which component and which direction.

(iii) search space three: how far to turn.

This configuration produced the best success rate but with a relatively higher number of rules than two other configurations.

5.4 Comparison of the three paradigms

For the purpose of comparing the three paradigms a set of examples was collected using the "de-tune" procedure. This process involved a systematic shift of a tuned response to an untuned one. This procedure missed out the heuristics employed by the expert but a more complete set of examples was collected. By complete is meant a learning set which

contains most attribute values likely to arise thus eliminating the possibility of having only extreme or rare values. This was especially valuable in this part of the work in which numerical attribute values were used. This section of the work has been described previously[2].

5.4.1 Generation of de-tuned examples

This work employs, as previously, attributes with continuous attribute values (i.e. numerical format), and it concerns only the stopband region of the response. An algorithmic illustration of the de-tune procedure now follows.

(i) The expert operator was requested to tune the stopband region of the filter's magnitude response.

(ii) The attribute values were recorded, together with the class *end-of-process*.

(iii) The right component (C_7) was kept constant at its optimum position. The left component (C_4) was turned anticlockwise in steps of 0.25 revolutions, resulting in eleven examples.

(iv) Steps (i) and (ii) were repeated.

(v) As step (iii), but this time in a clockwise direction (eleven examples).

(vi) Steps (i) and (ii) were repeated.

(vii) As step (iii), but this time C_7 was turned anticlockwise (eleven examples).

(viii) Steps (i) and (ii) were repeated.

(ix) As step (vii) but in a clockwise direction (six examples).

In this way 43 examples were collected for one filter. Six filters were de-tuned resulting in a total of 258 examples.

5.4.2 Presentation of examples

Previous work resulted in three search spaces for the tuning of the stopband (see Section 5.4). The reader should note that the same examples were

presented to each technique for every search. The examples were introduced to the techniques in an incremental fashion. The number of classes were different in each search. Search one had two classes (*carry-on, end-of-process*), search two had four classes(C_4 and C_7 in clockwise and anti-clockwise direction), search three had eleven classes (distance to turn). Initially eight examples were used in the learning set. They comprised four *end-of-process* and four *carry-on* examples of the same filter. The latter included those examples generated when the components were adjusted to their maximum positions in both directions. Then four more examples were introduced, the ones generated when the components were turned halfway. Finally the four examples which arose when the components were adjusted to their minimal positions were presented. For the second and third search the same examples were presented but with the *carry-on* class replaced by either the component/direction or the distance respectively. The *end-of-process* examples were replaced by those examples generated with the minimum turn for these two searches. At each stage of the procedure the generated set of rules or weights was tested against the learning set (S1), the remaining unseen examples of the same filter (S2) and the unseen examples of the rest of the filters (S3). Finally, the total performance was calculated (TOTAL). Total was determined by testing the rules (or the weights) against all available examples.

5.4.3 Comparison criteria

Machine learning involves generalising from a set of examples and identifying those attributes and attribute values that can be used to discriminate between classes. The quality of generalisation depends heavily on the

selected attributes (sufficient or inadequate ?) and the number of examples present. At this stage of the work the hypothesis was that the chosen attributes were adequate. However, the number of examples necessary was unknown. The objective of the comparison was to identify that technique which used the least number of examples in conjunction with a satisfactory performance. Note that in using the set of examples, either to learn or to test, the assumption was being made that, given a set of attribute values, the only correct action is the one defined by the example. The comparison was then based on two criteria:

(a) The percentage of examples used in the final learning set and

(b) the predictive accuracy of the final learning set.

5.4.4 Search one comparison

Table 5 shows an example and how it was presented to each technique. The exact numbers were presented to ID3 and adaptive combiners. The numbers were scaled between zero and one for the neural net. This scaling is reported in the literature to be beneficiary[3].

Implementing ID3

The following points can be concluded regarding the results obtained using ID3 (Table 6). ID3 is seen to be always capable of predicting accurately those examples presented in the training set (S1). Furthermore, by taking into account the percentage success rate one can conclude that a satisfactory generalisation has been achieved with few examples. Introducing extra examples seems to improve the generalisation even further. Unfortunately this is misleading. Closer inspection of the test results shows that the high success rate was due to the presence of a large number of *carry-on* examples.

Table 5 : Two typical examples and their class representation for each technique (search I)								
Attribute								
P_1	P_2	P_3	P_4	R_1	R_2	ID3	A.C	N.N
1.3875	1.39602	1.412	1.422	55	56	ca-rry	1	1
1.3825	1.3956	1.402	1.423	60	62	end	0	-1

Table 6 : ID3 predictive accuracy (Search I)				
Number of learning examples	(%) Rate of success on...			
	S1	S2	S3	Total
8	100	82	80	81
12	100	81	80	81
16	100	100	93	94
18	100	100	96	97

‡ P denotes a positive peak
‡ R denotes a negative peak
‡ S1 denotes performance for examples of the training set
‡ S2 denotes performance for unseen examples of the same filter
‡ S3 denotes performance for unseen examples of other filters

ID3 predicted successfully the *carry-on* examples but failed to recognise the *end-of-process* ones, i.e. no true classification. The need arose for more *end-of-process* examples to be introduced in the learning set. Those additional examples were taken from the tuning process of another filter. It was found that by increasing the learning set to 18 the objective was achieved with a 96 percent success rate (Row 4 of Table 6).

Implementing adaptive combiners

Obtained results employing the adaptive combiner architecture are displayed in Table 7. The performance of the adaptive combiner also tends to improve through presentation of extra examples with the performance of the training set (S1) being the exception. Unfortunately, like ID3, a large number of *end-of-process* examples were misclassified. Therefore, experiments were carried out to investigate if any improvements in tuning by the combiner could be obtained by following one or more of the next options:

(a) Varying the forgetting factor.

(b) Re-train the combiner with the same learning set.

(c) Introduce another attribute with a constant value of one. This is similar to the biases of back-propagation. It has weights whose values are energised by an input of +1.

(d) Introduce further examples of *end-of-process*.

Implementing the first option

The value of *lambda* (ie. the forgetting factor) was set to values between 0.9 and 0.97. As the forgetting factor increased the learning of the *end-of-process* examples deteriorated. Oddly, the opposite occurred for the *end-of-process* examples of the training set. With the benefit of hindsight this can be

Table 7 : Combiner predictive accuracy (Search I)				
Number of learning examples	(%) Rate of success on...			
	S1	S2	S3	Total
8	87	57	67	67
12	100	77	82	82
16	75	100	91	91

‡ S1 denotes performance for examples of the training set
‡ S2 denotes performance for unseen examples of the same filter
‡ S3 denotes performance for unseen examples of other filters
‡ Total denotes performance for all available examples

attributed to the fact that a wide range of responses can be considered as tuned. Most likely, the examples of the training set belonged to analogous responses slightly different to those of the learning set.

Implementing the second option

The sixteen examples used previously were used again as the learning set. Rather than testing on the whole set of examples it was decided to test each loop only on the whole set of *end-of-process* examples. One loop occurs every time the adaptive combiner sees all the examples in the training set. Re-training stopped when the four *end-of-process* examples of the learning set were recognised as such. This happened after twenty-nine (29) loops but still it did not recognise the *end-of-process* examples of the training set. Then the adaptive combiner was tested against the whole set of examples. The results were as when the sixteen examples were learned in one single pass. The re-learning of the training set only proved beneficial for the learning set and it achieved nothing in terms of generalisation.

Implementing the third option

The introduction of an extra attribute which contained the value 1 showed no advantages towards learning the *end-of-process* examples. All *end-of-process* examples were given *carry-on* classifications.

Implementing the fourth option

Because of the sensitivity of the algorithm towards the order in which the learning examples are introduced, the new examples were placed between the existing ones and not at the end. No changes to the previous results occurred when one or two *end-of-process* examples were presented. Some improvements appeared when three *end-of-process* examples were presented but only for the

learning set. Some further improvement developed with four and five *end-of-process* examples. This was true for the training set as well. Despite the short scale improvements it was concluded that this option on its own does not offer any great advantage.

Implementing a mixture of options

Experiments were carried out to improve the performance of the combiner using a mixture of the available options. Table 8 shows the predictive accuracy of the combiner when the forgetting factor equals 0.9 and the learning set was presented to the combiner 9 times. With this combination the misclassification problem was resolved. All the *end-of-process* examples were correctly classified.

Implementing an additional option

This option involved the identification of any irrelevant attributes. This work was performed by Dr. Mirzai and involved the construction of an attribute matrix with 6 (attributes) times 16 (examples) dimension. Then the correlation matrix and the eigenvalues were calculated. The generated eigenvalues had a small value. This indicated that the problem was overspecified. Two of the attributes (the two negative peaks) responsible for the overspecification were then dropped. The remaining attributes were also scaled between 0 and 100. Results obtained with the reduced set of the scaled attributes are illustrated in Table 9. It is interesting to notice that the combiner performs best when only 8 examples were used. This is mainly due to the fact that when a large number of examples from one filter are shown to the combiner in the learning mode, it cannot recognise examples of the other filters (S3) very well. Also in this experiment the *end-of-process* examples were

Table 8 : Combiner predictive accuracy with adjusted parameters (Search I)				
Forgetting factor : 0.9	Re-learning loops : 9			
Number of learning examples	(%) Rate of success on...			
	S1	S2	S3	Total
16	94	100	91	92

Table 9 : Combiner predictive accuracy with scaled attribute values (Search I)				
Number of learning examples	(%) Rate of success on...			
	S1	S2	S3	Total
8	100	88	90	90
12	100	91	84	85
16	94	98	85	87

‡ S1 denotes performance for examples of the training set
‡ S2 denotes performance for unseen examples of the same filter
‡ S3 denotes performance for unseen examples of other filters
‡ Total denotes performance for all available examples

recognised.

Implementing neural networks

The work with neural nets concerns layered, feed-forward networks learning the classification task by back-propagation. There are some obstacles in using neural networks. One does not how many hidden units are required, nor with what values to initialise the weights etc. At this stage of the research the ease of usage of the technique was mainly explored. The following results arose through the use of a three layered (two hidden layers) network, with a decision threshold of 0.3, a gain of 0.1 and a momentum term equal to 0.0. The algorithm was written in Turbo Pascal and run using a Compaq 386 personal computer. Each architecture iterated for 500 times irrespective of whether convergence happened before the full number of iterations had been completed. Each architecture which learned the examples of the training set was tested against the unseen examples. The number of hidden nodes for each hidden layer was set, arbitrarily, equal to 1 up to 5. Using Table 10, where some results are displayed when eight examples were used, various points can be made:

- Irrespective of the architecture there was a 100 per cent success on the learning set with an average of 117 iterations.
- The prediction performance averaged 72 per cent.
- Ten out of 24 *end-of-process* examples were mis-classified.

Increasing the training set by four examples showed that:

- A number of architectures were unable to learn the training set even with a 1000 runs. The common entity of these architectures was that the number of hidden nodes of the first hidden layer was set to one.

Table 10 : Neural net predictive accuracy (Search I - eight examples)	
(%) Rate of success on the whole set of examples (Total)	Neural net architecture
74	6-1-2-1
76	6-2-2-1
74	6-4-2-1
74	6-1-3-1
74	6-2-3-1
71	6-3-5-1
72	6-4-4-1
74	6-4-5-1

‡ Neural net architecture of A-B-C-D as in 6-4-5-1 denotes A input units, B units in the first hidden layer, C units in the second layer, and D output units

■ The networks that were able to learn the training set took an average of 109 iterations.

■ The average prediction performance increased as well to 78 per cent (Table 11).

■ Again the mis-classification rate of *end-of-process* examples was 50 per cent except when the number of hidden nodes in the first layer was four. That was irrespective of the number of nodes in the second layer. The best true classification was achieved when the (6)-4-4-1 architecture was employed.

Increasing the examples in the learning set to 16 produced fewer architectures able to learn the training set (Table 12). In average it took them 515 iterations. They also produced an average performance of 93 per cent with a misclassification rate of 4 examples out of 24. It became clear that although increasing the size of the learning set can improve performance, having the right architecture is also important.

5.4.5 Search two comparison

Table 13 shows how an example was presented to each technique.

Implementing ID3

The three learning sets were introduced to the ID3 algorithm. Table 14 shows the results obtained. Note that even when eight examples were used the prediction rate was acceptable and that the performance did not improve with the introduction of further examples. This is probably an indication that further attributes are required if better performance was to be achieved. Alternatively, a larger number of examples could have been used in the training set, but at this stage this was not desirable since the comparison of the three techniques using sets of 8, 12, and 16 examples was the main task.

Table 11 : Neural net predictive accuracy (Search I - twelve examples)	
(%) Rate of success on the whole set of examples (Total)	Neural net architecture
77	6-2-2-1
78	6-3-2-1
78	6-4-2-1
77	6-2-3-1
78	6-3-3-1
77	6-4-3-1
77	6-2-5-1
77	6-3-5-1
78	6-4-5-1
78	6-4-4-1

‡ Neural net architecture of A-B-C-D as in 6-4-5-1 denotes A input units, B units in the first hidden layer, C units in the second layer, and D output units

Table 12 : Neural net predictive accuracy (Search I - sixteen examples)	
(%) Rate of success on the whole set of examples (Total)	Neural net architecture
93	6-4-2-1
91	6-4-3-1
93	6-5-2-1
93	6-5-3-1
93	6-6-2-1
93	6-6-3-1
93	6-4-4-1
93	6-5-4-1
93	6-3-5-1
93	6-4-5-1
94	6-5-5-1

‡ Neural net architecture of A-B-C-D as in 6-4-5-1 denotes A input units, B units in the first hidden layer, C units in the second layer, and D output units

Table 13 : Four typical examples and the representation of their class using each technique (search II)								
Attribute...						ID3	A.C	N.N
P_1	P_2	P_3	P_4	R_1	R_2			
1.372	1.383	1.405	1.415	45	62	C_4a	-1 0	-1 -1
1.393	1.404	1.412	1.423	54	55	C_4c	1 0	-1 1
1.381	1.392	1.418	1.429	49	52	C_7a	0 -1	1 -1
1.387	1.398	1.425	1.436	54	49	C_7c	0 1	1 1

Table 14 : ID3 predictive accuracy (Search II)				
Number of learning examples	(%) Rate of success on...			
	S1	S2	S3	Total
8	100	100	88	91
12	100	100	88	91
16	100	100	88	91

‡ P denotes a positive peak
‡ R denotes a negative peak
‡ S1 denotes performance for examples of the training set
‡ S2 denotes performance for unseen examples of the same filter
‡ S3 denotes performance for unseen examples of other filters
‡ Total denotes performance for all available examples

Implementing adaptive combiners

When the three learning sets, in their original form, were presented to the combiner the results were very poor (Table 15). It was thought that the combiner needs to encounter examples which can act as *reference points*. That role was played by the introduction of *end-of-process* examples to the learning set, once again. This way the combiners were trained to indicate the *end-of-process*, as well as which screw to adjust and in what direction. As for search one, it was found necessary to present the combiner with a reduced number of attributes (the four positive peaks) and to scale the values between 0 and 100 in order to improve the performance. The first training set contained 5 examples, i.e. one *end-of-process* plus four examples when the screws were mal-adjusted to their maximum positions. Then the examples corresponding to the minimum positions of the screws were added to the learning set (i.e. 9 examples all together) and finally the examples corresponding to the half-way mal-adjustments of the screws plus one more *end-of-process* example were added resulting in 14 examples. The performances of the combiners for the three new learning sets are summarised in Table 16. Introducing more examples from the same filter resulted in an acceptable performance when testing examples from the filter that the training examples were taken from. Instability in the learning occurred for examples generated from different filters. Again, the combiners successfully recognised all the *end-of-process* examples but their total percentage rate of success was not as high as for the ID3 algorithm. The reason behind the much lower overall performance of the combiner lies in the low percentage rate of success when examples of other filters are tested (S3 - Table 16). It is known that two filters of the same

Table 15 : Adaptive combiner predictive accuracy (Search II)				
Number of learning examples	(%) Rate of success on...			
	S1	S2	S3	Total
8	25	35	39	38
12	33	33	39	38
16	38	39	38	38

Table 16 : Combiner predictive accuracy with scaled attribute values (Search II)				
Number of learning examples	(%) Rate of success on...			
	S1	S2	S3	Total
5	100	93	75	78
9	100	93	81	83
14	93	95	58	64

‡ S1 denotes performance for examples of the training set
‡ S2 denotes performance for unseen examples of the same filter
‡ S3 denotes performance for unseen examples of other filters
‡ Total denotes performance for all available examples

family are not identical. Tolerancing errors and parasitic effects result in different attribute values. It seems that the combiner could not handle these situations while the selected attribute cut-off values and therefore lines using ID3 divided the 6-dimensional space properly.

Implementing neural networks

The three layer networks produced an average performance of 76 per cent with a (6)-3-3-2 architecture gaining the highest performance (80% with 8 examples). Note that adding an extra node to either layer did not produce a better performance. By increasing the examples the performance improved with architecture 6-5-3-2 reaching the highest performance (92%) using 16 examples. Irrespective of the number of examples and number of nodes used the nets produced a better performance for the direction to turn rather than the component to be used. Table 17 displays a sample of results.

5.4.6 Search three comparison

Table 18 shows the way that examples were presented to the techniques.

Implementing ID3

Problems arose when ID3 was implemented for search three. It is not reasonable to expect a prediction of, say, 2.25 when only examples with 0.25 and 2.75 classes were presented. This implied the necessity of a large training set consisting of all examples of one filter. However, due to the large number of classes (11) together with the relative small number of examples (43), the problem of bushy, unstructured decision trees arose. This resulted in a very poor performance. Even the introduction of a larger training set would not ensure success. The ID3 algorithm was originally constructed to deal with binary classification and it seems that better performances are achieved with

Table 17 : Neural net predictive accuracy (Search II)				
Number of examples	(%) Rate of success on...			
	S1	Component	Direction	Architecture
8	72	72	98	6-4-3-2
8	80	80	97	6-3-3-2
8	77	77	98	6-3-4-2
12	77	77	98	6-3-2-2
12	81	81	97	6-3-3-2
12	74	74	97	6-5-5-2
16	93	93	98	6-5-3-2
16	92	92	97	6-8-4-2
16	87	87	97	6-3-3-2

‡ S1 denotes performance for examples of the training set
‡ Neural net architecture of A-B-C-D as in 6-4-5-1 denotes A input units, B units in the first hidden layer, C units in the second layer, and D output units

Table 18 : Representation of class for each technique (search III)

ID3 Class	Adaptive combiner	Neural network
0.25	0.25	0 0 0 1
0.50	0.50	0 0 1 0
0.75	0.75	0 0 1 1
1.00	1.00	0 1 0 0
2.00	2.00	1 0 0 0
2.75	2.75	1 0 1 1

a low number of classes. The inability of ID3 to perform successfully when a large number of classes are present was the main factor in deciding to split the tasks into three separate searches, as reported previously.

Implementing adaptive combiners

The main advantage of the combiner and neural net architecture over that of ID3 is due to their capability of producing continuous output. For this search space experiments were carried out with the original learning sets. When the reduced set of attributes and the scaled values were used the combiner performance improved. Notice that *end-of-process* examples were used once again. The combiners were trained on the exact values of mal-adjustments for both screws. Figures 12a and 12b show the correct mal-adjustment levels for screws C_4 and C_7 respectively. Figures 12c and 12d illustrate the output of the combiners when 5 learning examples were used. Figures 12e and 12f show the same outputs when 9 learning examples were used and finally figures 12g and 12h show the outputs with 14 learning examples. With this limited number of examples the combiners have managed to reach the desired outcomes (Figure 12g and 12h) although not to 100 per cent accuracy. In order to improve the performance of the combiner for this search space, it would be necessary to include learning examples generated when both screws are mal-adjusted together.

Implementing neural networks

Unfortunately, the three layer network did not produce very good results even when a large number of nodes were used. For that reason when over one hundred nets were run further investigation was suspended. An interesting, and somewhat predictable, fact arose with the use of the nets. Increasing the

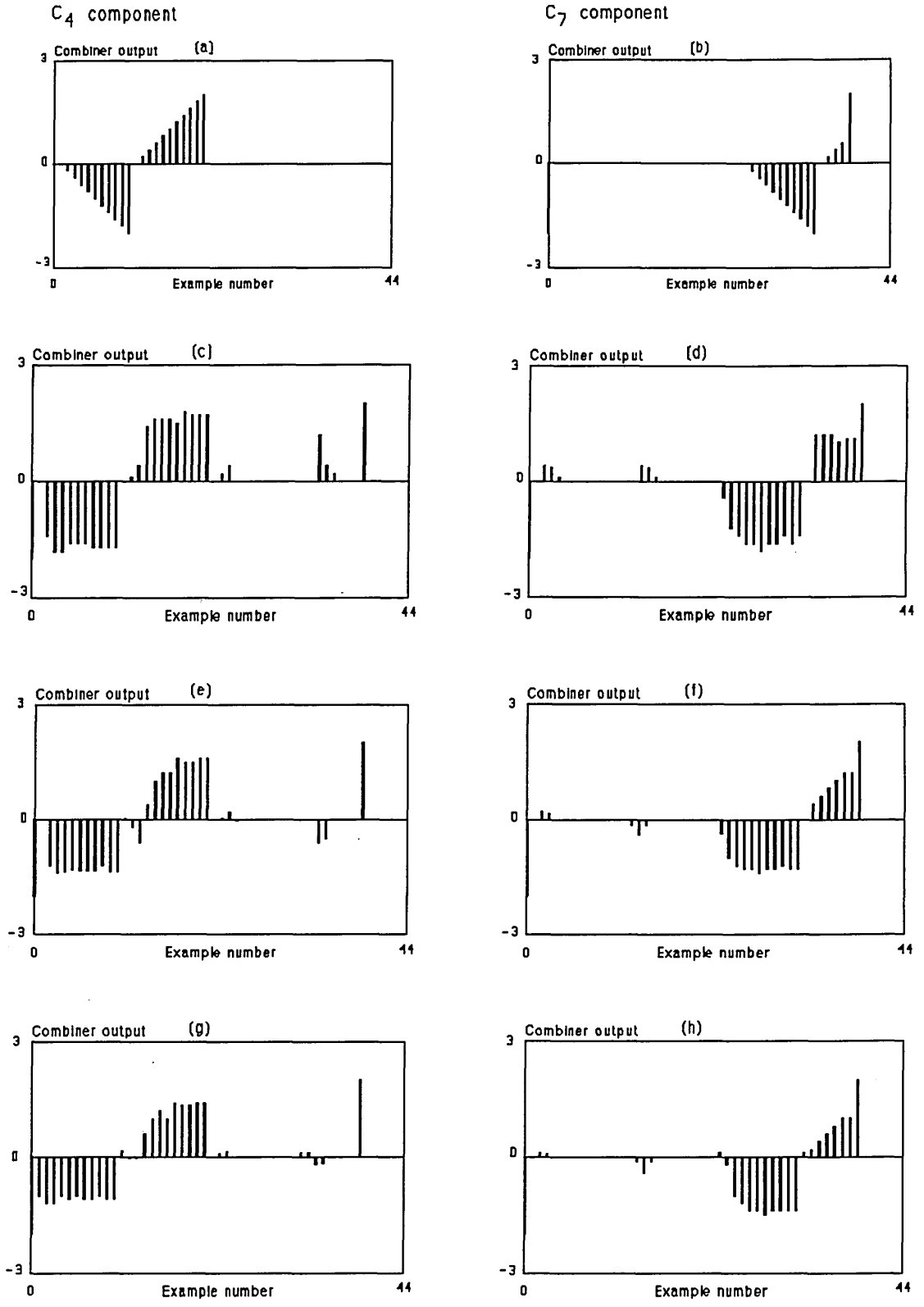


Figure 12: Output of the combiners (Search III)

hidden nodes drastically improved (Table 19) the performance for each individual node. The nets had started to behave as 'look-up' tables.

5.4.7 Problems encountered

The experiences gained in implementing the three techniques are presented in terms of learning, testing and learning refinement. This work has been published[4].

During the learning phase various problems arose due to the structure of the training set. To improve the adaptive combiner performance it was necessary to manipulate the attribute set. The manipulation took the form of scaling the attribute values and/or the elimination of certain attributes. The scaling of the attribute values was important. Without proper scaling an ill-conditioned problem was created in terms of the auto-correlation matrix in the RLS algorithm[5]. It was possible to find if the problem was ill-conditioned by using eigenvalue analysis[6]. Initial work with neural networks and ID3 employed examples generated while tuning a number of filters. When both techniques were tested using unseen examples ID3 performed better. Neural nets failed to classify correctly a number of test examples. Those examples contained at least one attribute with a value previously found in an example with a different classification. Because of the large range of numerical values each attribute can take, a different set of learning examples was required which included all likely values or the extreme values (i.e. maximum and minimum). This was also necessary when using the adaptive combiners. Another obstacle was that learning with neural nets was time consuming. Additionally, unlike adaptive combiners or neural networks which can provide continuous output ID3 had to be presented with examples covering all eleven

Table 19 : Neural net predictive accuracy (Search III)					
Number of examples:39	Number of correct predictions on node..				Architecture
	1	2	3	4	
	20	0	0	0	6-1-3-4
	25	3	6	0	6-2-3-4
	21	22	5	0	6-4-5-4
	29	28	16	8	6-20-10-4
	23	16	9	3	6-10-20-4

‡ Neural net architecture of A-B-C-D as in 6-4-5-1 denotes A input units, B units in the first hidden layer, C units in the second layer, and D output units

classes for the third search. The large number of classes meant that the examples were less representative with the consequence of poor performance. Another problem was the inability of ID3 to mix numeric and symbolic attribute values. This created a problem during testing as will pointed out later.

Testing the rules generated by ID3 was more time consuming than testing the two other techniques. More importantly, though, was that in running the ID3 algorithm, examples with unknown attribute values could not be used when numerical values were employed. A notation to indicate that an attribute value was not significant was available but not to indicate that an attribute value was unknown. It was then impossible to use both numeric and symbolic descriptions for an attribute. If at any point an attribute value was requested and this value was unknown then the system failed completely. Using the other two techniques this could not happen. Unknown values were presented with a constant. During testing the combiner or the net did not perform appropriately but they did not fail.

The presentation of attributes holding numerical values to ID3, caused the following problem. The enlargement of the learning set resulted in a slight change of the threshold values of the decision tree. This led to different classifications of a number of testing examples. In addition the presentation of new examples, resulted to new attributes being introduced or old attributes being excluded from the newly generated decision tree. Therefore the *architecture* of the decision tree generated by ID3 is dependable on the examples of the training set. The introduction of further examples to the adaptive combiner or the neural net did not alter the architectural structure

but strengthened or weakened the individual weights.

5.5 Discussion of the comparison results

Although the three techniques are different, a comparison was possible. The main difference between the three techniques, apart from the algorithmic approach used, is that adaptive combiners and neural nets learn in an incremental fashion while ID3 sees all the examples at the same time. ID3 performed slightly better than the other two for the first two searches. For the third search ID3 failed significantly. The use of ID3 for the first two searches was elected. The decision was based on the following advantages of ID3, as seen by the author.

√ An expert system cannot ever be completed. Such systems should expand their knowledge through time. The augmentation of the learning set by presenting new examples demonstrated that running ID3 was faster. Neural networks took a long time to train. Some architectures took up to 17 hours to train.

√ ID3 always gives correct predictions for the examples used in the learning set. This is not guarantee with the other two techniques.

√ ID3 generated decision trees which can be transformed into production rules. These rules can be used directly to explain the relationships between the attributes and the decisions made. With weights a direct explanation is not feasible. Some work towards this has been reported[7,8].

√ ID3 gave slightly better results with less manipulation of parameters and without the need to worry about the order of introduction of the examples. With adaptive combiners a lot of time was spent in experimenting with parameters. The problem with neural nets was the absence of any theory for

determining the architecture.

From the preceding sections it can be inferred that the adaptive combiner performed well for the third search. At the moment the discussion about the third search will be suspended. The next chapter introduces the way the ID3 problems were resolved.

5.6 Conclusions

The tuning of the stopband and the passband regions were to be treated independently. It was decided each region to be divided into three search spaces. The comparison of the three algorithms led to the use of ID3 for the first two search spaces of both regions. Furthermore, it was decided to research further the use of neural networks for the third search in the future.

References

1. Michalski R.S., Carbonell J.G., and Mitchell T.M., *Machine learning: an Artificial Intelligence approach*, Vol. 2, Morgan Kaufmann, 1986.

2. Tsaptsinos D., Mirzai A.R., and Jervis B.W., *Comparison of machine learning paradigms in a classification task*, Proceedings of the Fifth International Conference of Artificial Intelligence in Engineering, Vol. 2, Computational Mechanics publications, pp. 323-339, 1990.

3. Lapedes A., and Farber R., *How neural nets work*, Neural Information Processing Systems (Ed. D.Z. Anderson), American Institute of Physics, pp. 442-456, 1982.

4. Tsaptsinos D., Jervis B.W., and Mirzai A.R., *Learning by analytical methods or induction - a case study*, IEE Colloquium on Machine Learning, Digest No. 117, pp. 10/1-10/3, 1990.

5. Cowan C.F.N., and Grant P.M., *Adaptive filters*, Prentice-Hall, 1985.

6. Widrow B., and Stearns S.D., *Adaptive signal processing*, Prentice-Hall, 1985.

7. Mirzai A.R., Cowan C.F.N., and Crawford T.M., *Learning using pattern recognition and adaptive signal processing*, In: Artificial Intelligence - concepts and applications in engineering (Ed. A.R. Mirzai), Chapman and Hall, 1990.

8. Gallant S.I., *Connectionist expert systems*, Communications of the ACM, Vol. 31, No. 2, pp. 152-168, 1988.

Chapter Six

Further Work With ID3

6.1 Introduction

During the comparisons of the performance of the three classifiers no attempt was made to achieve high performance but experience gained through the experiments was employed at a later stage. The representation of the input examples, of the output descriptions and the available knowledge (i.e. number of examples) influence the success of any machine learning system. Using ID3 in particular the actual learning time is negligible (a matter of seconds) but the most critical and time consuming part is the one of example selection. Chapter 6 presents the work taken to identify the attributes and their format to be used for each search.

The use of attributes with logical values was selected mainly for two reasons:

- A substantial set of examples was generated but when the attributes had numerical values huge possible combinations between attribute values were missing.
- The unsatisfactory performance with numerical valued attributes when testing with unseen examples.

6.2 ID3 problems

The employment of the de-tune data for the comparison of the three techniques, as described in the previous chapter, served the purpose of comparing machine learning techniques. The ID3 technique was selected for the first two searches. Problems arose due to the use of numerical attribute values. This resulted in a problem associated with the cut-off point. The algorithm produced rules of the form *'if attribute X is less than cut-off point T then...'*. The cut-off point, which took values such as 1.39765, was calculated

by using those values that were currently present in the learning set. The introduction of more examples would probably result in different cut-off points. But in the filter tuning task values such as, say, 1.39765 and 1.39766 can be considered as same whereas the ID3 algorithm regards them as two different entities. Therefore it was important to develop a way of relating values that were close.

Using the de-tune data one could attempt to generate the whole set of possible examples and then present it to the algorithm. It was though the intention from the start to use data generated through the tune procedure. This way the expert's knowledge was to be utilised. Unfortunately it was impossible to produce an entire set of examples since these examples should have included every possible numerical value each of the attributes could have taken.

Problems using ID3 as stated above had to be solved before proceeding any further. In this chapter a report is given on results obtained in an attempt to identify any advantages in using one attribute presentation form over another. The investigation involved the evaluation and comparison of decision trees produced by using logical and numerical attribute values for the first two searches. This work has been reported elsewhere as well[1].

6.3 Further selection of attributes and generation of examples

It was considered that the inclusion of further attributes might be helpful in order to decrease the number of empty and/or clash situations. In total seven more attributes were introduced. These took the form of the six differences

between positive peaks, for example peak1-peak2, peak3-peak4, and the difference between the two negative peaks. These attributes were introduced because not only the position of each individual peak was regarded as important but also the peak's position in relation to where the rest of the peaks are. A new set of examples was collected. This time the tune procedure was employed. Therefore, the operator was requested to tune a number of filters and the data were recorded as previously. In this way 34 filters were tuned (only the stopband) resulting in 138 examples.

6.4 Generation of logical values

Schemes have been proposed[2] which attempt to define supplementary cutpoints for each cut-off point. Producing such confidence intervals enhances the classification of examples with values near the cut-off points. An alternative scheme was followed in this work. Instead of using the raw numerical values a transformation was applied. The numerical values were placed into ranges which were given logical names. The term *logical* is borrowed from the ID3 literature and simply means a linguistic term, similar to fuzzy predicates of the fuzzy set theory[3]. Due to the absence of *a priori* knowledge for determining the ranges within which attribute values must lie for the filter to be considered tuned, the membership was calculated as below:

- Collect all those examples with an *end-of-process* as their class
- Calculate the mean (m) and the standard deviation (sd) value of each attribute.
- For each attribute determine the range (m-sd.m+sd). This range represents all those numerical values an attribute can have and be considered to be tuned. Label the range as 'ok'.

- Determine the rest of the ranges. For example values within (m-2sd.m-sd) are labelled 'close-left' etc. (Figure 13)

That way 8 (ok, far-left, far-right, close-left, close-right, left, right, absent) or 4 (ok, left, right, absent) logical values were generated and assigned to each numerical value. Note that the eighth (or the fourth) logical value takes the label 'absent'. This label was used when a value for an attribute could not be determined (ie. absence of a peak) and not because it was unknown. This way three attribute formats were available for each search space (ie. numerical, 8-logical, 4-logical).

6.5 Criteria for the evaluation of decision trees

The evaluation and comparison was based on the following criteria.

(i) Percentage errors on classifying unseen examples

(ii) Number of branches in the decision tree

(iii) Number of rules in rule base

(iv) Number of clash labelled leaves

(v) Number of empty labelled leaves

(vi) Total number of preconditions in rule base

The first criterion assessed the performance of a decision tree in terms of accuracy on classifying unseen examples. This indicated how good the generalisation was (i.e. predicting future performance). The rest of the criteria are of secondary importance and can be applied in order to determine the complexity and intelligibility of a decision tree. Figure 14 displays a decision tree and Table 20 the equivalent set of rules. They both illustrate the terms

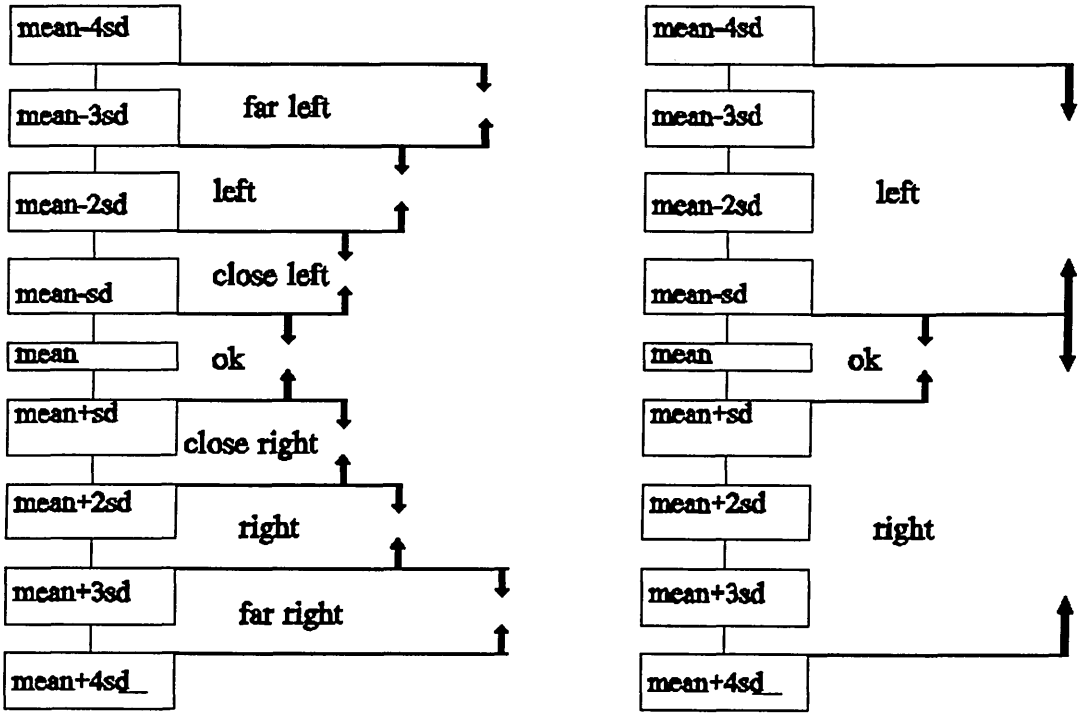


Figure 13: Distribution of logical values. The left hand side shows the distribution of the 8 logical values (the 8th logical value is *absent*). The right hand side shows the distribution of the 4 logical values (the 4th logical value is *absent*).

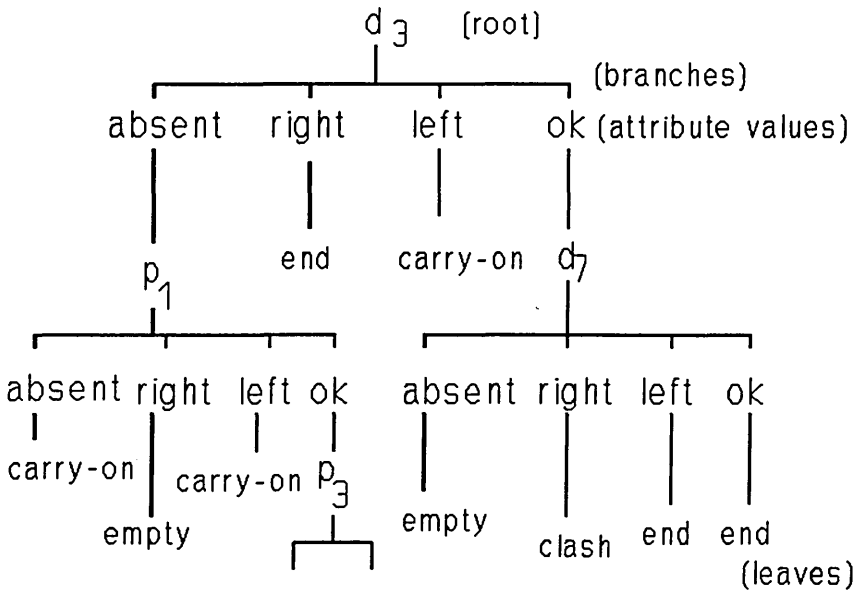


Figure 14: Subset of a decision tree generated with four-valued logical attributes

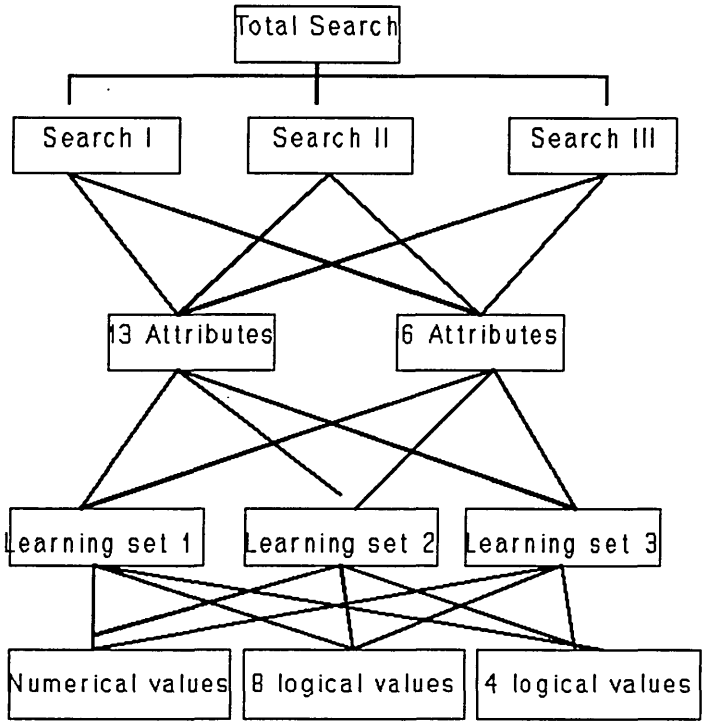


Figure 15: Configuration of generated decision trees

Table 20 : Set of rules produced using the decision tree of Figure 14	
IF d_3 is absent AND p_1 is absent THEN class is carry-on	IF d_3 is ok AND d_7 is ok THEN class is end
IF d_3 is absent AND p_1 is left THEN class is carry-on	IF d_3 is ok AND d_2 is left THEN class is end
IF d_3 is right THEN class is end	IF d_3 is left THEN class is carry-on

Table 21 : Configurations key

Config. Number	Description	Number of attributes	Attributes used
F1	numerical attributes	13	$p_1..p_4$ $r_1..r_2$ $d_1..d_7$
F2	4 logical-value attributes	13	
F3	8 logical-value attributes	13	
F4	numerical attributes	6	$p_1..p_4$ $r_1..r_2$
F5	4 logical-value attributes	6	
F6	8 logical-value attributes	6	

6.6 Presentation of tuned examples

The object of this part of the work was the identification of the 'best' configuration for the first two search spaces. By configuration is meant the choice of attributes to be used and their format (ie. numerical or logical). The six configurations used are summarised in Table 21. To test how well the six configurations measure up to the criteria, the available examples were divided into three randomly chosen batches. The first batch included 42 examples, the second 43 and the third 53. Initially, the first batch was used as the training set and the other two as the testing set (Test 1). That was followed by introducing the second batch to the training set which was then tested against the third batch (Test 2). Both tests were evaluated for all configurations for each search space. In total, thirty six decision trees were generated, viz. eighteen per search space (Figure 15).

6.7 Evaluation of results and discussion (Search One)

<u>Percentage errors on classifying unseen examples</u>

Table 22 shows the results for each configuration for both tests, expressed as the percentage error of mis-classification. Observing Table 22, one can establish the following:

(i) All performances, but one, improve as the size of the training set increases.

(ii) The amount of classification improvement varies between configurations. Trees generated using logical value attributes seem to perform better than those produced using numerical ones. The drawback of numerical value

Table 22 : Misclassification errors (Search I)

Configuration	Test-1 (% error)	Test-2 (% error)	Classification improvement (%)
F1	42.7	41.5	1.2
F2	31.3	22.6	8.7
F3	20.8	28.3	-7.5
F4	42.7	41.5	1.2
F5	31.3	26.4	4.9
F6	27.1	22.6	4.5

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set

decision trees is their inability to handle examples with absent attribute values.

(iii) Upon increasing the number of attributes no major differences were noted with configurations F5 and F6 in terms of improvements in their classification capabilities.

(iv) With thirteen attributes one can see that the performance improves further with the F2 configuration.

Number of empty labelled leaves

For the algorithm to be effective, the number of situations where knowledge (i.e. examples) has not being provided and hence nothing can be learned must be kept to a minimum. To illustrate the concept of emptiness consider Figure 14. Such a situation arises when attribute D3 takes the value absent. If that is true then attribute P1 has to be considered. When P1 takes the value right then the system will respond with the message empty, indicating the lack of knowledge of what to advise. If the number of empties is large, the performance will be poor when testing with such examples.

For each of the six configurations three decision trees were generated by increasing the learning set with the addition of the third batch. Table 23 shows the results. It is worth noting the following:

(i) When numerical values are used there are not empty situations.

(ii) Increasing either the number of attributes (from 6 to 13, compare configurations F4, F5, F6 versus F1, F2, F3 respectively), or the number of logical values each attribute can take (from 4 to 8, compare configurations F2, F5 versus F3, F6 respectively), an increasing number of empty situations is generated. The reason is that the use of a large number of attributes or

Table 23 : Number of empties per configuration (Search I)			
Configuration	Test-1	Test-2	Test-3
F1	0	0	0
F2	6	15	29
F3	16	66	100
F4	0	0	0
F5	6	14	24
F6	20	58	69

Table 24 : Number of clashes per configuration (Search I)			
Configuration	Test-1	Test-2	Test-3
F1	0	0	0
F2	0	3	2
F3	0	3	2
F4	0	0	0
F5	0	4	8
F6	0	4	6

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set
‡ Test 3 denotes three learning sets of data

attribute values rendered the learning set less representative and nothing was gained by the introduction of further examples.

Number of clashes

Similarly, the number of clashes has to be kept to a minimum. A large number of clashes indicates the need for the introduction of further attributes or examples for the algorithm to be able to discriminate between examples. A situation where the system will respond with the message that clash is present, it is illustrated in Figure 14. It occurs when attribute D_3 takes the value ok and attribute D_7 is assigned the value right. The configurations were tested as before and Table 24 shows the results obtained. Notice the following:

(i) The absence of any clashes when numerical values are used is noticeable. This was to be expected. By definition, a clash occurs when two (or more) examples have the same attribute values but are classified differently. This is unlikely to occur when numerical values with six significant figures are employed.

(ii) When the number of attributes is kept small the introduction of more examples results in an increase in clashes. (iii) When the number of attributes is increased, the number of clashes tends to stabilise, irrespective of the number of values an attribute takes.

Number of nodes

Large, bushy trees reduce the intelligibility of the results and increase the execution time. Table 25 shows the results obtained when testing the six configurations. No attempt was made to perform any kind of pruning[4] or to compare various selection criteria[5]. The following observations can be made:

Table 25 : Number of nodes per configuration (Search I)			
Configuration	Test-1	Test-2	Test-3
F1	3	11	23
F2	21	45	77
F3	33	105	161
F4	3	13	23
F5	21	45	69
F6	41	97	121

Table 26 : Number of preconditions per configuration (Search I)			
Configuration	Test-1	Test-2	Test-3
F1	2	18	51
F2	21	53	106
F3	23	65	132
F4	2	19	37
F5	21	45	66
F6	31	57	84

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set
‡ Test 3 denotes three learning sets of data

(i) Decision trees generated using numerical attributes produce a smaller number of nodes, irrespective of the number of attributes used.

(ii) Using logical attributes created much larger trees, especially when the number of attribute values increased.

Number of preconditions in rule base

This criterion has been suggested[6] in order to measure the generality of the entire set of rules. It has been mentioned that any decision tree can be transformed into a set of rules. For example, in Figure 14 the rule *if* D_3 *is absent and* P_1 *is absent then carry-on* can be extracted from the decision tree. This rule has two preconditions. The total number of preconditions in the rule base can then be measured. Table 26 shows the results obtained. A list of remarks now follows:

(i) As the number of examples in the learning set increases, the total number of preconditions increases as well, resulting in less efficient execution timing. This is true irrespective of the number of attributes used but the rate of increase is smaller when the number is kept small.

(ii) Increasing the number of values of the attribute resulted in a greater number of preconditions. This was anticipated since the algorithm has no means of determining if it is necessary to branch for all defined values of an attribute. Perhaps, in some cases various attribute values are relevant, yet the rest may not be.

Number of rules

Every leaf of a decision tree corresponds to a rule of the form *if X1 and X2 and... and Xn then Y* where the X's are the branches and Y is the class of the leaf. By measuring the number of rules extracted from a decision tree the

goal of achieving the minimal set of rules representing the domain was reached. Table 27 shows the results obtained by transforming each tree to a collection of rules. The following comments can be made:

(i) It appears that the introduction of extra attributes, as in configurations F1, F2 and F3, diminishes any benefits (i.e. in most tests more rules were generated).

(ii) Noticeable are the identical results obtained when numerical attributes are used (F1, F4), whereas the difference in the number of rules when logical attributes are used was minor.

(iii) Comparing the results with a view to the attribute format one can deduce that numerical attributes produce less rules than logical attributes. The rate of increase of the number of rules though was much greater as the number of examples increased. For example, considering the F1 and F2 configurations (both have 13 attributes) one can see that the number of rules of F1 tripled (200%) from test-1 to test-2 and doubled (100%) from test-2 to test-3 whereas the number of rules of F2 increased with a lower rate (60 and 56.25 per cent) in both tests.

(iv) Decision trees generated using logical attributes with four values produce less rules than when eight logical values were used. This became more significant as the learning set expanded.

The reader should note that since the introduction of the algorithm the transformation of a decision tree to a set of rules has received much attention. The objective of the proposed schemes[7,8] is to produce a minimal set of rules, which in turn affect the number of preconditions and nodes, but in

Table 27 : Number of rules per configuration (Search I)

Configuration	Test-1	Test-2	Test-3
F1	2	6	12
F2	10	16	27
F3	13	23	39
F4	2	7	12
F5	10	16	20
F6	16	23	31

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set
‡ Test 3 denotes three learning sets of data

this work the results discussed were obtained using the primitive transformation.

6.8 Selection of configuration for search one

Taking into account all the criteria with equal weighting attached to each suggested the use of numerical attributes since they produced smaller trees with fewer clashes etc. However, the most important criterion of percentage errors in classification of unseen examples, showed the use of numerical values to be unsatisfactory. The mis-classification error of approximately 42 percent was too large to be ignored. The use of logical values resulted in a more acceptable error rate. It was necessary to select between the choice of 6 or 13 attributes. There was not much difference between their performances as far the secondary criteria were concerned but the use of F2 almost doubled the classification improvement. Therefore F2 was selected as the most promising configuration. Notice that though the performances using logical values based on the secondary criteria were not satisfactory these can (and were) improved, as is reported in the following Chapter. In conclusion, the tuning of the first search of the stopband was to be achieved by using the location of the peaks and their differences as attributes. Each numerical value was to be assigned one logical value out of four.

6.9 Evaluation of results and discussion (Search Two)

Percentage errors on classifying unseen examples

Table 28 shows the results for each configuration for both tests, expressed as the percentage error of mis-classification. From the table, the following can be established:

(i) All performances, except the ones with 4 logical values, improve as the size of the training set increases.

(ii) The minimum mis-classification error can be found at both configurations with 8 logical values.

(iii) The amount of classification improvement varies between configurations. Trees generated using 8 logical value attributes have slightly higher average performance than the ones produced using numerical attributes.

(iv) The initial error (i.e. column one) for the 8 logical configurations is considerably smaller than the error when using numerical configurations.

(v) Comparing with the counterpart results of search one (Table 22) the error is much higher. The main contributor to the error is due to the choice of the component. The direction was given right 50 per cent of the time which might have been achieved by pure chance. It seems that by increasing the number of classes generates worse results, hence the need for a larger training set.

Number of empty labelled leaves

For each of the six configurations three decision trees were generated by increasing the training set with the addition of an extra batch each time. The following comments can be made using Table 29.

Table 28 : Misclassification errors (Search II)

Configuration	Test-1	Test-2
F1	77.5	68.5
F2	60.6	68.3
F3	66.9	56.1
F4	80.3	65.9
F5	56.3	61.0
F6	70.4	56.1

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set

Table 29 : Number of empties per configuration (Search II)			
Configuration	Test-1	Test-2	Test-3
F1	0	0	0
F2	8	24	31
F3	26	71	137
F4	0	0	0
F5	9	24	29
F6	26	87	124

Table 30: Number of clashes per configuration (Search II)			
Configuration	Test-1	Test-2	Test-3
F1	0	0	0
F2	1	4	10
F3	0	2	9
F4	0	0	0
F5	1	4	10
F6	1	3	12

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set
‡ Test 3 denotes three learning sets of data

(i) The numerical configurations did not produce any empty situations.

(ii) All logical configurations generate a considerable number of empty situations which increase as more examples are entered in the training set. The largest amount is created with 8 logical value attributes.

(iii) Once again an increase in the number of classes generates more empty situations, notably for the 8 logical value configurations (compare with Table 23).

Number of clashes

Table 30 shows the results obtained . The following can be noticed:

(i) There are no clashes of attributes with the numerical configurations.

(ii) Irrespective of the number of attributes used 4 logical values produce the same number of clashes for each test. The number of clashes is comparable to when 8 logical configurations were used.

(iii) More clashes were generated (compare with Table 24) but the effect of increasing the number of classes is not as dramatic as when considering mis-classification errors or empty situations.

Number of nodes

Table 31 shows the results obtained testing the six configurations. The following observations can be made:

(i) Using numerical attributes the generated tress have a smaller number of nodes irrespectively of the number of attributes used.

(ii) Using logical attributes created larger trees especially when the number of attribute values increased.

Table 31 : Number of nodes per configuration (Search II)

Configuration	Test-1	Test-2	Test-3
F1	9	21	33
F2	25	69	89
F3	49	121	201
F4	11	23	37
F5	25	65	81
F6	49	137	193

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set
‡ Test 3 denotes three learning sets of data

Number of preconditions in rule base

Table 32 shows the results obtained. A list of remarks now follows:

(i) Unlike search one (see Table 26) this time numerical configurations generated more preconditions as the training set increased in comparison with 4 logical configurations.

(ii) Increasing the number of logical values an attribute can take resulted in a greater number of preconditions.

Number of rules

Table 33 shows the results obtained by converting each tree to a group of rules. The following can be noticed:

(i) The introduction of extra attributes is beneficial only when numerical values are used.

(ii) Numerical configurations produce less rules than those in which logical attributes are used. The rate of increase though was much greater as the number of examples increases. Notably configurations with 4 logical attribute values (F2, F5) and F6 configuration seem to stabilise.

6.10 Selection of configuration for search two

The numerical based configurations were not considered for the same reasons as discussed in Section 6.7. Again the selection was between either 4 or 8 logical values with 13 or 6 attribute values. The use of 8 logical values had the better mis-classification error. The employment of six attribute values resulted in a better performance as far as the secondary criteria were concerned. Hence F6 was selected as the most promising configuration.

Table 32 : Number of preconditions per configuration (Search II)			
Configuration	Test-1	Test-2	Test-3
F1	14	57	95
F2	24	85	89
F3	33	93	121
F4	20	64	126
F5	23	72	79
F6	30	86	94

Table 33 : Number of rules per configuration (Search II)			
Configuration	Test-1	Test-2	Test-3
F1	5	11	17
F2	2	24	25
F3	17	33	41
F4	6	12	19
F5	9	21	22
F6	16	30	32

‡ Test 1 denotes one learning set of data and two testing sets
‡ Test 2 denotes two learning sets of data and one testing set
‡ Test 3 denotes three learning sets of data

6.11 Discussion

The configuration choice for each search was made empirically, as shown above. At the same time the configurations chosen seemed to be sensibly right. For search one 13 attributes, each with 4 permissible logical values were selected. Search two had 6 attributes with 8 allowable logical values. Using search one, one tried to discover if further tuning was required so a strict testing was required. This is a *full-scale* approach. It involved not only the checking of a position of an attribute but also its relative position to other attributes - hence the need for the differences. Using search two, one tried to find the combination of component and direction for correcting the position of an individual attribute (i.e. one of the peaks) at a time. This can be described as a *reductionist* approach. The outcome influenced the position of one attribute and we were not worrying about the effect it will have, if any, on the rest of the attributes. Therefore, there is no need for differences to be included. The exact position of an attribute is therefore very important and 8 logical attribute values are needed to provide a fuller description of the position.

6.12 Conclusions

It was decided to use linguistic labels for the description of the position of each attribute of the magnitude response rather than the raw numerical values. The work undertaken showed that hereafter thirteen attributes each taking a linguistic label from a set of 4 to be employed for the first search. The second search to use six attributes each taking a linguistic label from a set of 8.

References

1. Tsaptsinos D., Mirzai A.R., Jervis B.W., and Cowan C.F.N., *Comparison of knowledge elicitation techniques in the domain of electronic filter tuning*, IEE Proceedings-F, Vol 137, No. 5, pp. 337-344, 1990.

2. Quinlan J.R., *Decision trees as probabilistic classifiers*, Proceedings of the Fourth International Workshop on Machine Learning, Morgan Kaufmann, pp. 31-37, 1987.

3. Zadeh L.A., *Fuzzy logic*, IEEE Computer, Vol. 21, No. 4, pp. 83-93, 1988.

4. Quinlan J.R., *Simplifying decision trees*, Knowledge-based systems, Vol. 1, pp. 241-254, 1988.

5. Quinlan J.R., *Decision trees and multi-valued attributes*, Machine Intelligence, Vol. 11, pp. 305-318, 1985.

6. Cheng J., Fayyad U.M., Irani K.B., and Qian Z., *Improved decision trees: a generalised version of ID3*, Proceedings of the Fifth International Conference on Machine Learning, Morgan Kaufmann, pp. 100-106, 1988.

7. Quinlan J.R., *Generating production rules from decision trees*, Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Vol. 1, pp. 304-307, 1987.

8. Corlett R.A., *Explaining induced decision trees*, Expert Systems Conference, pp. 136-142, 1983.

Chapter Seven

The Knowledge-base Construction

παθηματα μαθηματα

Greek proverb

7.1 Introduction

The generation of the decision tree and therefore of the rule set of each search is the subject of Chapter 7.

Techniques for decreasing the complexity of a rule set without reducing performance are also given. The manual inspection for the identification and elimination of rules which will never be active and the erasure of rule conditions after testing their relevance using contingency tables were two techniques found to be most effective.

The final section details the evaluation of the quality of the rules by considering domain knowledge which reinforced belief in the trustworthiness of the generated rules.

7.2 Induction of decision tree for the stopband region

Three visits to Newmarket Microsystems produced a total of 159 examples. These are the same examples employed for the comparison of decision trees, as reported in the previous chapter. Twenty one of these were examples where the user had realized that the wrong action had been taken. For that reason, they were not included in the induction process.

The ID3 algorithm was developed by Quinlan[1] for problems associated with the game of chess, in particular for endgame knowledge. In the chess domain, an entire database of examples was used. In our application it was not feasible to generate a complete set of examples. For example, it was not possible to ask the operator to place, say, the first positive peak in a 'far left' position and at the same time to have the second negative peak in a 'close right'

location. The generated tree had to, eventually, analyze unknown examples. At that stage of the research, there was no evidence to show that the collected examples were sufficient or that they constituted a small sample. Two ways to find out are for either the expert to investigate the rules or to test the tree against a new set of data. The first way involves the expert looking at the selection of relevant attributes and the relationships between them as presented by the rules. This was not possible in this study. The choice of relevant attributes from an initial set of attributes provided by the expert in the first place does not reveal much. Furthermore, the expert was not aware of what rules existed anyway, so the rules formed by ID3 were mentally uncheckable. Testing against a new sample would have given only an indication of the validity of the rules. Also, the indication would have been very dependent on the sample. The problem once again is that there were many routes towards the goal. It was then decided to test the system on-line and to record and observe its performance. The actual testing is reported in Chapter 9. An account of work performed prior to the testing in order to bypass certain ID3 problems and to optimise the execution efficiency of the rules now follows.

7.2.1 Modifying the rule set of search one

Removing the 'unsuccessful' examples left 138 examples. The distribution of the examples is shown in Table 34.

Prior to the execution of the algorithm a change to the training set was made. This involved a reduction of the set of attribute values assignable to the fourth positive peak (P4). In particular, all references to the label 'right' were renamed 'ok'. The reasons behind this were that P4 lies to the right side of

Table 34 Distribution of examples per class category	
Class	Number of examples
Carry-On	104
End-of-process	34

the ok-range only a few times (8 out of 138). Also, the objective of the expert operator was to place P4 to the far right of the display, irrespective of the exact position. The modified example set was fed to the algorithm and a new decision tree, incorporating the changes, was generated. It is worth mentioning that identical examples were not used. That way the possibility of elevating an attribute's significance was reduced.

The decision tree had 34 leaves classified as *empty* and two leaves with the *clash* class (the two terms have been defined in the previous chapter). Before continuing any further it was thought to be beneficial to investigate the nature of the clashes. This involved finding those examples that contributed to each clash.

Table 35 displays the two 'clash' rules. The first clash arose due to the difference in class of the following two examples.

P1	p2	p3	p4	r1	r2	class	example number
1.397269	1.399436	1.40472	-	62	-	end	65
1.397262	1.399425	1.404553	-	58	-	carry-on	95

Example 65 reports an *end-of-process* whereas example 95 a *carry-on*. It became apparent that example 65 was wrongly classified. The operator should not have ended the process at that stage. The value of the first negative peak (R2) was not identified. Ending a stopband tuning process when unknown values are present contradicts the existing knowledge. The expert had mentioned the need for the two negative peaks to be about the same level for the filter to be tuned. This is not possible when one of the values is missing. Further class revealed that the expert never terminated a stopband tuning if a value was missing. The class of example 65 was then changed to *carry-on*. Another matter to notice is that example 65 was not used when the ok-ranges

Table 35 Rules with a clash action		
Rule Number:	One	Two
	if d3 is absent	if d3 is ok
	and p1 is ok	and p3 is ok
	and p2 is ok	and r2 is ok
		and d1 is ok
		and r1 is ok
		and d4 is ok

were specified, because of the appearance of unknown values, so there was no need to re-calculate the ranges.

The second clash was of a different nature. Ten of the twelve contributing examples had an 'end' outcome. Because of the vast difference (10-2) the majority ruled so the classes of the two examples were changed to 'end'. The ranges were then re-calculated in order to take into inspection of the rest of the examples with an *end-of-process* account the two examples and the transformation of the numerical values to logical values took place again. The new decision tree generated 58 rules which determined the class an example took. The values were carry-on, end or empty. There were no clash rules anymore. Table 36 shows the distribution of rules per class. Work undertaken was concerned with the identification and, possible, removal of the empty rules. The thirty-three empty rules had in common that the attribute of the first *if-branch* was Diff3. Diff3's value is calculated by subtracting the first positive peak (P1) from the fourth positive peak (P4). In thirty-one rules out of 33 Diff3 had the value 'ok'. In order for Diff3 to take this value then <u>both</u> peaks (ie. P1 and P4) must be present. Bearing this in mind, each empty rule was examined.

Two empty rules were eliminated since they had an additional *if-branch* which stated that *P1 is absent*. This implies that Diff3 cannot be 'ok' while P1 is absent, ie. this rule will never apply. Similarly, two more empty rules were removed since *Diff1 is absent* appeared in the left hand side of the rule. Diff1 is calculated by subtracting P1 from the second positive peak (P2). In order for Diff1 to be absent then either P1 or P2 or both are absent. If both are absent or if P1 is absent then Diff3 can never be in the ok-range. If P1 is

Table 36 Distribution of rules per class after elimination of clash rules

Class	Number of rules	Percentage
Carry-on	13	22.41
Empty	33	56.90
End-of-process	12	20.69

present then P2 must be absent. But a further *if-branch* states that *P2 is right*. Since it is impossible for this situation to arise the empty rules were erased.

It was mentioned previously that one rule for determining the continuation of the tuning is that every attribute must be known. Using this rule 13 empty rules were initially changed to *carry-on* rules. Then it was recognised that they were redundant rules since they can be replaced with a set of rules which state that if any attribute is missing then class is *carry-on*. Furthermore, two more empty rules were eliminated. The reasons for their dismissal will be explained since these rules demonstrated a drawback of the ID3 algorithm. One of the rules stated:

if Diff3 is ok
and P3 is ok
and Diff1 is right
and R2 is ok
and Diff4 is absent
then class is empty

For Diff4 to be absent then either P2 or P3 or both are missing. But *P3 is ok* appears as an *if-branch*, so P2 might be missing. This argument is also invalid since *Diff1 is right* appears which implies that both P1 and P2 are present. Therefore, P2 cannot be missing, causing this rule to be unnecessary since all of the conditions can never occur. This is a demonstration of the ID3 problem known as **irregular branching**. The algorithm could not possibly determine that branching for *Diff4 is absent* is not actually possible. Table 37 presents the new distribution of rules per class.

Table 37 Distribution of rules per class after elimination of empty rules		
Class	Number of rules	Percentage
Carry-on	18	40.00
Empty	15	33.33
End-of-process	12	26.67

7.2.2 Simplifying the rule set of search one

Tree generation inevitably creates immense decision structures. While comparing the various decision trees (see Chapter 6) the number of pre-conditions was used as a criteria of what constitutes a well formed tree structure. Work in this section describes how the number of pre-conditions was reduced and efficiency was kept at the same level.

There exist two methods of creating an efficient and at the same time understandable decision tree. The first method is known as *windowing*[2]. The basic idea is to select a small subset of the examples (*the window*) rather than the complete training set. A tree is then generated and the remaining examples are tested using the tree. The incorrectly classified examples are added to the window and the process is repeated until there are no mis-classifications. This technique has been tested in a series of experiments[3] and it was found to have some problems of its own. In the filter domain the technique was not considered firstly due to the small collection of data and secondly due to the presence of noise in the data.

The second method is concerned with the *pruning* of decision trees. Quinlan[4] proposed and empirically compared four techniques. One technique, simplifying the production rules, was proved by Quinlan to be especially powerful since it matched or outperformed the rest of the techniques on nine out of twelve tests. Therefore this technique was implemented for the tree of the first search of the stopband. First, the decision tree was *compiled* into production rules. The extraction of production rules was achieved by following a path through the tree to one of the leaves.

The technique will be better explained using an example. As an illustration

of the process the rule in Table 38 will be considered.

Step 1 For every condition branch a 2 x 2 contingency table is created. Table 39 shows the contingency table created for the first condition branch, i.e, Table 38, condition 1. The numbers in the cells were obtained from the training set. The number in cell *a* represents the number of examples (of the training set) that satisfy the condition (i.e. the entire left hand side) and belong to the *carry-on* class (i.e. the one given by the rule). The number in cell *b* represents the number of examples (of the training set) that satisfy the condition (i.e. the entire left hand side) and belong to any other class other than the one given by the rule. In this case this means the *end-of-process* class. The number in cell *c* represents the number of examples (of the training set) that belong to the class given by the rule (i.e. *carry-on*) without satisfying the condition (i.e. it is irrelevant what value d_3 takes). The number in cell *d* represents the number of examples (of the training set) that do not belong to the class given by the rule (i.e. *carry-on*) and do not satisfy the condition (i.e. it is irrelevant what value d_3 takes).

Step 2 Having created the table the necessity of the presence of the condition is examined. In other words the effect of its removal on the accuracy of the rule is observed. The accuracy, with the condition present is estimated[4] as

$$C_1 = \frac{a-0.5}{a+b}$$

where C_1 represents the probability of needing the condition, whereas the accuracy, without the condition present is estimated[4] as

$$C_2 = \frac{a+c-0.5}{a+b+c+d}$$

where C_2 represents the probability that the condition arose by chance.

Table 38 A sample rule to demonstrate post pruning	
Condition 1	if d3 is ok
Condition 2	and p3 is ok
Condition 3	and d1 is ok
Condition 4	and r2 is right
Condition 5	and p2 is right
Condition 6	and d7 is right
Action	then carry-on

Table 39 Contingency table for the first condition of rule of table 38		
	Carry-on class	End-of-process class
Keep first condition	**a** 2	**b** 0
eliminate first condition	**c** 4	**d** 0

If $C_2 \geq C_1$ then the condition is dropped, otherwise it is kept. For the above example, C_1=0.75 and C_2=0.917, therefore the condition is dropped.

Steps 1 and 2 are repeated for all conditions of the original rule (i.e. no conditions are dropped at this stage). For our example the first three conditions and the fifth condition were found to be non-contributors, hence they were eliminated. The process is then carried out for the remaining conditions until the stage where no condition can be dropped is reached. For the rule example only the last condition was retained (i.e. d_7 *is right*).

Step 3 A certainty factor in a percentage form, given by the calculation of C_1 times 100, is assigned to the simplified rule. For our example the simplified rule has a certainty factor of 86.5 per cent.

Unfortunately the elimination of conditions and/or the calculation of the certainty factors cannot be done using the expert system shell. This is a facility worth having in order to save time on tedious tasks. The allocation of a certainty factor to a rule was also unavailable with the Xi-Plus package. This again would had been of value when considering *conflict resolution* (i.e. resolving the problem of which rule to choose when more than one rule applies). The only solution was to include the certainty factor *by hand* in the right hand side of the rule and the user to solve the conflict.

7.2.3 Evaluation of reduced rule set of search one

The examination of the significance of conditions resulted in the discarding of rules. This was true when the entire left hand side of certain original rules were found to be irrelevant. Table 40 presents the new distribution of rules

Table 40 Distribution of rules per class after post pruning		
Class	Number of rules	Percentage
Carry-on	15	39.47
Empty	15	39.47
End-of-process	8	20.06

per class. The next stage was to determine how well the cut down rule set functioned. This involved the evaluation of how the new set of rules performed on the training examples.

Table 41 presents the successful results obtained. Notice the low success performance of the reduced set. This was especially true for the examples with *end-of-process* class. This though can be misleading as will be shown. Table 42 presents the unsuccessful results obtained. Unsuccessful being either when the wrong class or a clash is given. Clash situations arise due to:

(i) more than one rule applies but all rules have the same class, or

(ii) more than one rule applies but the rules have different class.

The wrong outcome state arises due to:

(iii) X outcome is expected and something else is generated.

Breaking down the *end-of-process* unsuccessful results it was found that all 34 were due to clashes. Twenty-two of them because of category (i) and 12 because of category (ii). Since all 22 clashes were generated by rules with *end-of-process* as their class can be allocated to the success region. Examination of the *carry-on* unsuccessful results showed that 32 were due to clashes and 2 due to category (iii). Unfortunately no action can be taken to correct category three errors. Concentrating on category (ii) clashes for both classes a heuristic rule was used in order to eliminate some of them. The heuristic rule adopted states: choose the rule with the higher certainty factor. This way 4 *end-of-process* and 30 *carry-on* category (ii) clashes were resolved and allocated to the success region. Tables 43 and 44 present the new right and wrong classification figures respectively. The tables show that the elimination of some rules and the improvement of the comprehensibility of the rest with

Table 41 Testing reduced set of rules with training set	
Number of examples in training set	138
Successful classification	72
Number of examples in training set with end-of-process classification	36
Successful end-of-process classification	2
Number of examples in training set with carry-on classification	102
Successful carry-on classification	70

Table 42 Testing reduced set of rules with training set	
Number of examples in training set	138
Unsuccessful classification	66
Number of examples in training set with end-of-process classification	36
Unsuccessful end-of-process classification	34
Number of examples in training set with carry-on classification	102
Unsuccessful carry-on classification	32

Table 43 Testing reduced set of rules with training set	
Number of examples in training set	138
Successful classification	128
Number of examples in training set with end-of-process classification	36
Successful end-of-process classification	28
Number of examples in training set with carry-on classification	102
Successful carry-on classification	100

Table 44 Testing reduced set of rules with training set	
Number of examples in training set	138
Unsuccessful classification	10
Number of examples in training set with end-of-process classification	36
Unsuccessful end-of-process classification	8
Number of examples in training set with carry-on classification	102
Unsuccessful carry-on classification	2

Table 45 Distribution of examples per class category (search 2)	
Class	Number of examples
c4a	32
c4c	29
c7a	22
c7c	21

subsequent faster execution resulted in an insignificant percentage drop of classification accuracy in comparison with the classification accuracy prior to pruning.

7.2.4 Modifying the rule base of search two

Removing the 'unsuccessful' and the *end-of-process* examples left 102 examples. The distribution of the examples is shown in Table 45. It was not possible to judge the rules about irregular branching etc. so the number of clash and/or empty rules could not be reduced.

7.2.5 Simplifying the rule set of search two

Work, similar to the one described in Section 7.2.2, was undertaken resulting in twelve rules being removed altogether. Table 46 shows the new distribution of the rules.

7.2.6 Evaluation of reduced rule set of search two

No evaluation was performed due to time constraints.

7.3 Induction of decision tree for the passband region

Four more visits to Newmarket Microsystems produced a total of 196 examples for the tuning of the passband region. The nine attributes, different to ones used for the stopband, employed for the tuning of the passband are the following:

(i) Ripple (see explanation below in text)

(ii) Low Passband (passband value at 1.4005 MHz)

Table 46 Distribution of rules per class (search 2)

Class	Number of rules
c4a	16
c4c	7
c7a	10
c7c	7
empty	101
clash	11

(iii) High Passband (passband value at 1.4025 MHz)

(iv) Insertion Loss (see explanation below in text)

(v) Carrier rejection (attenuation at reference frequency)

(vi) Low Stopband (stopband value at 1.3993 MHz)

(vii) High Stopband (stopband value at 1.405 MHz)

(viii) Low Stopband Return (see explanation below in text)

(ix) High Stopband Return (see explanation below in text)

The selected attributes are outlined in Figure 16. Low refers to the left side of the response. High refers to the right side of the response. A program was written, in HP-Basic, with the help of the expert in order to automate the extraction of the attributes. The program samples the response, at appropriate points, in order to find the values for the attributes. Fifty-one sample points are used for attributes (viii) and (ix) and twenty points for the ripple. For example, the sampling for the calculation of the ripple starts at 1.4008 MHz and ends at 1.402 MHz. The maximum and the minimum sample are found and their difference is the ripple. The minimum sample of the ripple is the insertion loss. The low and high stopband returns are calculated similar to the insertion loss but the sample ranges are 1.38 MHz to 1.398 MHz and 1.406 MHz to 1.42 MHz respectively. Prior to the generation of examples for the passband tuning two assumptions had to be met:

(α) The stopband had already been tuned, and

(β) the three components (T1, T2, T3) were, almost, screwed in.

Strictly speaking, this part of the system does not deal with the passband only. It incorporates further tuning of the stopband, if the need arises. Therefore, the set of possible components (i.e. classes) comprise of all the

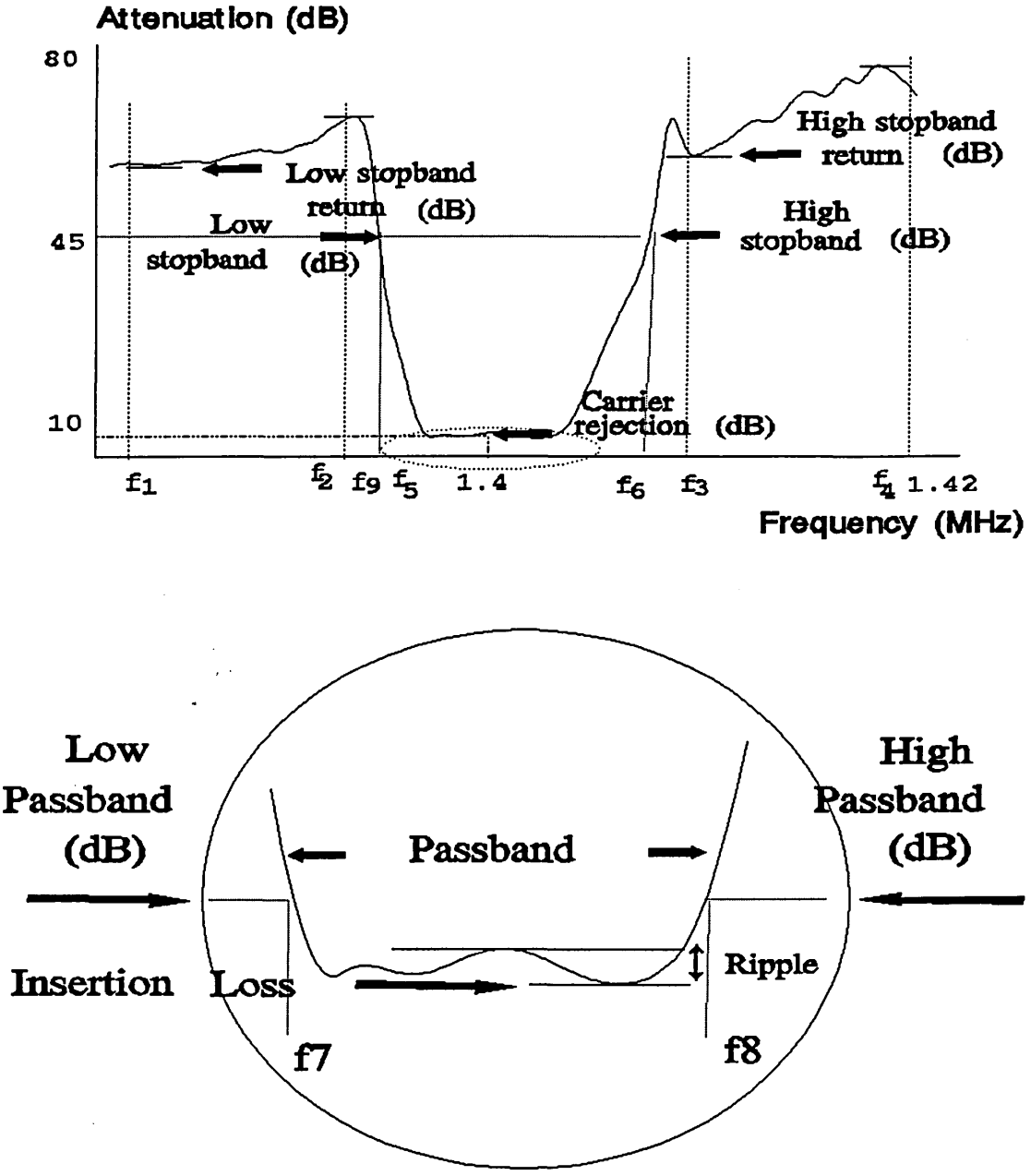


Figure 16: Magnitude response showing the attributes used for the tuning of the passband. The Low Stopband Return is the minimum attenuation (dB) measured between the $f_1=1.38$ MHz to $f_2=1.398$ MHz range. The High Stopband Return is the minimum attenuation (dB) measured between the $f_3=1.406$ MHz to $f_4=1.42$ MHz range. The Low Stopband attenuation (dB) is measured at $f_9=1.3993$ MHz. The High Stopband attenuation (dB) is measured at $f_6=1.405$ MHz. The Carrier Rejection attenuation (dB) is measured at $f_5=1.4$ MHz. The Ripple is the difference between the maximum and minimum attenuations measured in dB's between 1.4008 MHz and 1.402 MHz. The minumum attenuation (dB) measured in this range is the Insertion Loss. The Low Passband is the attenuation (dB) measured at $f_7=1.4005$ MHz. The High Passband is the attenuation (dB) measured at $f_8=1.4025$ MHz.

tunable components.

7.3.1 Partition of search spaces

The search spaces were partitioned as for the stopband region. This meant that search one specifies the conclusion or not of the tuning process, search two the appropriate component and direction combination and search three the distance to turn.

7.3.2 Search one rule set

Unlike the stopband, this time the expert had knowledge of the range of values in which each attribute must lie to be considered tuned (i.e. the *ok-range*). The specification, after a couple of revisions, is given in Table 47. Therefore a single rule stating that if every attribute is within the *ok-range* then the process can stop was sufficient. Obviously, if any of the attributes did not conform to this then the tuning process was to be continued. With Xi-Plus this was represented as a default rather than a rule. The rule was augmented after some initial testing by including one more condition in the left hand side: the low stopband return and high stopband return values should also be close to each other (i.e. their difference is less than 3 db). An analysis of the distribution of values, for each attribute, for all the collected examples confirmed the suitability of the specification. Table 48 gives the distribution values of the attributes collected during the acquisition of the training examples. It can be seen, for example, that the ripple is within the *ok-range* for all examples with *end-of-process* class. The same applies for the rest of the attributes except for the two *return* attributes. These attributes had one or two values not within the *ok-range* but still the expert terminated

Table 47 Specification for attributes used in the passband	
Attributes	Acceptable Values (db)
ripple	0 - 1
insertion loss	0.5 - 5.0
carrier rejection	> 10
low stopband	> 45
high stopband	> 45
low passband	< 4
high passband	< 4
low stopband return	> 45
high stopband return	> 45

the tuning process. Furthermore, the expert changed the two *return* attributes' value from 45 db to 48 db.

7.3.3 Search two rule set

The withdrawal of the *end-of-process* examples from the training set left 152 examples. The second search for the passband utilised eight logical values, in a similar fashion to the respective search of the stopband. The logical values were close-left, close-right, far-left, far-right, ok, left, right, unknown. The generation of logical values was as before (See Section 6.3). The standard deviation of all the *carry-on* examples was computed and the ranges were determined by adding the standard deviation to the limits of the *ok-range*. Some attributes were not assigned all the logical values. For example, the ripple does not take any value less than zero, so references to left, close-left, far-left are not necessary. Additionally, the *ok-range* was split into three ranges (close-ok, far-ok, middle-ok) for some of the attributes (carrier rejection, high stopband, low stopband, low stopband return, high stopband return). The reason being that the expert continued even if the attributes were within the *ok-range* (See Table 48). The examples were fed to the Xi-Rule package and 63 rules were generated (without considering the empty and the clash rules). The distribution of the rules is given in Table 49.

7.3.4 Modifying rule set of search two

No work was performed due to time constraints.

7.3.5 Simplifying rule set of search two

No work was performed due to time constraints.

Table 48 Distribution of attribute values of training set

Attribute	Range	End-of-process	Carry-on
ripple	0 -1	38	84
	> 1	0	74
insertion loss	0.5- 5.0	38	158
	> 5	0	0
low passband	0 -4	38	153
	> 4	0	5
high passband	0 -4	38	113
	> 4	0	45
carrier rejection	> 10	38	158
	0 -10	0	0
low stopband	> 45	38	145
	0 -45	0	13
high stopband	> 45	37	114
	0 -45	1	44
low stopband return	> 45	36	107
	0 - 45	2	51

Table 48 continued Distribution of attribute values of training set			
Attribute	Range	End-of-process	Carry-on
high stopband return	> 45	38	137
	0 - 45	0	21

the tuning process. Furthermore, the expert changed the two *return* attributes' value from 45 db to 48 db.

7.3.3 Search two rule set

The withdrawal of the *end-of-process* examples from the training set left 152 examples. The second search for the passband utilised eight logical values, in a similar fashion to the respective search of the stopband. The logical values were close-left, close-right, far-left, far-right, ok, left, right, unknown. The generation of logical values was as before (See Section 6.3). The standard deviation of all the *carry-on* examples was computed and the ranges were determined by adding the standard deviation to the limits of the *ok-range*. Some attributes were not assigned all the logical values. For example, the ripple does not take any value less than zero, so references to left, close-left, far-left are not necessary. Additionally, the *ok-range* was split into three ranges (close-ok, far-ok, middle-ok) for some of the attributes (carrier rejection, high stopband, low stopband, low stopband return, high stopband return). The reason being that the expert continued even if the attributes were within the *ok-range* (See Table 48). The examples were fed to the Xi-Rule package and 63 rules were generated (without considering the empty and the clash rules). The distribution of the rules is given in Table 49.

7.3.4 Modifying rule set of search two

No work was performed due to time constraints.

7.3.5 Simplifying rule set of search two

No work was performed due to time constraints.

Table 49 Distribution of rules per class (passband)	
Class	Number of rules
$c4_a$	10
$c4_c$	4
$c7_a$	8
$c7_c$	4
$t1_a$	3
$t1_c$	0
$t2_a$	12
$t2_c$	5
$t3_a$	12
$t3_c$	5
empty	117
clash	18

‡ The character 'c' as in $C4_c$ donates the clockwise direction
‡ The character 'a' as in $C4_a$ donates the anti-clockwise direction

7.3.6 Evaluation of the quality of the rules

Only seven attributes appear on the decision tree. The absent attributes are: carrier rejection and insertion loss. This was welcomed as the reasons given beneath will demonstrate.

Somewhere hidden in one of the transcriptions of the protocol analysis the following statement appears: *if short or long way out (referring to insertion loss) reject the filter*. This statement indicates that the insertion loss must be within the *ok-range* for the filter to be accepted but it does not influence the choice of the tunable component to be used. The analysis of the values' distribution, shown before in Table 48, shows that the values of the carrier rejection and for the insertion loss always lie within the *ok-range* irrespective of whether it is an *end-of-process* or a *carry-on* example. This is further evidence that these two attributes do not contribute to the selection of the tunable component. Therefore, the algorithm did well to recognise the irrelevance of these two attributes.

Sixty three rules (excluding empty and clash rules) were generated. The attribute *ripple* appeared as the root of the tree. This attribute took the following attribute values: ok (34), close-right (15), far-right (7), right (7). The number in the bracket indicates in how many rules the attribute ripple with the applicable value appeared in the left hand side. Now, if the rules with *ripple is ok* are considered then one would expect the class to be given as either C_4 or C_7, except if the low or high passband is wrong (where component should be given as T_1, T_2, or T_3). The reason for that lies in the recognition of the contents of the transcripts where the expert said that: *if something is wrong with the ripple or the low or high passband then use T-components, for*

the rest C-components. Twenty seven out of thirty two rules gave C-components. Seven rules can be judged as wrong due to the above observation, but again maybe they hold some special cases. Considering rules with the ripple being far-right, right or close-right one again expects T-components. This is exactly what happened except for two cases when ripple had the value close-right. The induction avoided the generation of *non-logical* rules such as: *if the ripple is ok and the low passband is ok then use* T_1.

The above remarks reinforced belief in the trustworthiness of the generated rules.

7.4 Conclusions

The work undertaken in order to reduce the complexity of the rules generated by ID3 showed that the ID3 algorithm produces irrelevant rules which had to be identified and eliminates manually. Additionaly the use of contingency tables proved to be effective when eliminating branches of the decision rules. The performance of the reduced rule set was not affected.

References

1. Quinlan J.R., *Discovering rules by induction from large collections of examples*, In: Expert systems in the micro-electronic age (Ed. D. Michie), Edinburgh University Press, 1979.

2. Quinlan J.R., *Induction of decision trees*, Machine Learning, Vol. 1, pp. 81-106, 1986.

3. Wirth J., and Catlett J., *Experiments on the costs and benefits of windowing in ID3*, Proceedings of the Fifth International Conference on Machine Learning, Morgan Kaufmann, pp. 87-99, 1988.

4. Quinlan J.R., *Simplifying decision trees*, Knowledge-Based systems, Vol. 1, pp. 241-254, 1988.

Chapter Eight

Neural Networks for Search 3 of the Stopband

8.1 Introduction

Neural networks offer an alternative approach for constructing learning systems and in Chapter 8 a detailed chronicle of the development of a multi-layer perceptron for the third search (i.e. how far to turn) of the filter tuning task is presented.

The performance of a neural network depends on a number of parameters such as the network architecture (i.e. number of hidden nodes), the number and presentation of the training examples, when to stop the learning process etc. These are usually determined through repeated experiments. The network architecture (57-11-10-1) was empirically determined and it was found necessary to build four networks (one for each component/direction combination) after some data analysis. Each network was trained for 75000 runs and a record was maintained of those weights that yield the minimum error as encountered during the learning. While the selected sets of weights did not manage to perform 100% on the training set. Section 8.7.2 shows that the errors were acceptable. In addition it has been cited that over-learning results to bad generalisation.

The lesson learned was that performing data analysis results in more selective training data hence better performance.

8.2 Reasons for implementing neural networks for search three

Neural networks are generally implemented for two reasons: understanding of the human brain and for achieving goals in computing. The primary incentive for implementing neural networks for the third search of the

stopband was the computational one. Testing the expert system (see next chapter) in a *live* environment indicated that it will be beneficial to have some sort of hint of how far to turn. It was obvious, during testing, that even when the correct outcomes are given for the first two searches if the operator turned arbitrarily, the tuning will take some time to terminate. Timing consideration was the main motivation. Additionally, initial accomplishments using neural networks for the third search were poor (See Chapter 5). Another motivation arose due to this. The work undertaken, described in this chapter, tried to establish if neural networks are generally inapplicable to the problem area or whether the neural networks were *incorrectly* applied previously. The latter is not difficult to do since a change to a parameter (e.g. number of input units) effects the network performance. The challenge was then to see if there were any network architectures that were more suitable. The ultimate goal was to produce a connectionist expert system[1], meaning an expert system interfaced to a neural network with the latter being another knowledge source. This was seen as increasingly important in the development of a total solution to the tuning problem. Recent research has begun to indicate the merits of such a union[2,3].

8.3 Collection of the training data

It was apparent by now that the expert did not have any knowledge of how far to turn. It was a matter of trial and error. Therefore, the training data consisted of examples generated using the *de-tune* procedure. The examples were created by having C_7 either at its optimum position (i.e. where it was placed when the expert finished the tuning) or mal-adjusted in steps of half a revolution up to 1.75 revolutions in a clockwise direction or up to 2.5

revolutions in an anti-clockwise direction. At each position of C_7 the other component (C_4) was mal-adjusted in steps of half a revolution from its optimum position up to 1.5 revolutions in a clockwise direction or up to 2.5 revolutions in an anticlockwise direction). In total 358 examples were generated for each filter. Each example comprised fifty seven point samples from the frequency range of 1.38 MHZ to 1.42 MHz, plus the class to which it belonged (i.e. the distance turned). Nineteen points were sampled from the left hand side of the stopband, nineteen from the passband region and the rest from the right side of the stopband (see Figure 17). The sample points were equally spaced for each part of the response.

8.4 Software implementation

Software from an available package (NeuralWare Explorer) was used to simulate the learning algorithm on an 80386 based microcomputer.

8.5 Development of the network architecture

The software package liberates the builder from the task of writing programs but still a group of crucial decisions concerning the architecture of the network have to be made. What is the optimal number of processing units in each layer? How many layers are to be built? How long should the training last? etc are a few of these decisions. Unfortunately, current literature and research does not provide a general method to the design issue for a given problem area. The following sections describe how the network for the third search of the stopband was constructed.

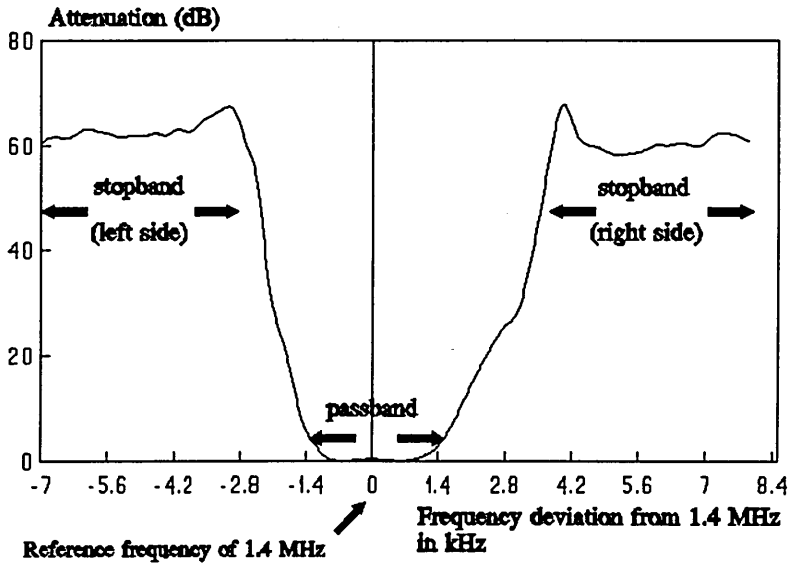


Figure 17: Tuned magnitude response

8.5.1 Network architecture

The back-propagation architecture described in Section 4.4 was implemented. This is a feedforward architecture consisting of a number of processing units with each unit belonging to one of the four available layers. Three layer neural networks (i.e. the input layer is not counted) were constructed.

8.5.2 Network type

The back-propagation architecture can be used in two modes:

(α) The auto-associative mode where the network is expected to generate an output which is the same as the input.

(β) The hetero-associative mode where the desired output is different to the input.

The last mode was used since the task of filter tuning falls into the sphere of classification.

8.5.3 Input layer configuration

The number of processing units in the input layer was set to 57, thus each sampled point was assigned to one and only one unit. In the literature it has been reported that numerical problems may be avoided by scaling the inputs to the [0,1] range. Hence, the input of each input unit was subjected to a simple transformation using the software package of the following form

Transformed input= Input Value * Scale Factor + Offset

where the values of 0.01 and 0.1 were used for the scale factor and offset respectively. The offset was used to avoid having any zero inputs. The new value of each input unit was then transferred to the processing units of the first hidden layer.

8.5.4 Output layer configuration

The number of processing units in the output layer was set to 1. The output of the single unit is simply the summation of all its inputs, multiplied by their associated weights, from the second hidden layer. Hence, the linear transfer function was applied. The desired output was represented in the learning set using a value from the [0,1] set. Table 50 displays the real desired output and the equivalent coded representation. The result obtained was limited to both an upper (1.0) and lower (0.0) bound and then compared to the desired output. Using the software package learning was inhibited when the error was lower than a pre-set value. This was accomplished with the use of a coefficient (C_3). Whenever the absolute difference between the desired and obtained output was less than the value of C_3 the error was set to zero. In the use mode (i.e. after learning) the output result was scaled using the linear transformation mentioned above in order to transform the data into more understandable units.

8.5.5 Hidden layers configuration

The sigmoid function (see Chapter 4) was used as the transfer function. The selection of the number of processing units for each hidden layer was not as natural and effortless as for the other layers. Their numbers were determined empirically (11 and 10 for the first and second hidden layer respectively) and no claim is made that they are the most appropriate. In the literature a number of approaches have been described. For example enter a large number of units and then freeze units to see the effect it has on the performance. This is known as *skeletonisation*[4].

Table 50: Presentation of classes to the neural networks	
Real Value	Coded Value
0.00	0.0
0.25	0.1
0.50	0.2
0.75	0.3
1.00	0.4
1.25	0.5
1.50	0.6
1.75	0.7
2.00	0.8
2.25	0.9
2.50	1.0

Alternatively, train the net with P_1 units until some optimum learning has been achieved. Add P_2 units to the net, re-train and continue in this way until some termination test has been satisfied[5].

In preliminary work the *optimum* number of hidden units was found by adding new hidden units to each layer and observing the effect. The next section describes this preliminary work undertaken in order to investigate the feasibility of neural networks and the architecture to be used for the tuning of electronic filters and search 3 in particular.

8.6 Determining the size of the training set

One has to be cautious when examples are employed for automatic learning. Prior to the neural network implementation various questions arose concerning the size of the learning set. For example, should the learning set include examples generated from different filters, should it include examples covering *de-tuning* of both components etc. Before implementing the network in full, some experiments were carried out with a smaller example set in order to get a feeling for the process and to assess the feasibility of the networks.

The training set included eleven examples from one filter generated with the C_4 component mal-adjusted in an anti-clockwise direction. Each example contained the 57 points sampled from the response. After running the network (57-11-10-1) a number of times (≈ 100,000) the performance was flawless viz. the system had learned to discriminate between those eleven classes (Table 51). Then the generalisation capabilities of the network were evaluated. This involved testing the network using unseen examples from other filters (Table 52). These examples were once again generated under the

Table 51: Network training with (100000 runs)	
Desired Values	Generated Values
0.0	0.03
0.1	0.1
0.2	0.2
0.3	0.3
0.4	0.4
0.5	0.51
0.6	0.62
0.7	0.73
0.8	0.83
0.9	0.91
1.0	0.96

Table 52: Network testing				
Desired Values	Generated values for different filters			
0.0	0.06	0.06	0.08	0.03
0.1	0.03	0.008	0.01	0.01
0.2	0.13	0.1	0.11	0.09
0.3	0.23	0.18	0.2	0.16
0.4	0.3	0.24	0.27	0.22
0.5	0.36	0.28	0.34	0.27
0.6	0.46	0.33	0.39	0.32
0.7	0.56	0.37	0.45	0.36
0.8	0.62	0.41	0.52	0.41
0.9	0.82	0.43	0.54	0.44
1.0	0.68	0.45	0.6	0.46

same *de-tune* procedure conditions. The results ranged from very good to useless. Similar results were obtained with different network architectures. For instance, when only the 19 sampled points from the left side of the stopband were used (19-11-10-1). Table 53 shows the learning results after 100,000 trials and Table 54 when the network was tested with unseen examples. The enigma of this situation (i.e. bad performance during testing versus good performance during learning) was resolved by simply plotting the responses. The initial positions of each response were all correct but not the same. This resulted in overlapping of classes. Figure 18 displays the left hand side of the stopband of six filter responses all classed as tuned. For example, the waveform generated with a 0.25 mal-adjustment was above the tuned waveform for some filters whereas other filters generated a waveform which could be found below the tuned response. The variety of the position of the responses which can be considered as tuned created an overlapping of classes. For that reason, it was decided to employ the *de-tune* data of just one filter. This would force the tuning of other filters towards one model *solution*. Additionally, in some cases, maladjustment by more than 2 revolutions caused negligible changes in the response (Figure 19). Those examples remained in the learning set but they were assigned the class of the earlier example. An overlapping of classes similarly occurred when the complete learning set was used (i.e all component and direction combinations). Responses generated using the left component (C_4) with, say, 0.5 turns resembled the ones generated using the right component (C_7) with 1.25 turns. For that reason it was thought appropriate to break the learning set into four sub-sets (Table 55).

Table 53: Network training with 19-11-10-1 nodes

Desired Values	Generated Values
0.0	0.02
0.1	0.08
0.2	0.17
0.3	0.27
0.4	0.37
0.5	0.48
0.6	0.60
0.7	0.72
0.8	0.82
0.9	0.90
1.0	0.96

Table 54: Network testing				
Desired Values	Generated values using different filters			
0.0	0.03	0.03	0.04	0.01
0.1	0.01	0.0	0.0	0.0
0.2	0.09	0.07	0.07	0.06
0.3	0.16	0.14	0.14	0.12
0.4	0.22	0.20	0.19	0.17
0.5	0.27	0.28	0.25	0.21
0.6	0.32	0.29	0.28	0.26
0.7	0.37	0.33	0.32	0.30
0.8	0.41	0.37	0.35	0.34
0.9	0.44	0.39	0.37	0.36
1.0	0.45	0.45	0.39	0.38

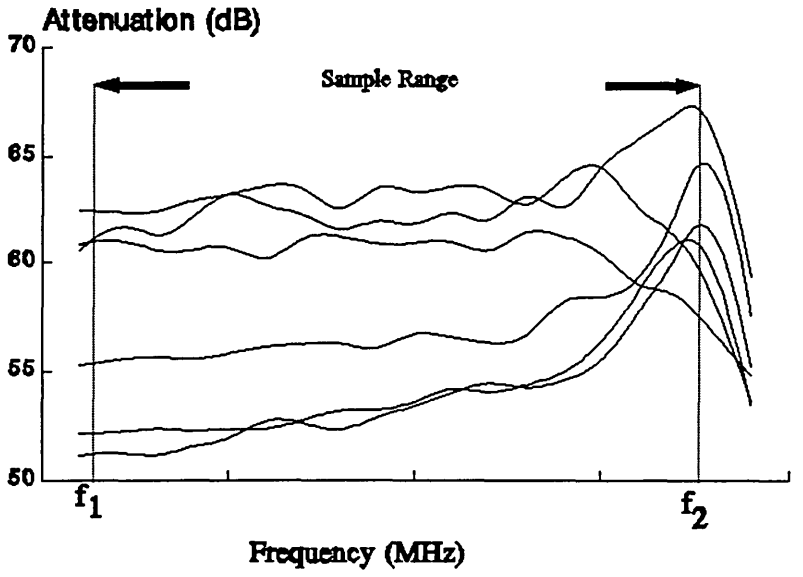


Figure 18: Six stopband responses (left side only). Nineteen equally spaced values are sampled from the left side of the magnitude response between $f_1 = 1.38$ MHz and $f_2 = 1.4$ MHz, in steps of 0.001 MHz.

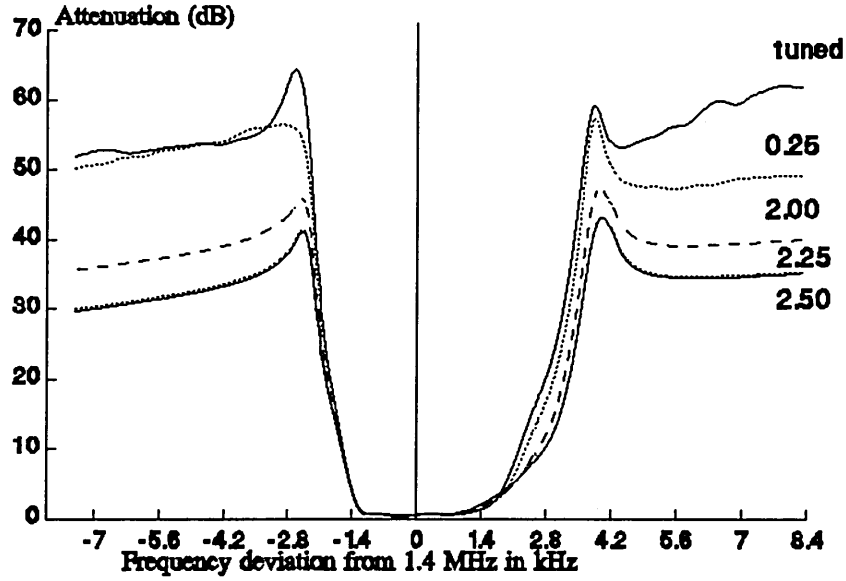


Figure 19: Magnitude responses generated by mal-adjusting C_4. The reference frequency is denoted by zero at the frequency-axis.

Table 55: Learning sets				
Learning...	C_4a	C_4c	C_7a	C_7c
No. of examples	215	144	216	178

8.7 Neural networks in learning mode

8.7.1 Stopping learning criterion

For each learning set the network was executed for 75000 runs. Every 1000 runs the learning was momentarily paused and the total sum of squared errors was calculated. The total squared error being the sum of all the squared differences between the desired class and the generated class. The network with those weights which generated the smallest squared error was selected. This way it is suggested that the better a system performs on the learning set, the better it will perform on tests.

8.7.2 Measuring the performance of the networks

Figures 20 to 23 display the learning curve for the four networks. These graphs show the plots of the total squared error against iteration number. The minimum total squared error (3.3) when learning C_4 turned anti-clockwise was found in run 56000 (Figure 20). The minimum total squared error (0.55) when learning C_4 rotated clockwise was found in run 66000 (Figure 21). The minimum total squared error (1.25) when learning C_7 turned clockwise was located in trial 55000 (Figure 22). Finally, the minimum total squared error (4.62) when learning C_7 rotated anti-clockwise was located in trial 62000 (Figure 23).

Various indicators can be used to measure the performance of a network. One such measure is the fraction of patterns classified correctly. The network for learning C_4 anti-clockwise is investigated below. Since any generated outcome

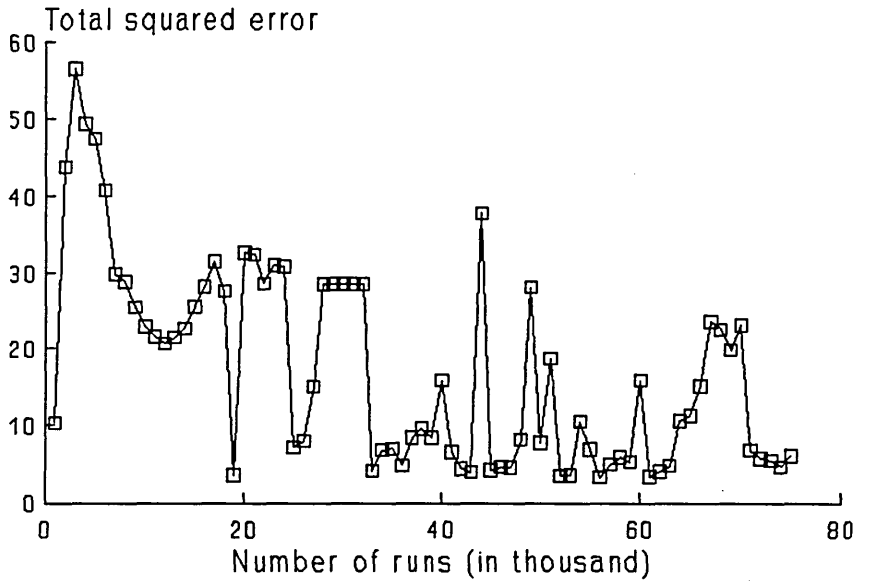


Figure 20: C_4 anti-clockwise learning curve

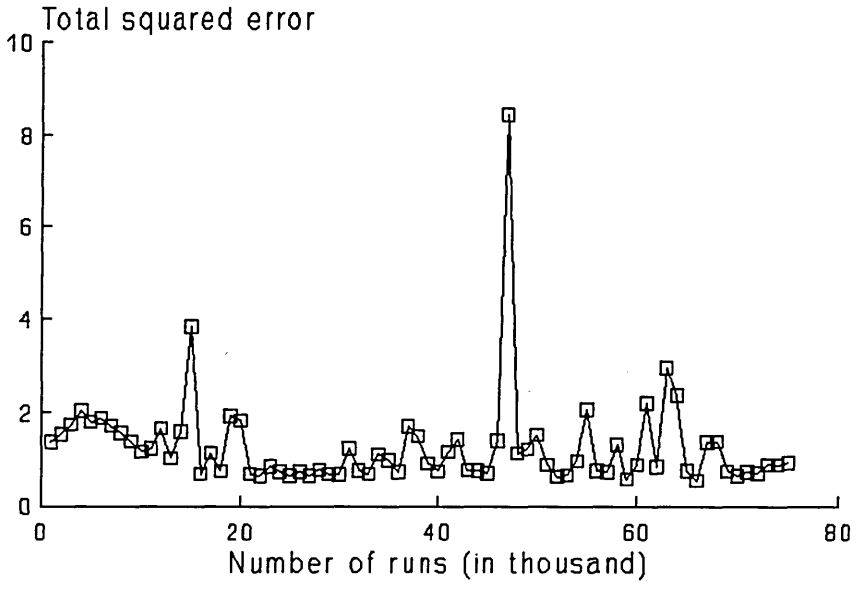


Figure 21: C_4 clockwise learning curve

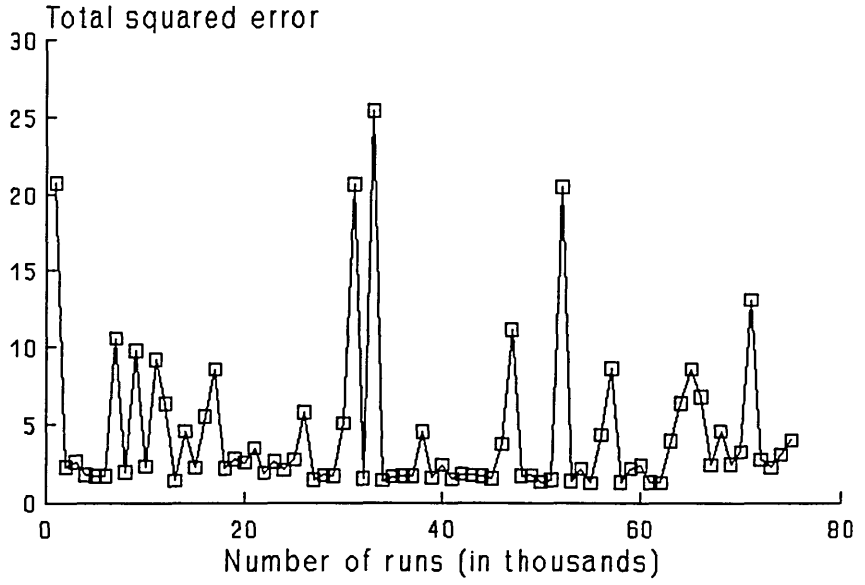


Figure 22: C_7 clockwise learning curve

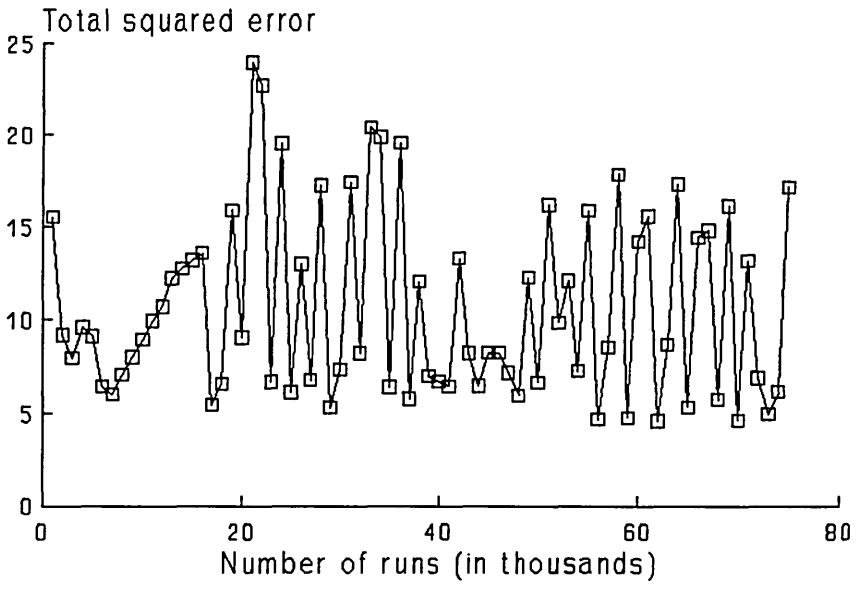


Figure 23: C_7 anti-clockwise learning curve

within ± 0.04 units from the desired outcome is considered to be correct (for example any value between 0.36 and 0.44 is considered right if the desired output is 0.4) the optimum minimum total squared error is then 0.3456 (i.e 215 examples times 0.04^2). The minimum total squared error reported above was 3.3. This is almost 10 times higher than the optimum. A discussion breaking down the results now follows.

In total 46.8 percent of the examples were assigned the correct number of turns. Individual outcomes will be examined below. Looking at all the examples with a desired class value of zero (Figure 24) it is obvious that except in one case the present set of weights generates the correct results. Figures 25(a) to 25(i) show similar plots for classes 0.1 to 0.9 respectively. A number of predicted values are not very close to the desired values. Those that predict higher are clearly wrong but a wrong action can be corrected later, just as in the performance of the human user. A prediction of smaller values will force a longer tuning. What was important was for the network to discriminate between extremely dissimilar values like 0 and 0.9 rather than neighbouring values like 0.8 and 0.9. In each desired class there were examples which stood out as being more difficult to learn, hence increasing the error.

At this stage the network was examined against examples not previously seen. One testing involved *de-tune* examples obtained by mal-adjusting C_4 in a clockwise direction. The expected outcome was 0 (i.e. because C_4 does not need maladjustment in an anti-clockwise direction) and Figure 26 illustrates the output of the network. Figure 26 shows that in the majority of the cases the predictions of the neural network were within the correct area. Some

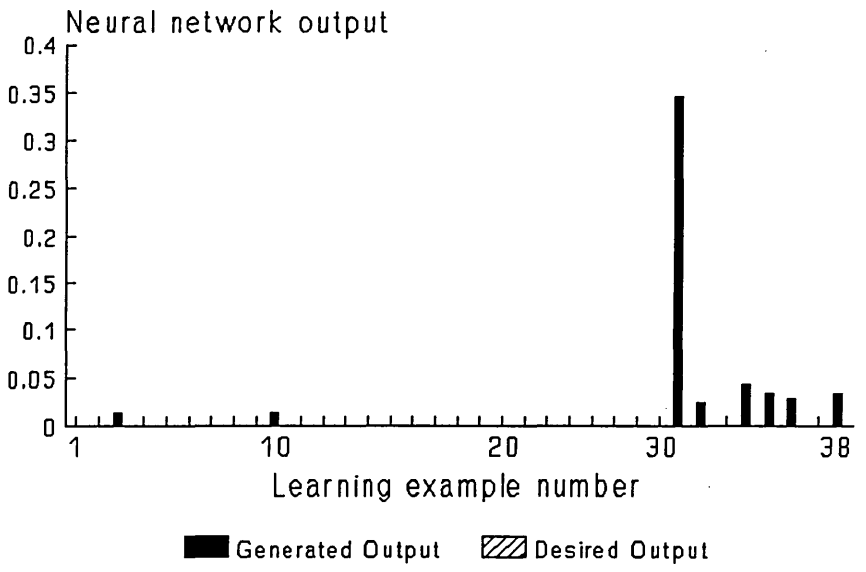
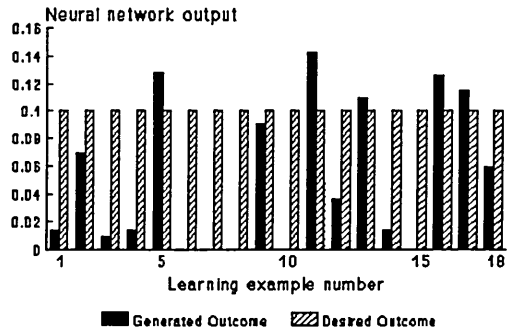
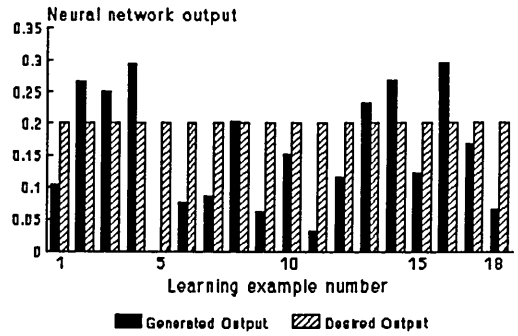


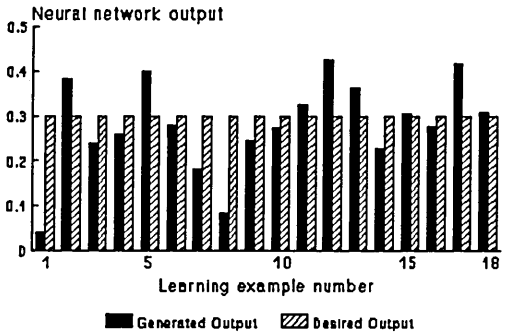
Figure 24: Output of the neural network for the examples in the training set with class 0



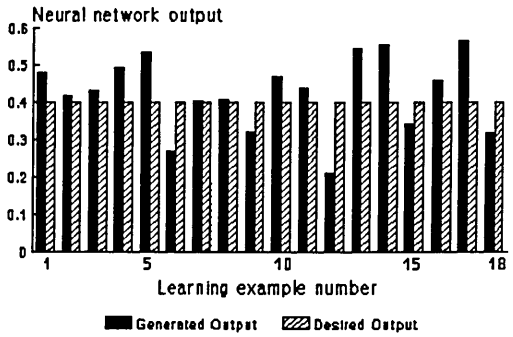
(a)



(b)

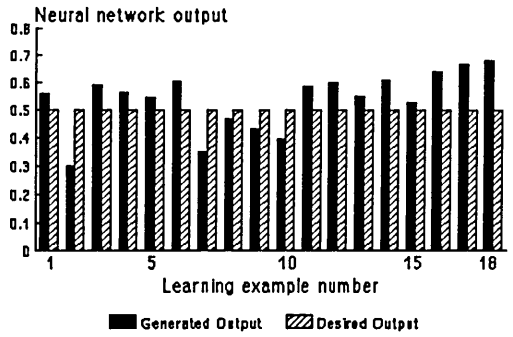


(c)

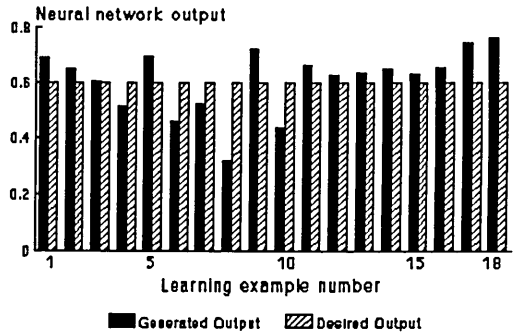


(d)

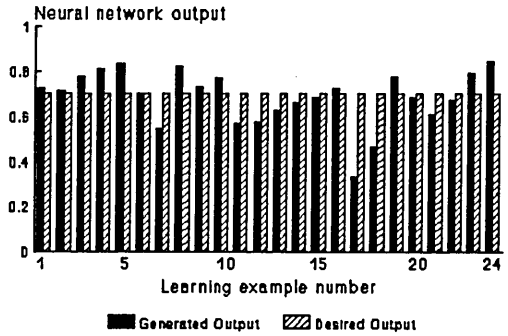
Figure 25 (a,b,c,d): Output of the neural network for the examples in the training set with class 0.1, 0.2, 0.3, 0.4 respectively



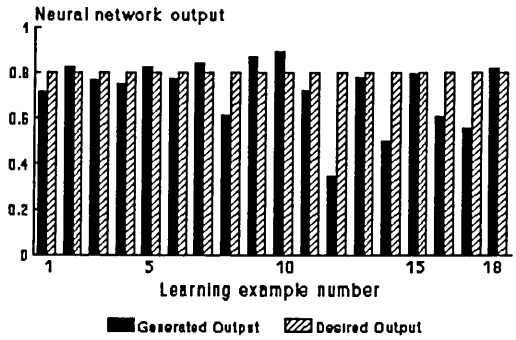
(e)



(f)

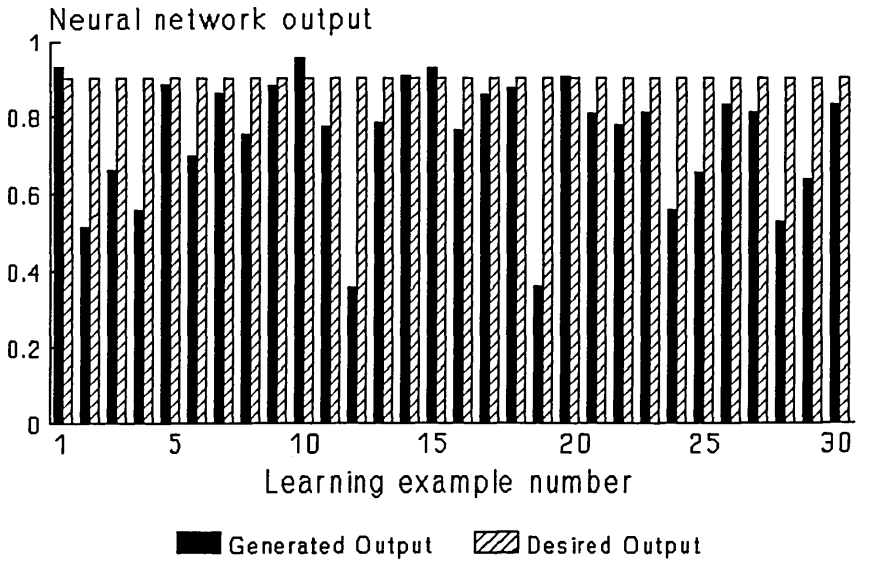


(g)



(h)

Figure 25 (e,f,g,h): Output of the neural network for the examples in the training set with class 0.5, 0.6, 0.7, 0.8 respectively



(i)

Figure 25 (i): Output of the neural network for the examples in the training set with class 0.9

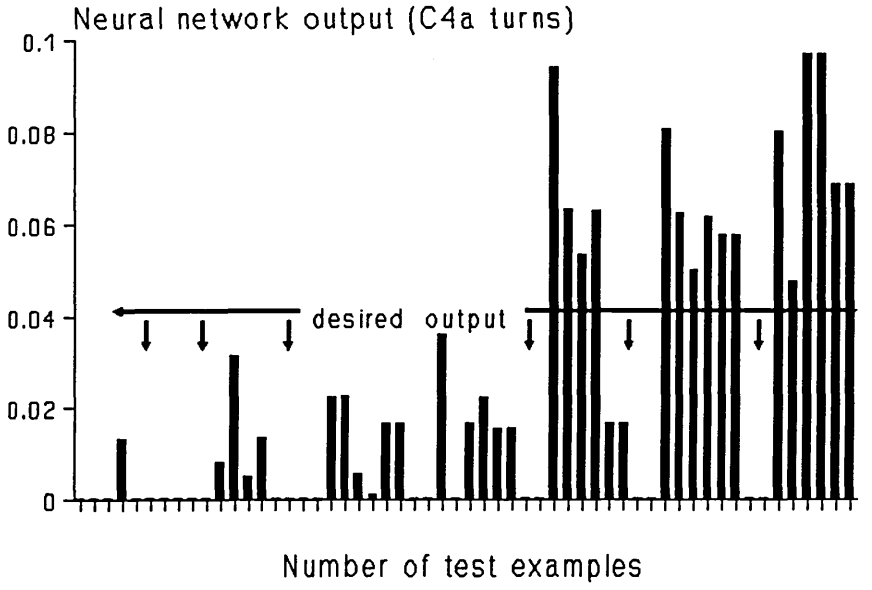


Figure 26: Testing the C_4 anti-clockwise network with previously unseen examples

predictions were wrong but the error was not dramatic. These predictions were made using examples where the C_7 component was mal-adjusted. This suggests that there is a certain interaction between the two adjustable components.

Is it then appropriate to stop learning here? Did further trials have to be carried out? Questions as such could only be answered after the network is tested live and only then the capabilities of the four neural networks will be known.

8.8 Conclusions

It was decided to employ neural networks with the back-propagation learning rule for the third search of the stopband region. As a training set it was found necessary to employ data from the process of de-tuning of just one filter. The topology of the network was determined empirically but for this particular domain two hidden layers were found to be required. One such neural network was constructed for each component/direction combination. The testing of the learning set showed that a number of predicted values are not very close to the desired values but at the same time not dramatically erroneous. Agreeing or disagreeing with the learning set was not sufficient to determine whether the network made the correct judgement therefore it was decided to test the networks on the production line.

References

1. Gallant S.I., *Connectionist expert systems*, Communications of the ACM, Vol. 31, No. 2, pp. 152-168, 1988.

2. Rich E., *Expert systems and neural networks can work together*, IEEE Expert, Vol. 5, No. 5, pp. 5-7, 1990.

3. Minsky M., *Logical versus analogical or symbolic versus connectionist or neat versus scruffy*, AI Magazine, Vol. 12, No. 2, pp. 34-51, 1991.

4. Mozer M.C., and Smolensky P., *Using relevance to reduce network size automatically*, Connection science, Vol. 1, No. 1, pp. 3-16, 1989.

5. Gallinar P., *Some properties of linear multilayer perceptrons*, In: New developments in neural computing (Eds. J.G. Taylor and C.L.T. Mannion), Adam Hilger, pp. 201-216, 1989.

9.1 The AEK system for tuning crystal filters

The AEK system was developed to semi-automatically assist an operator during the tuning of crystal filters on the production line. This chapter describes the hardware and software that was used. The operating instructions are also given.

9.1.1 System components

Figure 27 shows the hardware and software used as well as the flow of information between the system components.

The crystal filter was placed on an HP8721A test chassis. The test chassis was connected to the HP3577A network analyser which was linked with an HP9816 computer using the IEEE488 Control bus. Additionally, an MS-DOS running Compaq 386/25 computer was employed. The Compaq computer contains the Xi-Plus expert system shell and the Knowledge-bases (see below for further information). The HP9816 computer contains the Neural network test module (see below for further information) and the HP-Basic program employed for interfacing.

9.1.2 Description of software

A brief description of the knowledge-bases, the neural network test module, and the HP-Basic program now follows.

1. Name of knowledge-base: *Search1*

The rules of this knowledge-base determine if (further) tuning of the stopband is required. The decision is based on some or all values of six peaks (as entered by the user) and their differences (as calculated by the system). All numerical values are assigned logical values taken from a set of eight possible

values. Appendix 5 contains a listing of the knowledge-base.

2. Name of knowledge-base: *Search2*

The rules of this knowledge-base advise which component and in which direction to turn. The decision is based on the values of the peaks already entered on Search1 or newly requested ones. The values are then re-assigned logical values taken from a set of four possible values. Appendix 6 contains a list of the knowledge-base.

3. Name of knowledge-base: *Numbers*

This knowledge-base determines if (further) tuning of the passband, and therefore of the filter is needed. The resolution is based upon the values of all attributes. The numerical values are assigned one logical value each selected from a set of eight possible values. Appendix 7 contains a list of the knowledge-base.

4. Name of knowledge-base: *Pamod*

This knowledge-base recommends which component and in which direction to adjust. The recommendation is based upon the values as entered previously in the Numbers knowledge-base and additional input is not required. The attributes are assigned logical values selected from a set of four possible values. Appendix 8 contains a listing of the knowledge-base.

5. Name of Neural Network test module: *Last_One*

The neural network test module is an HP-Basic program which contains commands for the interfacing with the test equipment in order to acquire the input in the form of 57 sampled points taken from the complete amplitude response. Furthermore, a data list of all the weights and threshold values for the neural networks are present which in turn are used to calculate the

output (i.e. how far to turn). This program is used only for the stopband and Appendix 9 contains a listing of the program.

6. HP-Basic program for interfacing: *Pband_2*

This is an HP_Basic program written in order to provide interface with the measuring equipment for the measurement and printing of the attribute values used for the tuning of the stopband and passband regions of the amplitude response. Appendix 10 contains a listing of the program.

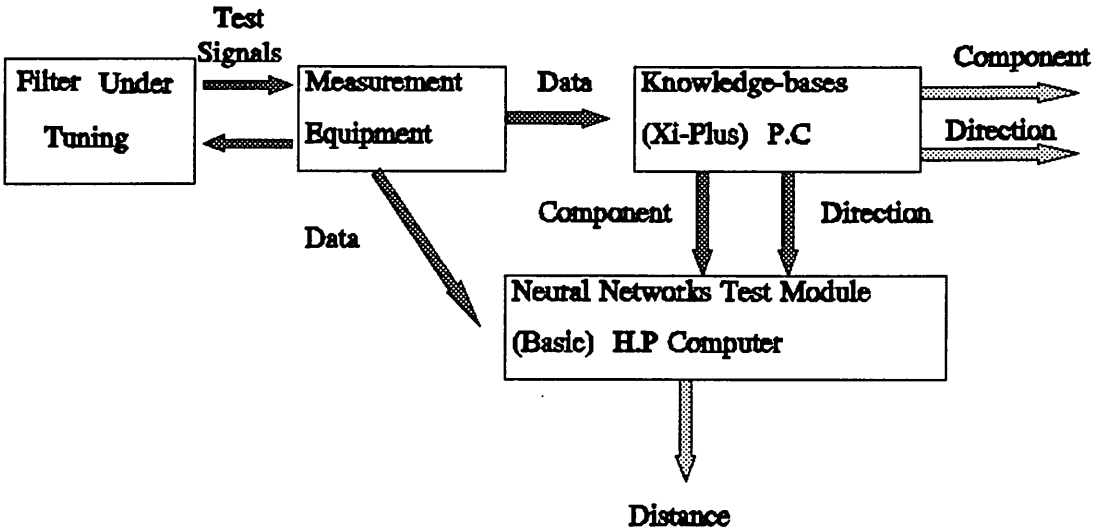
Figure 27 illustrates the configuration of the measuring equipment.

9.1.3 Operating instructions

The operator carries out the following procedure.

A filter is fitted into a temporary alignment can which is connected to the measurement equipment. This is set to the approximate frequency and range in order to measure the amplitude response. The Xi-Plus expert system shell and the first knowledge-base (i.e., stopband search 1) are loaded on to a MS-DOS computer. The execution of the first knowledge-base requires the operator to allocate and type the values of some or all of the peaks in order to determine if the stopband region needs tuning. The peak values are measured by the HP-Basic program. In the case where stopband tuning is needed the second knowledge-base (i.e., stopband search 2) is loaded automatically in order to determine which screw needs turning and in which direction to turn. The distance to turn is provided by the neural network test module program. The operator then makes the proposed adjustment. The first knowledge-base is automatically re-loaded and the operator enters the values taken from the new amplitude response. This is repeated until no further stopband tuning is required. Then the third knowledge-base (i.e., passband

search 1) is automatically loaded and this requires the operator to use the HP program to obtain values of the passband attributes. In the case where passband tuning is needed then the fourth knowledge-base (i.e., passband search 2) is automatically loaded and informes the operator which screw and in which direction to turn. The distance to turn is decided by the operator. When the adjustment is completed the third knowledge-base is automatically re-loaded and the new amplitude response is re-measured. This procedure reiterates until no further passband tuning is required. The user then connects a new filter and the process re-commences.



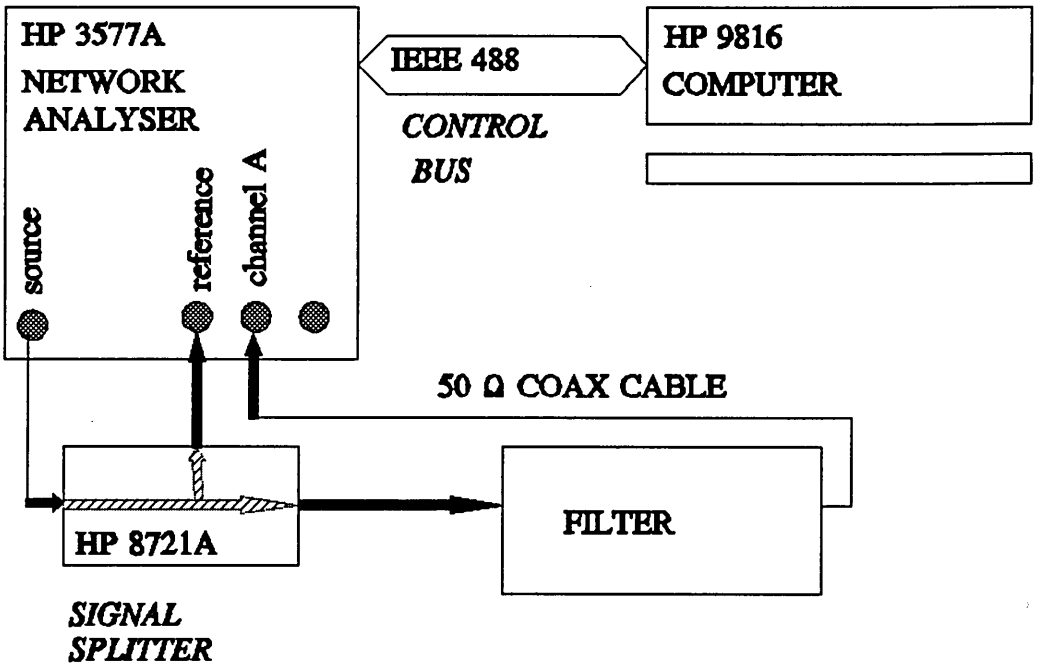


Figure 27 : Configuration of measuring equipment

Chapter Ten

Verification, Validation and Testing

10.1 Introduction

Clearly there is need to evaluate the performance of the system with *on-line* data rather than using examples of the learning set. Without any feedback in the form of assessment of performance one cannot judge if the tuning process has been learned. In this chapter the testing of the tuning of a number of filters is reported.

Issues such as consistency, validity and usability of the rule set of the expert system are discussed. Three approaches are investigated for the tuning of the stopband region. These approaches include the rule-based expert system attached to the human operator or to the neural network, and a stand-alone neural network. Whereas the tuning of the stopband region of a number of crystal filters was managed with all approaches the expert system - neural network approach provided an increase in the efficiency with which a solution was produced. The passband region was tuned in all cases but the operator was required to provide the distance to turn. Part of this work has been published[1].

10.2 Evaluation criteria

The evaluation of the performance of the system comprised three aspects, namely consistency, validity and usability. Consistency referred to the reliability of the system in the sense that the system produced similar answers to similar questions. Of course a consistent system did not necessarily guarantee that the rules were correct. This was examined under the validity aspect. The validity term referred to whether the system provided the results it should. The system was studied to see if it could deliver

everything that was true (completeness) and if what it delivered was true (soundness). Usability referred to the user-friendliness of the system, viz. can the system being used by a non-expert in terms of the human-computer interaction.

The AEK system was tested many times by the author and by the author and the expert together. The system scored high marks for consistency.

Initially, a small-scale testing took place to check for complete/correct rules. For each testing the system suggested one component/direction combination and the operator was allowed to veto prior to execution. In the presence of a strong disagreement the decision of the operator was selected. The execution was carried out by the operator. In subsequent tests, see Section 10.3, the operator was following the instructions of the system.

The other aspects are considered in the following sub-sections.

10.2.1 The completeness of the system

In the AEK system completeness was examined primarily by looking at the correctness of the final decision. That was when the system had arrived at an *end-of-process* conclusion, for stopband and/or passband, and the expert was consulted. He either then agreed or disagreed with the conclusion of the system. Second the correctness of the intermediate decisions (component, direction) were checked. Since there are numerous paths to the tuning only the prominently wrong were identified and corrected. This also applied for the third search when neural networks were employed.

10.2.2 The soundness of the system

The system contains rules with incomplete knowledge (empty rules). These

are rules which contain a number of conditions with an empty consequent, possibly because the particular combination of conditions was not covered within the learning set. When such a rule appears the consequent part is overwritten with the action the operator took.

The system also contains rules with conflicting knowledge (clash rules). These are rules delivered by examples from the learning set with different classes. When such a rule appears all clashing actions are reported and the decision is left to the operator. The decision of the operator can differ from the ones advised by the system. The consequent part of the rule is amended in order to contain the new action. It was hoped in this way either some actions would disappear or new rules would be discovered.

10.2.3 The usability of the system

The user communicates with the expert system using a textual user interface. The interface displays questions such as *Please enter the value of the first peak* and some questions are accompanied by instructions. The value read in the measurement set by the operator or acquired using software has to be typed. Because the operator has to enter a numerical value (i.e the transformation into logical values is performed by the system) no other type of communication, e.g a menu, is provided. The operator can be given an initial training in using the shell. Facilities such as *trace* can be used if necessary to examine the reasoning behind a recommendation. No extended or complicated training is required for running the system. The system is programmed to guide the user.

The neural network system does not have any form of communication with the operator other than by providing a conclusion. The system is interfaced

with the measuring equipment and the sampling and calculation is performed within the software.

10.3 Definition of test cases

Testing of the tuning of a number of filters was undertaken using three different systems.

Case 1: Knowledge-based system plus user

The knowledge-based system provided advice for when to stop the tuning of the stopband otherwise the component to turn and the direction to turn. The user had to decide on how far to turn.

Case 2: Hybrid system (Knowledge-based system plus neural network

As for Case 1 but the distance to turn was indicated by the network.

Case 3: Neural network

Because each component/direction combination had a net associated with it then the outcome of the net was used to define all decision levels.

The above three cases were tested on-line. The neural network was also tested with data where the desired outcome was known beforehand.

Case 4: Neural network testing using artificially generated data

In this case a tuned filter was maladjusted by a known amount of turn using one of the two components in a particular direction. For example if the maladjustment was done using C_4 in a clockwise direction turning half a revolution then we would expect the system to provide an output of the same form but indicating the opposite direction.

10.4 Specific testing criteria

The following criteria were employed to compare the various systems:

- The average number of turns required for the entire tuning
- The number of successful tunings
- The number of unsuccessful tunings.

The last two criteria give rise to other criteria:

- The number of empty rules which arose at each tuning
- The number of clash rules which arose at each tuning.

The term tuning refers to stopband or to passband or to a complete tuning.

10.5 Example of AEK in action

The way the AEK system (only the expert system part) arrives at decisions is demonstrated by walking through the processing of one tuning case. The interaction of the system with the operator, and the formation and reporting of conclusions are also illustrated. Italic text is used to show the questions asked by the system. Bold text represents the answer typed by the operator. Comments in between are included within square brackets.

[The operator enters the Xi-Plus shell and loads the stopband application which in turn loads the first knowledge-base of the stopband. This knowledge-base determines the continuation or not of the stopband tuning. The loading of the knowledge-base is followed by the automatic execution of the outcome query.]

What is the value of the fourth peak? **1.402852**

[This question is asked in order to determine the distance between the first and the fourth peak. At the same time the fourth peak is assigned a logical value (left)]

What is the value of the first peak? **1.399494**

[The question is asked for the same reason as previously. The first peak is

assigned a logical value (right). Their distance (i.e. peak4-peak1) is calculated by the system and a logical value is specified (left). With these attribute values rule 1 of the system with the outcome of *carry-on* holds. At this point the demon rule (13) takes over and resets the current knowledge-base and loads the second knowledge-base of the stopband. This knowledge-base determines the component and the direction to be used. It is worth noticing at this stage that if the demon rule was absent the search of more rules which can apply would have been continued. This continuation would have resulted in one of three situations:

- no other rule applies
- another rule with a *carry-on* outcome applies
- a rule with an *end-of-process* outcome applies.

The first two situations clearly indicate waste of processing time. As for the third situation it was decided in case of such a clash the *carry-on* rule to have priority. Therefore, the demon is required.]

Enter the value of the second return level: **48.9**

[Notice that the term *return level* was used in order to employ the phraseology of the user and it is synonymous to the term *negative peak*, as used previously.

The system tries to determine the logical value of the second return level for use in rule 122. The allocated logical value (left) satisfies the second condition of the rule. The first condition of the rule involves the value of the first peak. Since the numerical value was already entered in the first knowledge-base the system realises there is no need to question again. The system treats the value as volunteering information and automatically allocates a logical value

(fright). It is worth remembering that the configuration of the logical values of the two knowledge-bases are different (See Chapter 6).]

Enter the value of the first return level: **53.2**

[The system checks whether the third condition of rule 122 can be satisfied. The attribute is allocated the logical value of *close-left* which does not satisfy the condition. Consequently, the search for another rule which can be satisfied with the present information continues. Rule 128 has all its conditions satisfied therefore its outcome C_4a (i.e. turn component C_4 anticlockwise) is reported. Then the system resets the current knowledge-base and re-loads the first. In this case this is repeated twice until the end of tuning of the stopband is reached. The system then resets the current knowledge-base and the first knowledge-base of the passband is loaded. This knowledge-base discovers if there is a need for any tuning of the passband. It is worth mentioning the following:

- The breaking-up of the tuning application into four knowledge-bases was necessary due to the limited memory capacity of the computer used.
- The operator had to locate the peaks and the return levels on the oscilloscope manually. Then the values were read on the display.]

Enter the value of the ripple: **5.4**

[The system asks a series of questions in order to obtain the values of all the attributes used in the passband. The values are calculated using a program written in HP-Basic and they are displayed on the screen of the computer. Logical values are then assigned to the attributes by the system. Ripple in this example takes the logical value of *far-right*.]

Enter the value of the insertion loss: **2.3**

[The associated logical value is *ok*]

Enter the value of the low passband: **0.6**

[The associated logical value is *ok*]

Enter the value of the high passband: **8.1**

[The associated logical value is *right*]

Enter the value of the carrier rejection: **28.7**

[The associated logical value is *far-ok*]

Enter the value of the low stopband: **59.6**

[The associated logical value is *far-ok*]

Enter the value of the high stopband: **56.9**

[The associated logical value is *middle-ok*]

Enter the value of the low stopband return: **59.3**

[The associated logical value is *far-ok*]

Enter the value of the high stopband return: **53.3**

[The associated logical value is *middle-ok*]

[At this stage the decision of the system is that the passband requires tuning. The decision was based on the fact that the ripple and the high passband attributes are not within specification. As a result of this the system resets the current knowledge-base and loads the second knowledge-base of the passband. This knowledge-base provides the component to be used and the direction to turn. Since the two knowledge-bases share the same configuration of logical values it is not necessary to ask any further questions. The system simply searches through the rule set using the allocated logical values from the previous knowledge-base. In this case the reported component is T_3 in an anti-clockwise direction (rule 198). The system then resets the current

knowledge-base and the first knowledge-base of the passband is loaded once again. This procedure was repeated twice until the end of the passband tuning and subsequently of the filter was reached. Then the user was requested to connect another filter and the process started again.]

10.6 Case 1 testing for the stopband

The stopband region of the magnitude response of the filter was tuned using the expert system and an expert human operator. The system advised which component to turn and in which direction and the operator turned as much or little as he wanted. In case of turning too far the operator was asked not to correct his action by backtracking. He was also encouraged to provide any comments on the decision of the expert system.

10.6.1 Starting positions of the tunable components

Theoretically, the expert system should be able to assist the operator irrespective of how distorted the magnitude response be. The distortion of the response obviously depends on the position of the screws. For that reason, the testing involved filters for which the initial positions of their components were different (Table 56).

10.6.2 Presentation and discussion of results

<u>Distribution of successful tunings</u>

Table 57 shows the total number of attempts made together with the number of successful and unsuccessful tunings categorised per component configuration.

Table 56: Initial position of tunable components during stopband testing (case 1)		
Configuration	C_4 position	C_7 position
A	screwed-in	screwed-in
B	screwed-out	screwed-out
C	screwed-in	screwed-out
D	screwed-out	screwed-in
E	halfway	halfway
F	as found	as found

‡ Case 1 designates the knowledge-based plus human system

Table 57 : Tuning attempts per configuration using case 1 system for the stopband			
Configuration	Number of attempts	Successful tunings	Unsuccessful tunings
A	6	3	3
B	4	4	0
C	1	1	0
D	1	1	0
E	1	1	0
F	8	8	0
TOTAL:	21	18	3

‡ Case 1 designates the knowledge-based plus human system

The system failed to tune a filter in three cases, all of them with component configuration (A). In all three cases the tuning was abandoned due to the factthat the system seemed to supply oscillatory advice (i.e. turn C_4 anti-clockwise followed by C_4 clockwise). It was interesting to see how these three cases contrasted with the three cases of the same component configuration where tuning was achieved. Again, it was expected that the system, ideally, should have been able to tune the stopband even if a different number of turns were employed for the same situation. Observing all cases it seemed that the system managed to tune each filter when the operator turned about three revolutions. When he turned less, except for one case, the system failed to achieve the objective. Therefore, some blame could also be assigned to the amount of turning, hence the need for the provision of some indication of how far to turn.

A number of encouraging points were made concerning the overall performance of the system:

- When one or both components were screwed all the way in the system always gave the direction as anti-clockwise.
- When one or both components were unscrewed all the way out the system always gave the direction as clockwise.
- The system did not suggest the end of the stopband tuning where it was obviously not appropriate.

<u>Average number of adjustments</u>

Table 58 shows the average number of adjustments taken for the tuning of the stopband. These results refer only to the successful tunings. The minimum and maximum number of adjustments are also recorded. The

Table 58 : Average number of adjustments using case 1 system for the stopband	
Average number of turns:	3.22
Minimum number of turns:	1
Maximum number of turns:	7

‡ Case 1 designates the knowledge-based plus human system

reader should recognise that these figures depend on two related factors: (i) The revolutions turned, and(ii) the position of the tunable components. For example, when both components were screwed all the way in it took on average 3.7 adjustments whereas when they were all the way out it took 5.3 adjustments on average.

Examination of end of tuning events

Table 59 shows the rules that were executed for the 18 events where the tuning was ended.

Only 5 out of the 12 *end-of-process* rules were used. Although only a few attributes were considered by each rule (Column 4 of Table 59) it is worth noticing that the system had at its disposal the values for all thirteen attributes. This indicated that *carry-on* rules did not apply, otherwise the tuning would had been continued. Initially, rule 45 was an *empty* rule but its action was replaced by *end-of-process* (by the system builder). In all cases the expert operator commented that the right advice was given.

Examination of tuning steps by the empty rule category

It was felt appropriate to examine not only the validity of the proposed actions but also the completeness of the system. Therefore the number of situations where the system could not provide advice (*empty* rules) or provided more than one advice (*clash* rules - see following heading) were also inspected. Table 60 shows the number of occurrences by category of action. Table 61 displays, in more detail, the *empty* rules that were executed together with the action taken by the operator. These rules, in most cases, were applied on more than one occasion but only the first time is counted as an empty step. It is noticeable that the operator always took the same action.

Table 59 : End of tuning rules executed using case 1 system for the stopband			
Rule number	Occurrences	Rule confidence	Number of attributes used
50	1	68.2	2
27	8	91.7	6
119	4	60.0	1
4	3	87.5	3
45	2		5

‡ Case 1 designates the knowledge-based plus human system

Table 60 : Number of occurrences per action using case 1 system for the stopband		
Action given by the system	Number of occurrences	Number of occurrences (%)
Empty	4	6.9
Clash	12	20.7
Component and Direction	42	72.4
TOTAL:	58	100.0

Table 61 : Number of occurrences for situations where the system could not provide advice for the tuning of the stopband (case 1)	
Rule number	Action taken
106	C_7 clockwise × 6
100	C_7 clockwise
53	C_7 anti-clockwise × 3
87	C_4 clockwise × 2

‡ The character 'x' as in C_4 clockwise denotes multiple occurrences
‡ Case 1 designates the knowledge-based plus human system

Also 3 out of 4 empty actions were replaced with a C_7 action and/or a clockwise direction (by the system builder). Since 41 per cent of the trainingset were examples with a clockwise class it was expected that more situations where a clockwise turn was required would occur.

Examination of tuning steps by the clash rule category

Table 62 shows the *clash* rules that were executed together with the action taken by the operator. The third column displays the actions taken previously (i.e. under training) under the same circumstances. One can observe that a clash of actions also occurred during the testing. Re-appearing *clash* rules were left as before, i.e. with a clash outcome. When such a rule executes it reports the diverse actions taken previously, including training and testing occurrences, and the decision is left to the operator. This is a possible shortcoming of the system and the need arises for discovering the reasons behind such diversity.

Examination of tuning steps by component and direction category

For a significant number of tuning steps (72.4% - Table 60) the system made the decision of which component to turn and in which direction. Table 63 shows the distribution per rule. Rules are in ascending order. Table 64 shows the total number of rules and their occurrences. The third column shows the available number of rules for each combination. The comparison of the last two columns of Table 64 shows that the frequency of a particular action occurring is close to its associated percentage of available rules in the rule base (especially for the clockwise direction). Additionally, one can observe that approximately 51 per cent of the available rules came into use. Combinations with an anti-clockwise direction used half or less of their available rules.

Table 62 : Number of occurrences of situations where the system provided conflicting outcomes for the tuning of the stopband (case 1)

Rule number	Action taken during testing	Action taken during training
90	C_4 clockwise	C_4 clockwise × 4
	C_4 anti-clockwise	
	C_7 clockwise	C_7 clockwise × 3
88	C_4 clockwise × 2	C_4 clockwise × 2
	C_7 clockwise	C_7 clockwise
93	C_4 anti-clockwise × 2	C_4 anti-clockwise
	C_4 clockwise	C_7 clockwise
97	C_4 clockwise	C_4 clockwise
	C_4 anti-clockwise	C_4 anti-clockwise
113	C_7 clockwise	C_7 clockwise
		C_4 clockwise

‡ The character 'x' as in C_4 clockwise denotes multiple occurrences
‡ Case 1 designates the knowledge-based plus human system

Table 63 : Distribution per rule for the tuning of the stopband using case 1 system

Rule number	Number of occurrences	Combination
2	1	C_4 anti-clockwise
5	2	C_4 clockwise
9	1	C_7 clockwise
15	1	C_4 clockwise
17	3	C_7 anti-clockwise
18	4	C_7 anti-clockwise
42	1	C_4 anti-clockwise
44	3	C_7 anti-clockwise
53	2	C_7 anti-clockwise
70	2	C_4 anti-clockwise
76	1	C_4 anti-clockwise
87	1	C_4 clockwise
101	5	C_7 clockwise
102	1	C_4 clockwise
103	2	C_4 clockwise
106	2	C_7 clockwise

Table 63 continued : Distribution per rule for the tuning of the stopband using case 1 system		
Rule number	Number of occurrences	Combination
115	2	C_7 clockwise
120	2	C_7 clockwise
128	2	C_4 anti-clockwise
132	1	C_4 anti-clockwise
134	1	C_4 anti-clockwise
144	2	C_4 anti-clockwise

‡ Case 1 designates the knowledge-based plus human system

Table 64 : Total distribution of rules per combination using case 1 system for the stopband

Combin-ation	No. of rules	Occur-rences	Available number of rules	Occur-rences (%)	Availa-bility (%)
C_4 a	8	11	16	26.2	37.2
C_4 c	5	7	7	16.7	16.3
C_7 a	4	12	11	28.6	25.6
C_7 c	5	12	9	28.6	20.9

‡ The character 'c' as in C_4 c denotes the clockwise direction
‡ The character 'a' as in C_4 a denotes the anti-clockwise direction
‡ Case 1 designates the knowledge-based plus human system

Therefore the possibility of eliminating a number of redundant rules after some further extensive on-line testing arises.

10.7 Case 2 testing for the stopband

The testing results with case 1 were obtained with an experienced operator who had some idea of how far to turn the screws. The objective though was to construct a system which could be used by anyone irrespective of his or her level of experience and proficiency. An inexperienced operator would probably turn the screws too far or too little. This could result in a larger number of iterations and while the tuning would eventually be done it would take longer. For this reason the stopband region of the magnitude response of the filter was tuned using the expert system and the neural networks. The expert system advised, as before, on which component to turn and in which direction but the operator turned the distance given by the appropriate neural network. For example, if the expert system indicated C_4 anti-clockwise, then the C_4 anti-clockwise network was used.

10.7.1 Starting position of the tunable components

For the reasons given in Section 10.6.1 the testing involved filters which had different initial positions of their components (Table 65).

10.7.2 Presentation and discussion of results

<u>Distribution of successful tunings</u>

Table 66 shows the total number of attempts made together with the number of successful and unsuccessful tunings arranged per component configuration.

Table 65 : Initial positions of tunable components during testing using case 2 system (stopband)

Configuration	C_4 position	C_7 position
A	screwed-in	screwed-in
B	screwed-out	screwed-out
C	screwed-in	screwed-out
D	screwed-out	screwed-in
E	halfway	halfway
F	as found	as found
G	halfway	screwed-in
H	screwed-in	halfway
I	screwed-in	as found
J	screwed-out	as found

‡ Case 2 designates the knowledge-based plus neural network system

Table 66 : Tuning attempts per configuration using case 2 system (stopband)			
Configuration	Number of attempts	Successful tunings	Unsuccessful tunings
A	1	0	1
B	1	1	0
C	1	1	0
D	1	1	0
E	3	3	0
F	8	7	1
G	1	0	1
H	1	1	0
I	1	1	0
J	1	0	1
TOTAL:	19	15	4

‡ Case 2 designates the knowledge-based plus neural network system

The hybrid failed to tune a filter in four cases. Unlike the results reported previously the failures this time occurred with more than one configuration. In all four cases the tuning was abandoned due to the choice of the component and direction by the expert system rather than the output of the neural network. Of course it is possible that the wrong outcome of a neural net in a preceding step contributed in the first place but this was uncheckable. It is interesting to notice that in only one case the outcome of the expert system was C_4 anti-clockwise while for the rest of the cases the C_7 clockwise combination was given. In these situations the system would have continued advising C_7 clockwise despite the fact that it was obviously the wrong choice. It was likely that the blame can be assigned to the C_7 component rather than the C_4 component. The latter component had a different position for each unsuccessful tuning and they covered all the possible testing positions. On the other hand it seemed that C_7 created problems when it was placed half-way or screwed all the way in.

The observations made with case 1 testing about the choice of direction when one or both components were in their extreme positions applied here also.

Average number of adjustments

Table 67 shows the average number of adjustments required for the tuning of the stopband considering only the successful tunings. The minimum and maximum number of adjustments are also shown. Comments made in the respective section of case 1 testing still apply.

It is interesting also to compare case 1 and case 2 tests for those filters with configuration (F). In both tests the number of attempts was the same (8) but case 1 system successfully tuned all attempts whereas case 2 system failed

Table 67 : Average number of adjustments using case 2 system (stopband)	
Average number of turns:	3.53
Minimum number of turns:	1
Maximum number of turns:	8

‡ Case 2 designates the knowledge-based plus neural network system

in one situation. There is a slight difference in the average number of adjustments and the maximum and minimum number of turns required in favour of the case 2 system (Table 68).

Examination of end of tuning events

Table 69 shows the rules that were executed for the 15 situations where the tuning was terminated. Six out of the 12 available *end-of-process* rules were utilised. A peculiarity was that with the exception of two rules (119, 27) the rest of the rules had not appeared with case 1 testing. (Table 59). It is also worthwhile to examine the generated outcomes of the neural networks at these 15 situations. Table 70 shows the generated outcomes of each network for each termination of tuning. The expected outcomes are values close to zero. These rules indicate that, allowing a ± 0.1 error, for the majority of cases the expert system and the neural networks agreed. The disagreements arose due to the anti-clockwise networks. That was something experienced for case 4 testing as will be discussed in Section 10.10.

Examination of tuning steps by the empty rule category

Table 71 shows the number of occurrences for each action advised by the system. At this testing an increase in the number of occurrences of *empty* rules can be observed in conjunction with a significant drop of execution of clash rules.

Combining the results of Tables 60 and 71 it seems that the expected likelihood of the system giving a component -direction combination is 71.2% with the rest of the time (28.8%) the output being distributed approximately equally between *empty* and *clash* rules (Table 72). Table 73 shows the occurrences of empty situations distributed per rule but only the first time an

Table 68 : Average number of adjustments using case 1 and case 2 systems for the tuning of the stopband (only *as-found* configuration is considered)		
	Case 1	Case 2
Average number of turns:	3.5	2.9
Minimum number of turns:	2	1
Maximum number of turns:	7	5

Table 69 : End of tuning rules using case 2 rules (stopband)			
Rule number	Occurrences	Rule Confidence (%)	Number of attributes used
120	2	54.0	1
46	2	78.8	2
121	2	56.0	2
29	3	87.5	4
119	5	60.0	1
27	1	91.7	3

‡ Case 1 designates the knowledge-based plus human system
‡ Case 2 designates the knowledge-based plus neural network system

Table 70 : Neural networks generated outcomes when tuning was terminated								
Network	Generated values							
C_4 c	0	0.1	0.1	0.1	0.1	0.1	0.1	0.1
C_4 a	0	0.3	0	0.2	0.2	0	0	0.1
C_7 c	0	0	0	0	0	0	0	0
C_7 a	0	0	0	0	0	0	0	0.6

Table 70 continued : Neural networks generated outcomes when tuning was terminated							
Network	Generated values						
C_4 c	0	0	0	0.1	0	0	0
C_4 a	0.1	0.4	0	0	0	0.1	0
C_7 c	0	0	0	0	0	0	0
C_7 a	0	0.1	0	0	0	0	0

‡ The character 'c' as in C_4 c denotes the clockwise direction
‡ The character 'a' as in C_4 a denotes the anti-clockwise direction

Table 71 : Number of occurrences per action using case 2 system (stopband)		
Action	Number of occurrences	Number of occurrences (%)
Empty	13	24.1
Clash	3	5.6
Component and Direction	37	68.5
TOTAL:	53	100

Table 72 : Number of occurrences per action given by case 1 and case 2 testing (stopband)		
Action	Number of occurrences	Number of occurrences (%)
Empty	17	15.3
Clash	15	13.5
Component and Direction	79	71.2

‡ Case 1 designates the knowledge-based plus human system
‡ Case 2 designates the knowledge-based plus neural network system

Table 73 : Number of occurrences of situations where the case 2 system could not provide an advice (stopband)	
Rule number	Action taken
118	C_4 anti-clockwise
107	C_4 clockwise, C_4 anti-clockwise
5	C_4 clockwise
119	C_4 anti-clockwise
98	C_4 clockwise
95	C_7 anti-clockwise
58	C_7 anti-clockwise $\times$ 6
021	C_7 clockwise $\times$ 2
43	C_4 clockwise $\times$ 2
85	C_7 clockwise
10	C_4 anti-clockwise
19	C_4 anti-clockwise
1	C_7 anti-clockwise

‡ The character 'x' as in C_4 clockwise x 2 denotes multiple occurrences
‡ Case 2 designates the knowledge-based plus neural network system

empty situation was encountered counted as an empty step. A large number of the *empty* rules appeared only once which makes it difficult to appraise the proposed action. For rules that emerged more than once, except in one case (Rule 107), the operator followed the same action.

Examination of tuning steps by the clash rule category

Table 74 shows the clash rules that were executed and the action taken by the operator. The third and fourth columns display the actions taken during training and where applicable, testing case 1 respectively. Whereas the actions taken for circumstances covered by rule 105 seem to be different it is interesting to examine the other *clash* rule. The proposed actions for rule 88 were equally distributed between the two components with the direction given as clockwise for all cases. Rule 88 appeared in five cases during testing. The common link being that the execution of the rules occurred either at the beginning of the tuning(i.e. the first step) or at the second tuning step. What one can conclude is that both actions were correct and most probably both actions have to be implemented.

Examination of tuning steps by component and direction category

Table 75 shows the distribution per rule of the 37 circumstances where the system provided an advice. Rules are in ascending order. Four rules (14, 41, 86, 109) had not been executed with case 1 testing. Table 76 shows the total number of rules, their occurrences and the available number of rules in the rule base for each combination. Three observations can be made:

- The low number of C_4 anti-clockwise occurrences. Taking into account the 4 instances of *empty* rules (which were modified to C_4 anti-clockwise) and the one instance of *clash* rule (which again was modified to C_4 anti-clockwise)

Table 74 : Number of situations per rule where the case 2 system provided conflicting advices (stopband)

Rule no.	Testing	Training	Testing (case 1 system)
88	C_7 clockwise × 2	C_7 clockwise	C_7 clockwise
		C_4 clockwise × 2	C_4 clockwise × 2
105	C_4 anti-clockwise	C_4 clockwise	
		C_7 clockwise	

‡ The character 'x' as in C_4 clockwise x 2 denotes multiple occurrences
‡ Case 2 designates the knowledge-based plus neural network system

Table 75 : Distribution of outcomes per rule using case 2 system (stopband)		
Rule number	Number of occurrences	Combination
14	1	C_7 clockwise
15	1	C_4 clockwise
18	4	C_7 anti-clockwise
41	2	C_7 anti-clockwise
86	1	C_4 clockwise
87	6	C_7 clockwise
101	9	C_7 clockwise
102	5	C_4 clockwise
103	2	C_4 clockwise
109	1	C_4 anti-clockwise

‡ Case 2 designates the knowledge-based plus neural network system

- The C_4 component was maladjusted anti-clockwise half a turn (i.e. 0.50).
- The C_7 component was maladjusted anti-clockwise half a turn (i.e. 0.50).

Table 76 : Total distribution per combination using case 2 system (stopband)					
Combination	No. of rules	Occurrences	Available rules	Occurrences (%)	Availability (%)
C_4 a	1	1	16	3.1	37.2
C_4 c	4	9	7	28.1	16.3
C_7 a	2	6	11	18.8	25.6
C_7 c	6	06	9	50.0	20.9

‡ Case 2 designates the knowledge-based plus neural network system
‡ The character 'c' as in C_4c denotes the clockwise direction
‡ The character 'a' as in C_4a denotes the anti-clockwise direction

results in a total of six occurrences. This still reflects a low occurrence considering that C_4 anti-clockwise rules constitute more than a third of the available rules.

- The high number of occurrences (50.0%) for the C_7 clockwise combination
- Only 23.3% of the available rules were executed.

Testing the tuning steps in terms of distance turned

During the testing one filter was found to be tuned, something which was recognised by the expert system and the neural networks. Then the tuned filter was mal-adjusted by the following actions:

- C_4 anti-clockwise 0.50 turns
- C_7 anti-clockwise 0.50 turns

Then one expects that the opposite actions (i.e. 0.50 clockwise) would return the magnitude response to the initial (i.e. tuned) position. The outputs of the hybrid system for each of the four neural networks were:

- Turn C_4 clockwise 0.56 turns.
- Turn C_7 clockwise 0.36 turns.
- Turn C_4 anti-clockwise 0.14 turns.
- Turn C_7 anti-clockwise 0.25 turns.

Therefore, in total the actions taken were:

- Turn C_4 clockwise 0.42 turns.
- Turn C_7 clockwise 0.11 turns.

These demonstrate that the hybrid system advised on the right direction for both components but only the C_4 networks approximately matched the initial maladjustment. The estimates of the C_7 networks were far below the expected adjustments. Another observation was that the C_4 networks, in particular,

provided two estimates (0.56, 0.14) rather than realising that a single distance around 0.50 clockwise turns would have been sufficient. Despite that in this case the difference between the estimates was implemented the following section will show that selecting the largest estimate for each direction works better.

10.8 Case 3 testing for the stopband

Since the outcome of each net incorporates the component/ direction combination they were employed to define all decision levels. For example, if the output of the four networks were:

- C_4 anti-clockwise network : 0.1 (i.e. 0.25 in real turns)
- C_4 clockwise network : 0.3 (i.e. 0.75 in real turns)
- C_7 clockwise network : 0.5 (i.e. 1.25 in real turns)
- C_7 anti-clockwise network : 0.6 (i.e. 1.50 in real turns)

then the outputs could be interpreted in one of the following two ways:

- Implement the difference between the directions for each component. For the above example this would mean

C_4 clockwise 0.2 (i.e. 0.50 in real turns)

C_7 anti-clockwise 0.1 (i.e. 0.25 in real turns).

This way the fact that each network was taught using a different training set was taken into account.

- Select the largest of each component. In this example, that would had meant

C_4 clockwise 0.3 (i.e. 0.75 in real turns)

C_7 anti-clockwise 0.6 (i.e. 1.50 in real turns).

The necessary adjustments for both components were to be implemented one

after the other. It was hoped that the pseudo-simultaneous adjustment would result in a reduction in the number of tuning steps necessary.

10.8.1 Implementing the differences

Five filters were employed. In one filter only the stopband tuning was successfully terminated. For the rest of the attempts the tuning was abandoned for various reasons. In two situations the differences were close to zero indicating that further tuning was not required which was obviously wrong. Figure 28 illustrates the tuned position and the two positions where the tuning was abandoned. In another situation the same component (C_7) was constantly chosen which resulted in the worsening of the position of the magnitude response (Figure 29). Finally in the last abandoned situation the reason was that the networks provided oscillating outputs (Figure 30).

10.8.2 Selecting the maximum output of each component

Table 77 shows the total number of attempts made together with the number of successful and unsuccessful tunings shown. The networks failed to converge towards a tuned position due to the assertion of the C_7 component in an anti-clockwise direction after the third tuning step. Figure 31 illustrates the fact that the continuation of the this selection resulted in worse responses. Table 78 shows the average number of adjustments taken jointly with the minimum and maximum number of turns.

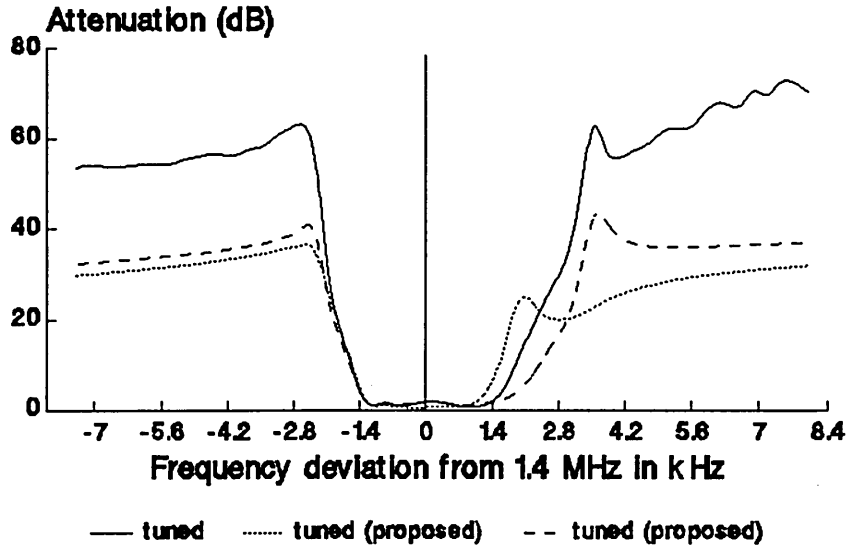


Figure 28: Illustration of two magnitude responses where the system proposed wrongly the end of tuning. The reference frequency is denoted as zero at the frequency axis.

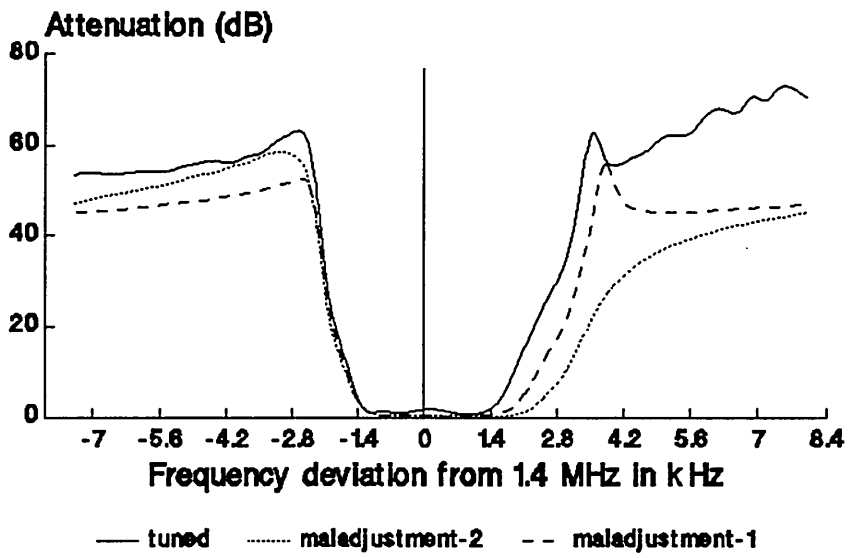


Figure 29: Illustration of deterioration of magnitude response due to the constant proposal of turning C_7 component. The reference frequency is denoted as zero at the frequency axis

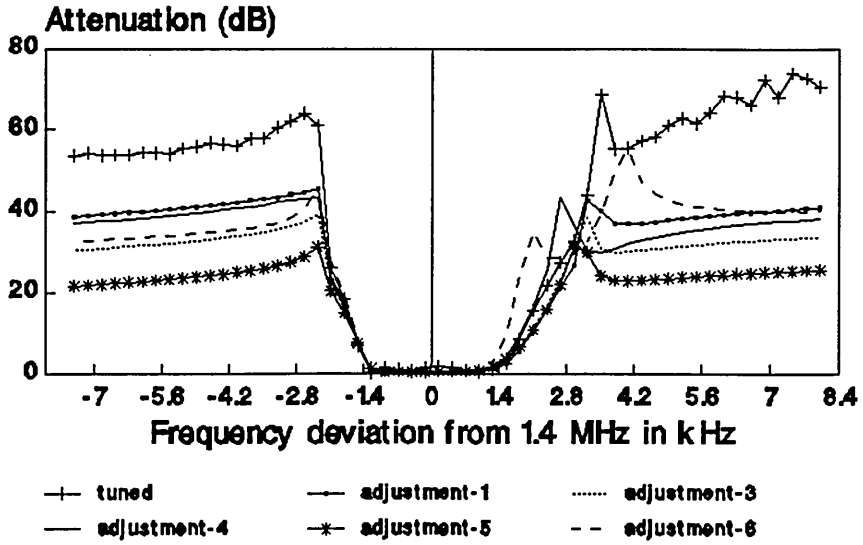


Figure 30: Illustration of oscillating adjustments resulting to oscillations. The reference frequency is denoted by zero at the frequency axis.

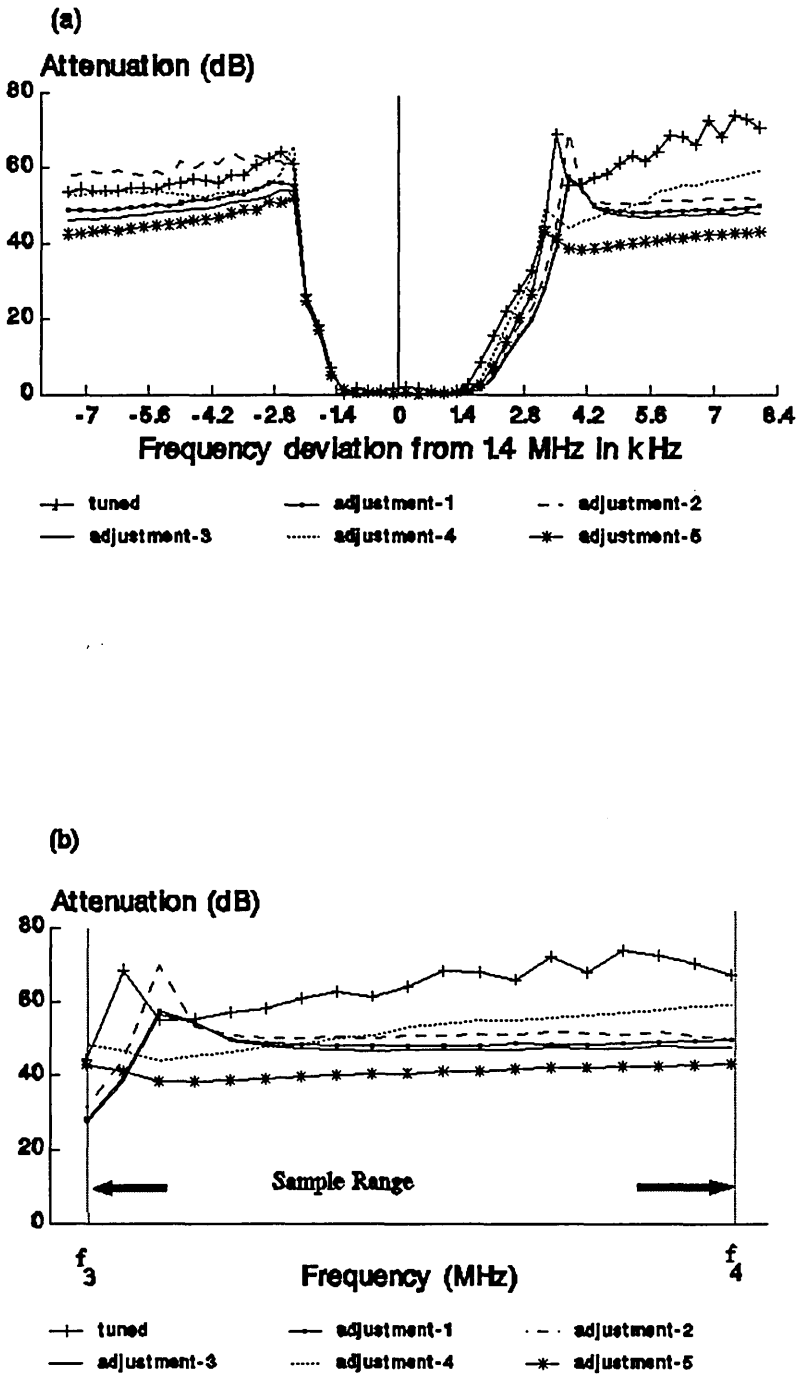


Figure 31: Illustration of (a) complete magnitude responses and (b) the right side of the responses where the proposal of turning the C_7 component resulted to non-converge. The reference frequency is denoted by zero at the frequency axis. Nineteen equally spaced values were sampled from the right hand side of the magnitude response between f_3=1.404 MHz and f_4=1.42 MHz, in steps of 0.051 MHz.

Table 77 : Number of attempts using case 3 system for the tuning of the stopband	
Number of attempts:	3
Successful tunings:	2
Unsuccessful tunings:	1

Table 78 : Average number of adjustments using case 3 system for the tuning of the passband	
Average number of turns:	10.5
Minimum number of turns:	7
Maximum number of turns:	14

10.9 Case 1 vs. Case 2 vs. Case 3 (stopband testing)

In Table 79 the number of successful and unsuccessful attempts made for the tuning of the stopband using the three systems are combined and shown. Testing with all three cases resulted in more successful than unsuccessful tunings but the most encouraging results were obtained with the case 2 testing. This was because of the lack of any human intervention (unlike case 1 testing). The results with the case 3 system, when using the nets with the largest outputs rather than their differences, were promising. Concrete conclusions about the case 3 system cannot arise due to the small number of attempts made and the various problems encountered. Table 80 combines and contrasts the three systems in terms of the number of turns required. The table compares the performances of the systems and of the human operator. The comparison shows that the use of any system did not necessarily reduce the required number of tuning steps but the expected benefit will be a reduction of the time an operator spends learning about the tuning procedure. This is apparent when comparing case 1 and case 2 systems. The results are comparable and encouraging. There is no need to have an experienced operator. At this stage it is preferable to use the case 2 system rather than case 3. The latter system seems to require more steps. There are two probable reasons for this. These are firstly the shortcomings of the C_7 anti-clockwise network as experienced during all the testing and secondly the single model solution effect (recall that only one tuned magnitude response was used in the learning set) which results in each network to aim to one and only solution

Table 79 : Number of successful and unsuccessful attempts of the tuning of stopband using case 1, 2 and 3 systems

	Case 1	Case 2	Case 3
Successful	85.7	78.9	66.7
Unsuccessful	14.3	21.1	33.3

‡ Case 1 designates the knowledge-based plus human system
‡ Case 2 designates the knowledge-based plus neural network system
‡ Case 3 designates the neural network system

Table 80 : Average number of turns for the tuning of the stopband using case 1, 2 and 3 systems

	Human	Case 1	Case 2	Case 3
Average number of turns:	3.67	3.67	3.22	10.5
Minimum number of turns:	1	1	1	7
Maximum number of turns:	9	9	7	14

‡ Case 1 designates the knowledge-based plus human system
‡ Case 2 designates the knowledge-based plus neural network system
‡ Case 3 designates the neural network system

space. This resulted in oscillating outputs.

From the test results shown above, it should be noted that it is possible for the hybrid system (case 2) and the connective equivalent (case 3) to tune the stopband region of the magnitude response. A decrease in the training time can be achieved with either system. However, each system has its own advantages. The case 2 system can generate basic explanations of its reasoning whereas the networks have a faster execution time despite the larger number of steps taken. Additionally, with the neural networks situations where knowledge would conflict or not exist cannot arise. Both systems are then promising but an extensive testing period would be required before they be introduced in the production line.

10.10 Case 4 testing for the stopband

Tables 81 and 82 show the results obtained when the neural networks were tested with data generated from three filters with single maladjustments. Tables 83 and 84 show similar results but with both components being maladjusted. Table 85 shows the output of each of the four networks when presented with a tuned filter. Observing the output of the neural networks in Tables 81 to 85 and allowing a ± 0.1 error rate, the following points can be made:

- The networks for learning C_4 and C_7 clockwise both give correct estimates.
- The network for learning C_4 anti-clockwise tends to under-estimate for values greater than 0.5 but worked well for one filter.
- The network for learning C_7 anti-clockwise does not perform well in general except in one case in which it worked correctly for values up to 0.7 but for greater values it provided conservative estimates.

Table 81 : Comparison of desired vs generated outputs						
	C_4 a network			C_7 a network		
Desired Values	Generated Values for test filter..			Generated Values for test filter..		
	1	2	3	1	2	3
0.1	0.1	0.1	0.1	0.0	0.0	0.0
0.2	0.1	0.1	0.1	0.0	0.1	0.0
0.3	0.2	0.2	0.1	0.1	0.5	0.2
0.4	0.4	0.3	0.3	0.1	0.5	0.2
0.5	0.6	0.4	0.3	0.1	0.5	0.2
0.6	0.7	0.4	0.4	0.1	0.6	0.2
0.7	0.8	0.5	0.5	0.2	0.6	0.2
0.8	0.9	0.5	0.5	0.2	0.6	0.2
0.9	0.9	0.6	0.6	0.2	0.6	0.2
1.0	1.0	0.6	0.6	0.2	0.6	0.2

‡ The character 'a' as in C_4a denotes the anti-clockwise direction

Table 82 : Comparison of desired vs generated output						
	C_4 c network			C_7 c network		
Desired Values	Generated Values for test filter..			Generated Values for test filter..		
	1	2	3	1	2	3
0.1	0.0	0.0	0.1	0.0	0.0	0.0
0.2	0.3	0.1	0.3	0.2	0.1	0.1
0.3	0.4	0.3	0.4	0.3	0.2	0.3
0.4	0.4	0.4	0.5	0.4	0.4	0.5
0.5				0.5	0.5	0.6
0.6				0.7	0.6	0.7

‡ The character 'c' as in C_4c denotes the clockwise direction

Table 83 : Comparison of desired vs generated output			
C_4 a network		C_7 a network	
Desired Values	Generated Values	Desired Values	Generated Values
0.1	0.1	0.1	0.0
0.2	0.1	0.1	0.0
0.3	0.1	0.1	0.0
0.4	0.3	0.1	0.1
0.5	0.3	0.1	0.2
0.6	0.4	0.1	0.3
0.7	0.5	0.1	0.3
0.8	0.6	0.1	0.3
0.9	0.6	0.1	0.4
1.0	0.6	0.1	0.5

‡ The character 'a' as in C_4a denotes the anti-clockwise direction

Table 84 : Comparison of desired vs generated output			
C_4 c network		C_7 c network	
Desired Values	Generated Values	Desired Values	Generated Values
0.1	0.0	0.1	0.0
0.2	0.1	0.1	0.0
0.3	0.1	0.1	0.0
0.4	0.2	0.1	0.0

Table 85 : Generated output with tuned filters									
Desired Values	Sample of generated values								
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.1	0.0	0.1	0.0	0.0	0.1	0.0	0.1
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

‡ The character 'c' as in C_4c denotes the clockwise direction

- All networks recognise a tuned state.

It seems that networks for learning the clockwise maladjustment for both components operated better. It is noticeable that both had fewer examples in their learning sets and less classes were represented than for the ones with anti-clockwise maladjustments. This testing also indicated that there is an interaction between the two components and if the neural networks were to be used on their own then the four learning sets must be combined together. This though will create problems as already have been discussed in Section 8.5.

10.10.1 Experiments to investigate the ± 0.1 error

Two types of experiments were performed in order to identify the probable source of the ± 0.1 error. First, multiple measurements were taken from an amplitude response without any adjustments in between. In that experiment the amplitude response represented a tuned state therefore the outcome of the neural network test module (See Section 10.1.1 for further explanation) should produce four values equal or close to zero indicating that no tuning is needed. The outcome of this experiment was expected to show if the ± 0.1 error could be allocated to the sampling. The second experiment involved adjusting one component in an arbitrary direction and then turning the same screw the opposite direction using the same number of turns. The goal of this experiment was to investigate if blame can be designated to the actual turning of the screw, owing either backlash in the screw or the inexactness of the operator.

Table 86 presents the results of the first experiment which showed that there is a certain element of error due to sampling but it is negligible. Table 87 provides the results of the second experiment for when only the C_4 component was adjusted in an anti-clockwise direction. The table presents the output of each of the four neural networks for two trials. For each trial four rows show the output of the networks initially, when mal-adjusted, when adjusted back and finally the difference between the predictions. The table shows that the turning of the screw affects mainly the performance of the C_4 anti-clockwise neural network approximately 0.17 turns. This of course does not mean that other combinations of adjustments would not result to similar results for the other networks. The number 0.17 is the mean of the two differences which is in excess of the ± 0.1 error as experienced during the case 4 testing. The error can then be allocated to the screw backlash.

10.11 Case 1 testing for the passband

The passband region of the magnitude response of the filter was tuned using the expert system and an expert human operator. The system advised on the component to employ and in which direction to turn. The operator turned as much or as little as he wanted.

10.11.1 Presentation and discussion of results

<u>Distribution of successful tunings</u>

It is worth remembering that at the start of each passband tuning the following were true:

- The three tunable components (i.e. T_1, T_2, T_3) for the passband were screwed all the way in.

Table 86 : Neural networks output when sampling a amplitude response without any adjustment in between

	Network output for...			
Sampling Number	C_4 anti-clockwise	C_4 clockwise	C_7 anti-clockwise	C_7 clockwise
1	0.052	0.007	0.091	0.065
2	0.051	0.008	0.085	0.065
3	0.051	0.007	0.105	0.070
4	0.052	0.0	0.105	0.064
5	0.052	0.012	0.095	0.068
6	0.052	0.011	0.097	0.070
Mean	0.052	0.008	0.096	0.067
Standard Deviation	0.00047	0.0038	0.0072	0.0024

Table 87 : Comparison of neural network outputs after adjusting the screws back and forward (Each adjustment equals two turns)

	Neural network output for...			
Position	C_4 anti-clockwise	C_4 clockwise	C_7 anti-clockwise	C_7 clockwise
tuned	0.052	0.026	0.0	0.0
C_4 anti-clockwise	0.830	0.072	0.532	0.097
C_4 clockwise	0.262	0.0	0.003	0.0
Difference:	0.210	0.026	0.003	0.0
tuned	0.262	0.0	0.003	0.0
C_4 anti-clockwise	1.596	0.078	0.706	0.452
C_4 clockwise	0.136	0.0	0.0	0.0
Difference:	0.126	0.0	0.003	0.0
Mean of Differences	0.168	0.013	0.003	0.0

■ The stopband was already tuned. Therefore C_4 and C_7 remained at the positions where found.

Table 88 shows the total number of attempts made together with the number of successful and unsuccessful tunings. The system managed to tune the passband region in all cases. Other observations made were:

■ For the first tuning step the system always advised a T component in an anti-clockwise direction.

■ The system advised T components at the beginning of the tuning (on average for two to three steps) and then, if necessary, C components. A situation where, for example, a T component was used, followed by a C component and then a T component was re-used did not arise. This resembles how the human operator proceeds.

Average number of adjustments

Table 89 shows the average number of adjustments required for the tuning of the passband. The minimum and maximum number of adjustments are also recorded. The equivalent figures for the performance of the human operator are given as well. In some filter tuning attempts the human seems to require one more step but on average a slight decrease of tuning steps required does appear to have been achieved.

Examination of tuning steps by the empty rule category

Table 90 shows the number of steps for each action advised by the system. Table 91 displays, in more detail, the *empty* rules that were executed together with the action taken by the operator each time. Again, only the first time of an *empty* rule appearance is counted. It is noticeable that the operator always took the same action and that most *empty* rules were substituted with a C

Table 88 : Number of attempts for the tuning of the passband using case 1 system	
Number of attempts:	26
Successful tunings:	26
Unsuccessful tunings:	0

‡ Case 1 designates the knowledge-based plus human system

Table 89 : Average number of turns for the tuning of the passband using case 1 system		
	Human	Case 1
Average number of turns:	4.03	4.3
Minimum number of turns:	2	2
Maximum number of turns:	9	8

Table 90 : Number of occurrences per action using case 1 system (passband)		
	Number of occurrences	Number of occurrences (%)
Empty	14	12.4
Clash	23	20.4
Component and Direction	76	67.2

‡ Case 1 designates the knowledge-based plus human system

Table 91 : Number of rules executed at situations where the case 1 system could not provide an advice for the tuning of the passband

Rule number	Proposed action
165	T_3 anti-clockwise
71	C_7 clockwise × 2
130	C_4 anti-clockwise
203	T_3 anti-clockwise × 3
81	T_3 anti-clockwise
66	C_4 anti-clockwise
84	C_4 anti-clockwise × 2
89	C_4 anti-clockwise
75	C_4 anti-clockwise
91	C_7 clockwise

‡ Case 1 designates the knowledge-based plus human system
‡ The character 'x' as in C_4 clockwise x 2 denotes multiple occurrences

component. This was expected since only 38.8% of the training examples had a C component class.

Examination of tuning steps by the clash rule category

Table 92 shows the *clash* rules that were executed together with the action taken each time by the operator during the testing and the training. For the majority of the cases clashes also occurred during the testing. One noticeable exception is rule 77 which appeared five times and the action taken was identical for all of them.

Examination of tuning steps by component and direction category

Table 93 shows the distribution per rule of the 76 situations where the system provided an advice. Rules are in ascending order. Table 94 shows the total number of rules, their occurrences and the available number of rules in the rule base for each combination. The following observations can be made:

- Only 29.2% of the available rules were used with the C component rules having the smaller percentage.
- The vast majority of the executed rules had an anti-clockwise direction part.
- The component T_1 was rarely used.
- The probable frequency of a particular combination arising is independent of the number of available rules in the rule set.

Table 92 : Number of rules providing conflicting advice for the tuning of the passband using case 1 system

Rule number	Testing	Training
163	T_2 clockwise	C_7 clockwise
	T_2 anti-clockwise	T_2 anti-clockwise
129	C_7 clockwise × 3	C_7 clockwise × 2
	T_2 anti-clockwise	T_3 clockwise
51	C_7 anti-clockwise × 2	T_2 anti-clockwise
	C_4 anti-clockwise	C_4 anti-clockwise
68	C_7 anti-clockwise × 2	C_7 anti-clockwise
		C_7 clockwise
		C_4 anti-clockwise
176	T_2 anti-clockwise × 2	T_2 anti-clockwise
	C_7 clockwise	C_7 anti-clockwise
77	C_4 anti-clockwise × 5	C_4 anti-clockwise
		T_2 clockwise × 2
		T_2 anti-clockwise

‡ Case 1 designates the knowledge-based plus human system
‡ The character 'x' as in C_4 clockwise x 2 denotes multiple occurrences

Table 92 continued : Number of rules providing conflicting advice for the tuning of the passband using case 1 system

Rule number	Testing	Training
102	C_4 clockwise	C_4 clockwise
	C_7 anti-clockwise	C_7 anti-clockwise x 2
		T_3 anti-clockwise
		T_2 clockwise
	C_4 anti-clockwise x 2	C_4 anti-clockwise
		C_7 anti-clockwise

‡ Case 1 designates the knowledge-based plus human system
‡ The character 'x' as in C_4 clockwise x 2 denotes multiple occurrences

Table 93: Distribution per rule for the tuning of the passband using case 1 system

Rule number	Number of occurrences	Combination
11	2	C_7 anti-clockwise
44	1	T_3 clockwise
45	1	C_4 anti-clockwise
47	2	C_4 anti-clockwise
52	5	C_4 anti-clockwise
58	2	C_7 anti-clockwise
90	6	C_4 anti-clockwise
112	1	T_2 clockwise
117	2	C_7 clockwise
118	1	T_3 clockwise
119	1	T_2 anti-clockwise
136	2	T_1 anti-clockwise
151	2	T_2 anti-clockwise
156	2	T_2 anti-clockwise
164	11	T_2 anti-clockwise

‡ Case 1 designates the knowledge-based plus human system

Table 93 continued : Distribution per rule for the tuning of the passband using case 1 system

Rule number	Number of occurrences	Combination
177	4	T_2 anti-clockwise
179	3	T_2 anti-clockwise
194	3	T_3 anti-clockwise
195	3	T_3 anti-clockwise
197	1	T_3 anti-clockwise
198	18	T_3 anti-clockwise

‡ Case 1 designates the knowledge-based plus human system

Table 94 : Distribution of rules per combination for the tuning of the passband using case 1 system

Combin-ation	No. of rules	Occur-rences	No. of available rules	No. of occur-rences (%)	No. of available rules (%)
T_1 a	1	2	3	2.7	4.2
T_1 c	0	0	0	0.0	0.0
T_2 a	6	26	02	60.5	16.7
T_2 c	1	1	5	1.4	7.0
T_3 a	4	25	14	34.2	19.2
T_3 c	2	2	5	2.7	7.0
C_4 a	4	14	15	19.2	20.8
C_4 c	0	0	4	0.0	5.6
C_7 a	2	4	9	5.6	12.5
C_7 c	1	2	5	2.7	7.0

‡ The character 'c' as in T_3 c denotes the clockwise direction
‡ The character 'a' as in T_3 a denotes the anti-clockwise direction
‡ Case 1 designates the knowledge-based plus human system

10.12 Summary of results

Using the expert system-neural network system combination, the stopband regions of 79% of the filters were successfully tuned with typically 3 adjustments needed. The passband regions of all filters were successfully tuned with typically 4 adjustments needed. The results are given separately for the two regions because for the passband region how far to turn was provided by the operator. Typically 80% of the filters were tuned completely (i.e., both stopband and passband) requiring on average 7 adjustments.

The results presented above demonstrate that no one system out-performed the rest considering all the performance criteria stated in Section 10.4. Comparing the three systems with the results obtained solely with a human operator one can deduce that there was not an increase in the efficiency with which the tuning was produced. The number of required adjustments, except for when the neural network were used on their own, remained about the same but the argument is that they can be used by non-experts viz. reducing the training time.

The employment of the expert system displays similar dynamics to the human operator including shortcomings such as the *empty* or *clash* situations. Neural networks on the other hand do not suffer from these shortcomings. An answer, not necessarily the correct one, is always given. Since in the filter tuning application checking the neural network predictions was not always possible then this advantage of neural networks is questionable.

The selection of the *best* system cannot depend only on the accuracy and speed of tuning but must also depend on the speed of training of the classifiers and the amount of effort that goes into their training, their ease

of use and implementation.

10.13 Conclusions

The evaluation of the performance of the three approaches for the tuning of the stopband region using *on-line* data showed that the hybrid system provided good results with an increase in the efficiency with which a solution was produced.

After using the neural networks as stand-alone systems with *on-line* data demonstrated that the actions given by the networks with the largest prediction values should be followed. When *de-tune* data was employed none of the networks which recognize a tuned position or the clockwise adjustment gave an incorrect result that was off by more than 0.1 unit. On the other hand results for the anti-clockwise adjustment were not as well-defined usually giving correct predictions up to a level and then under-estimating. The experiments which were performed in order to identify the probable source of the 0.1 error showed that the error arose mainly due to the screw backlash.

On the other hand the tuning of the passband region was performed successfully in all cases but the operator was required to provide the distance to turn.

References

1. Tsaptsinos D., Jervis B.W., and Mirzai A.R., *Practical aspects of using an expert system-neural network hybrid system for tuning crystal filters*, IEE Second International Conference on Artificial Neural Networks, pp. 314-317, 1991.

Chapter Eleven

General Observations and Conclusions

11.2 General observations

The knowledge acquisition bottleneck is a huge obstacle to the development of expert systems. This was also proven in the task of tuning electronic filters. The hybrid system AEK, as described in this thesis, uses knowledge in rule form for determining which component to turn and in which direction and in weight form for determining how far to turn. Rules were induced automatically using the ID3 learning algorithm and weights were derived after *teaching* feed-forward neural networks using the back-propagation learning algorithm. Sets of training examples representing past experience were at the disposal of the algorithms. The training examples presented to the neural networks were sampled values of the raw magnitude response while for ID3 the examples consisted of representations of the raw responses in term of waveform peaks. During the implementation of the two learning algorithms a number of issues were observed and they are summarised below in order to serve other researchers in the field.

- When using ID3 with attributes which take numerical rather than categorical values it is strongly advisable to introduce some inexact modelling by creating ranges of values and introducing linguistic values to label them. The reason for that is that the algorithm comprehends two numerical values like 1.75 and 1.76 as two different concepts when most of the time a human will consider both values as equal. The cut-off points of the decision tree are very dependent on the examples present in the training set whereas this dependency is greatly reduced with the introduction of ranges. Therefore, when the following are true:

(i) A complete set of examples is not available, and

(ii) the present set is somehow representative, and

(iii) it is expected to introduce new examples as the time passes on

then by employing attributes with linguistic variables one stabilises the appearance of the decision tree. This was observed during the comparisons of decisions trees as described in Chapter 6 and it held true especially for the top part of the trees.

■ The algorithms depend on the examples of the training set and whereas the question of training set size is important one must also pay attention to the available examples as well. A statistical detection of peculiar data is therefore useful and essential since their use will cause poor results. This can involve the discovery of data corrupted by noise and spikes or outliers using basic statistical concepts (e.g. the mean) and graphical representations. For example, an attribute value three standard deviations away from the mean is a candidate for further exploration of its validity. The drawing of scattergrams (see Chapter 8) was a simple method of looking at the data and demonstrated its usefulness by discovering the overlapping of classes.

■ ID3 assists but is fallible. One must always examine the generated decision tree. This due to the irrelevant branching problem as discussed in Chapter 7 which causes the algorithm to branch out for situations which can never hold.

■ For applications where data interpretation is desirable the output of ID3 is much more useful. A decision tree is more useful for understanding the structure of the data than the output of a neural network which has a much less clear (approaching zero) usefulness for interpretation.

■ Experiments for determining the topology of the networks showed that the

influence of the manner of representation of the data was critical and that scaling of the inputs and outputs was beneficial. In order to avoid confusion to the users and because response time in the filter tuning domain is important a simple procedure which required modest computation was followed. In fact, most time was spent in the design of the means for representing the data for both algorithms rather than the actual learning.

- The use of commercially available packages (Xi-Plus, Xi-Rule, and NeuralWorks Explorer) for the research period permits some assessment. The assessments that follow concern only the stated versions of the software packages.

Xi-Plus provides a good user interface and a good diagrammatic representation of conclusions. The drawbacks of the shell are:

(i) The use of a lot of memory which can cause the crash of the system.

(ii) The limit of the number of rules per knowledge base. This and the previous drawback can be avoided by creating an application layer with a number of knowledge bases calling each other. This was followed in AEK but revealed the following shortcoming.

(iii) When the execution of one knowledge-base terminates and another knowledge-base is called the conclusions of the previous knowledge-base can be kept but the explanations are missing. The explanation provided by the system is a single line stating that the conclusion was inferred during the execution of the previous knowledge-base.

(iv) The shell is lacking in the provision of mathematical functions and graphical facilities.

(v) The shell can be interfaced with other software but it is rather

complicated. Using the Lotus interface with AEK revealed that one can read one or more lines of the spreadsheet and assign the values of each spreadsheet cell to variables but if one desires to do a sequential reading this is impossible.

(vi) The assignment of certainty factors to the conclusion of the rules is not possible. Therefore, when one faces a situation where more than one rule holds it is not feasible for the system to perform the conflict resolution.

(vii) The ordering of the rules is important thus one has to be careful where the rules are located.

(viii) Xi-Plus is menu-based but a mouse driver is not available therefore a lot of keyboard usage cannot be avoided during execution.

(ix) Xi-Plus lacks the ability to represent knowledge acquired in the most suitable representation format since it provides only the production rule format.

Xi-Rule is based on the ID3 algorithm and creates decision trees easily and quickly. The user has only to provide the names of the attributes, to decide on the attribute values and either to type the examples or to read them in from an ASCII file. It does not provide a windowing facility or any pruning facilities. The decision tree can be easily sent to Xi-Plus transforming each leaf to a production rule. This is not possible though for leaves assigned an empty or a clash class and had to be done by hand.

NeuralWorks Explorer can generate a number of standard network types from an extensive library with more than enough summation, error, output and transfer functions. The use of the package is painless but the learning process will benefit in future with the introduction of graphical representation

of the weights, the output etc. as learning progresses.

11.3 Future work

The AEK system incorporating knowledge-bases and neural networks is proposed for now. At its present state the system still has one problem, namely that of the third search. This is particularly true for the tuning of the passband region where at the moment how far to turn is determined by the user. For eventual use in industry future work must concentrate on the following:

(i) Automation of the instrumentation by using for an example a robotic arm. This will be useful during the gathering of *de-tune* data for use by the neural networks.

(ii) Transferring the knowledge-bases to the Hewlett-Packard computer. This will increase the speed of execution of the rules since an interface between the measuring equipment and the Hewlett-Packard computer can be easily constructed. This has the disadvantage of being difficult to maintain, change and inspect the knowledge-bases.

(iii) Algorithms must be investigated and programs must be written in order to locate the peaks automatically.

All the above suggestions will contribute towards a fully-automatic system. Additionally, further work can be done in order to improve the performance of the neural networks. For example some form of pre-processing, such as filtering of the magnitude response using Fourier transforms, might improve the results.

11.4 Conclusions

This thesis presented the work undertaken in order to create a prototype expert system which is hoped to be of benefit to the industry. The system is taking the role of an advisor during the tuning of crystal filters.

Two main conclusions were drawn concerning the use of ID3.

(i) When using ID3 with attributes which take numerical rather than categorical values it is strongly advisable to introduce some inexact modelling by creating ranges of values and introducing linguistic values to label them.

(ii) It was not possible to create rules to predict how far to turn (i.e., search 3) using ID3 because of the vast number of attributes the class attribute can take. Information would have been lost if linguistic values were introduced. Therefore an alternative method was needed. Multi-layer neural networks learning using the back-propagation algorithm were constructed for the third search of the stopband sub-task. Hence, a hybrid expert system-neural network system was formed.

The testing and evaluation of the performance of three systems (see Chapter 10) showed that:

(i) For case 1 (i.e., the expert system-operator system combination), the stopband regions of 86% of the filters were successfully tuned with typically 3 adjustments needed. The passband regions of all filters were successfully tuned with typically 4 adjustments needed.

(ii) For case 2 (i.e., the expert system-neural network system combination), the stopband regions of 79% of the filters were successfully tuned with typically 3 adjustments needed.

(iii) For case 3 (i.e., the neural network system), the stopband regions of 67%

of the filters were successfully tuned with typically 10 adjustments needed. The results are given separately for the two regions because for the passband region how far to turn was provided by the operator. Typically 80% of the filters were tuned completely (i.e., both stopband and passband) requiring in average 7 adjustments.

The production of the hybrid system (i.e., case 2) indicated that the choice of which component to use and in which direction to turn can be easily determined. How far to turn presents a more complex problem.

The second system is recommended for use now with the third system being the most promising for the future. The results presented in the thesis concern only one type of crystal filter but the generic methodology can be applied to other types with little problem.

Appendix One

Transcription of video-tape for filter 4716

Taped at Newmarket 20..22 June 1988

These are the end coils they are just straight inductors effectively, in the center there is a center-tap coil which acts as the bridge circuit in the middle there. There are two trimmer capacitors which are used to adjust the stopband. The stopband is an asymetric one. It is an unusual shape. It is made this way to provide a single side band type of performance from what it is normally just a four-pole elliptic.

Stick it and we can have a go. Set the frequency to 1.4MHz coarse frequency that is. Adjust the display on log scale to give me a whole responce of the filter. Looking at, let me check that is locked on. Looking at the responce on the screen I want to increase the amount I am seen so I am reducing the sweep-width down to about 3.5 KHz, that is +-3.5 and then just centring up using the fine frequency control.

Now we see the basic untuned filter. If I can get my head out of the way. Right now the first thing I am gonna do is to adjust the trimmer capacitors to arrange these peaks into a more reasonable place because I once have done that I know the stray balancing capacitor in the circuit is more or less right and it is not going to effect the passband responce too much later on. It does not matter which end you do first. Turn the trimmer capacitor (R) anticlockwise and pull the peaks out on the other side. That is done one. Now move to the other end (L) turning anticlockwise again and pull that out just off the screen.

Increase the sweep-width to +-7KHz and see the two peaks have been moved off to one side and we are looking more like a decent asymetric single side band responce where the carrier frequency is against this edge.

Now ready to begin and try to tune the end coils to flatten the ripple. Again it doesn't matter which end you start from. I'll go left hand end and turn clockwise and see what happens and the ripple drops. Keep on turning, it is getting worst again I'll stop there. Go to right hand end turn the slug clockwise. Watch the ripple drop. So we are getting close to the theoretical shape now. Change now to a linear display about there and increase the bandwidth, reduce the bandwidth rather to something I cannot actually read 1.75, I think. Centre up the display. Right now I'm ready to tune the centre coil and I'll need to go and get a tuner.

It should bring down the lump on the upper side, we hope. I'm turning clockwise and that is going too far so I come back again anticlockwise again and that is going too far. So about there it looks about right. Now give the end coils another adjust just to make sure they are where they ought to be. If I screw in they get worst if I turn anticlockwise they get better. Turn this one anticlockwise as well that is too bad. So I go back again clockwise. Just put a bit of ripple it doesn't want to be too flat. The flatter you make your passband the less stopband attenuation you get so you want to put a little bit of ripple just to help you along. Right now what we'll do.

I think at this stage i will actually measure the passband to see what we've got, so I'll stop the sweep. Find the maximum power transmission, lowest point of passband which is there. Set the needle to zero with the generator output level. I can now measure the 4db bandwidth on the upper side it is, change that quickly, is 2.7 on the spec is 2.55 we are ok there. Go back to the other side 4db on the meter gives us +4.34 and we are looking for less than 4.5. That is ok, a bit tight but it is ok.
Um, right if I resume sweeping again and change the level to 40 (10,20, 30, 40) increase the sweepwidth, center the passband again, what I'll try to do now is to adjust the peaks so these return levels come to about, what we've got, about 48db. Check and see where they are at the moment. That one is at 53 and that one is about 57. If I sweep there and then just try to adjust the trimmer capacitors that effect those peaks in. ie. drop the return levels down. This is the LH one turning clockwise looks what I want to do it is coming down, probably too much now. I'll check the level again. That is 50 that is 50. I'll say that is probably about right.

Right, I can go back now to the passband and measure the whole filter right through make sure we are in spec still. So it is to the lowest point of the passband. Set to zero the meter. Measure 4db which is 2.6 and we are looking for 2.55 that is ok. Set to the other side that is 0.43 we are looking for 4.5, less than 4.5 that is ok, 0.45 beg your pardon, that is ok. The next thing is the where are we, carrier rejection I think yes if I set the frequency exactly on 1.4 MHz ie. zero on the counter as it stands at the moment, about there somewhere, the attenuation should be better than 15 so that is 10 we've got 24 so that is fine. Right now we are up to 20 on that scale. I'm gonna increase that to 40 and check the 45 db points. I think we'll have to change the bandwidth. Now there is this time 10's multiplier that comes on certain ranges of the bandwidth so now switched in the times 10's.

I'll have to readjust the fine frequency to put the filter back in the middle of the screen, that is, stop the sweep go to find the 45db points. That is the low one there which is -4.52 and we are looking for something better than 600, so that is okay. This side is +3.9 and we want it less than +4.8 that is ok too. So just a quick sweep there just to check the return levels. That is 50db, that is 50db.

LH capacitor trimmer effects the closest upper stopband peak, this will effect the upper 45db point more than anything else it also effects the return level but that can be compensated later.

On the LH trimmer both outer peaks on each side move and this will tend to leave the 45db frequencies where they are but they will effect the return levels.

So if you want to adjust the return levels your best bet is with the RH trimmer - if you want to adjust the upper 45db point your best bet is with the LH trimmer.

Appendix Two

Instructions given to the operator

Dear Sir,

We are making a study of the filter tuning process. We believe you are especially well qualified to tell us about the process as a whole, but at the present we would like you to concentrate only on the following task. You are just about ready to start tuning a filter. During the process you probably make some decisions based on *something*.

Those *something* are what we would like you to write down on a piece of paper as they occur. Please use your own terminology.

For example, consider the situation where you try to decide if an umbrella will be required to be taken to work. The decision might be made by just looking at the sky. Well, the sky is the *something*. Note that they do not have to be single words.

One thing we would like you to know is that there are no right or wrong answers. Different people judge things in different ways. We are interested in your results as an individual.

Please take as much time as you require.

Yours faithfully

Dimitris Tsaptsinos

Appendix Three

Transcription of video-tape for filter 4716

Taped at Newmarket 20..22 June 1988

These are the end coils
they are just straight inductors effectively,
in the center there is a center-tap coil
which acts as the bridge circuit in the middle there.
There are two trimmer capacitors
which are used to adjust the stopband.
The stopband is an asymetric one.
It is an unusual shape.
It is made this way to provide a single side band type of
performance from what it is normally just a four-pole elliptic.
Stick it and we can have a go.
Set the frequency to 1.4MHz coarse frequency that is.
Adjust the display on log scale
to give me a whole responce of the filter.
Looking at,let me check that is locked on.
Looking at the responce on the screen I want to
increase the amount I am seen
so I am reducing the sweep-width down to about 3.5 KHz,that is
+-3.5
and then just centring up
using the fine frequency control.
Now we see the basic untuned filter.
If I can get my head out of the way.
Right now the first thing I am gonna do is to
adjust the trimmer capacitors
to arrange these peaks into a more reasonable place
because I once have done that
I know the stray balancing capacitor in the circuit is more or
less right and it is not going to effect the passband responce
too much later on.
It does not matter which end you do first.
Turn the trimmer capacitor (R) anticlockwise
and pull the peaks out on the other side.
That is done one.
Now move to the other end (L) turning anticlockwise again
and pull that out just off the screen.
Increase the sweep-width to +-7KHz and
see the two peaks have been moved off to one side and
we are looking more like a decent asymetric single side band
responce where the carrier frequency is against this edge.
Now ready to begin and
try to tune the end coils
to flatten the ripple.
Again it doesn't matter which end you start from.
I'll go left hand end and turn clockwise and
see what happens and

the ripple drops.
Keep on turning,
it is getting worst again
i'll stop there.
Go to right hand end turn the slug clockwise.
Watch the ripple drop.
So we are getting close to the theoretical shape now.
Change now to a linear display about there
and increase the bandwidth,reduce the bandwidth rather to
something I cannot actually read 1.75 I think.
Centre up the display.
Right now i'm ready to tune the centre coil
and i'll need to go and get a tuner.
It should bring down the lump on the upper side,
we hope.
I'm turning clockwise
and that is going too far
so I come back again anticlockwise again
and that is going too far.
So about there it looks about right.
Now give the end coils another adjust
just to make sure they are where they ought to be.
If I screw in they get worst
if I turn anticlockwise they get better.
Turn this one anticlockwise as well
that is too bad.
So I go back again clockwise.
Just put a bit of ripple it doesn't want to be too flat.
The flatter you make your passband the less stopband attenuation you get
so you want to put a little bit of ripple just to help you along.
Right now what we'll do.
I think at this stage
I will actually measure the passband
to see what we've got,
so i'll stop the sweep.
Find the maximum power transmission,
lowest point of passband which is there.
Set the needle to zero
with the generator output level.
I can now measure the 4db bandwidth on the upper side it is,
change that quickly,
is 2.7 on the spec is 2.55
we are ok there.
Go back to the other side
4db on the meter gives us +4.34
and we are looking for less than 4.5.

That is ok, a bit tight but it is ok.
Um,right
if I resume sweeping again and
change the level to 40 (10,20,30,40)
increase the sweepwidth,
center the passband again,
what i'll try to do now is to
adjust the peaks
so these return levels come to about,what we've got,about 48db.
Check and see where they are at the moment.
That one is at 53 and that one is about 57.
If I sweep there and
then just try to adjust the trimmer capacitors
that effect those peaks in.
ie. drop the return levels down.
This is the LH one turning clockwise
looks what I want to do it is coming down,
probably too much now.
I'll check the level again.
That is 50 that is 50.
I'll say that is probably about right.
Right,I can go back now to the passband and
measure the whole filter right through
make sure we are in spec still.
So it is to the lowest point of the passband.
Set to zero the meter.
Measure 4db which is 2.6 and we are looking for 2.55
that is ok.
Set to the other side that is 0.43 we are looking for 4.5,less
than 4.5
that is ok,0.45 beg your pardon,that is ok.
The next thing is the where are we,carrier rejection
I think yes
if I set the frequency exactly on 1.4 MHz
ie. zero on the counter as it stands at the moment,
about there somewhere,
the attenuation should be better than 15 so that is 10 we've got
24
so that is fine.
Right now we are up to 20 on that scale.
I'm gonna increase that to 40 and
check the 45 db points.
I think we'll have to change the bandwidth.
Now there is this time 10's multiplier that comes on certain
ranges of the bandwidth
so now switched in the times 10's.

I'll have to readjust the fine frequency
to put the filter back in the middle of the screen,that is,
stop the sweep
go to find the 45db points.
That is the low one there which is -4.52 and we are looking for something better than 600,
so that is okay.
This side is +3.9 and we want it less than +4.8
that is ok too.
So just a quick sweep there
just to check the return levels.
That is 50db, that is 50db.
LH capacitor trimmer
effects the closest upper stopband peak,
this will effect the upper 45db point more than anything else
it also effects the return level but
that can be compensated later.
On the LH trimmer
both outer peaks on each side move and
this will tend to leave the 45db frequencies where they are but
they will effect the return levels.
So if you want to adjust the return levels
your best bet is with the RH trimmer -
if you want to adjust the upper 45db point
your best bet is with the LH trimmer.

Attenuation: A general term used to denote a decrease in signal magnitude in transmission from one point to another. May be expressed as a scalar ratio of the input magnitude to the output magnitude or in decibels.

Active filter: A filter network containing one or more active devices (usually an operational amplifier) in addition to passive elements (resistors, capacitors).

Coil: One or more loops of wire wound spirally, often around a cylindrical cardboard or iron core, and exhibiting the property of inductance. Also called an inductor.

Passband: A band of frequencies that pass through a filter with little loss.

Ripple: The variations on a frequency plot of an impedance function or of a transfer function.

Appendix Five

Listing of knowledge-base of the stopband (search 1)

```
fact     d1 = peak2 - peak1
fact     d2 = peak3 - peak1
fact     d3 = peak4 - peak1
fact     d4 = peak3 - peak2
fact     d5 = peak4 - peak2
fact     d6 = peak4 - peak3
fact     d7 = level2 - level1
question   peak4 =
   1 to 100 , unknown
  question text what is the value of the fourth peak ?
          and ( if r2 exists give the max on the right value )
question   peak1 =
   1 to 100 , unknown
  question text what is the value of the first peak ?
question   peak3 =
   1 to 100 , unknown
  question text what is the value of the third peak ?
question   peak2 =
   1 to 100 , unknown
  question text what is the value of the second peak ?
question   level1 =
   1 to 100 , unknown
  question text what is the value of the first return level ?
question   level2 =
   1 to 100 , unknown
  question text what is the value of the second return level ?
   when outcome is carry - on
   then command reset p1
    and command reset p2
    and command reset p3
    and command reset p4
    and command reset r1
    and command reset r2
    and command reset d1
    and command reset d2
    and command reset d3
    and command reset d4
    and command reset d5
    and command reset d6
    and command reset d7
    and command load c:\newmarket\search2
   when peak4 = unknown
   then force outcome is carry - on
   when peak1 = unknown
   then force outcome is carry - on
   when peak2 = unknown
   then force outcome is carry - on
   when peak3 = unknown
   then force outcome is carry - on
   when level1 = unknown
```

```
then force outcome is carry - on
when level2 = unknown
then force outcome is carry - on
when outcome is end
then command load a:\numbers
if   diff3 is left
then outcome is carry - on
 and report EXAMPLES USED 34 / 138 ( 24.61% )
 and confidence = 98.5
if   diff3 is right
 and p2 is left
then outcome is empty
if   p2 is right
then outcome is carry - on
 and report EXAMPLES USED 3 / 138 ( 2.17% )
 and confidence = 91.3
if   diff3 is right
 and p2 is ok
then outcome is end
 and report EXAMPLES USED 4 / 138 ( 2.9% )
 and report END - OF - PROCESS
 and confidence = 87.5
if   diff3 is ok
 and p3 is left
 and p2 is left
then outcome is empty
if   p1 is left
then outcome is carry - on
 and report EXAMPLES USED 2 / 138 ( 1.45% )
 and confidence = 75
if   diff3 is ok
 and p3 is left
 and p2 is right
 and diff1 is right
 and p1 is right
then outcome is empty
if   p3 is left
 and p2 is right
then outcome is carry - on
 and report EXAMPLES USED 6 / 138 ( 4.35% )
 and confidence = 97.2
if   p3 is left
then outcome is carry - on
 and report EXAMPLES USED 1 / 138 ( 0.72% )
 and confidence = 92.4
if   diff3 is ok
 and p3 is left
 and p2 is ok
 and diff1 is right
then outcome is empty
```

```
if   diff3 is ok
 and p3 is left
 and p2 is ok
 and diff1 is ok
 and p1 is left
then outcome is empty
if   diff3 is ok
 and p3 is left
 and p2 is ok
 and diff1 is ok
 and p1 is ok
 and diff7 is left
then outcome is empty
if   diff3 is ok
 and p3 is right
 and diff1 is left
then outcome is empty
if   p3 is right
 and diff1 is right
then outcome is carry - on
 and report EXAMPLES USED 2 / 138 ( 1.45% )
 and confidence = 87.5
if   p3 is right
 and diff1 is ok
then outcome is end
 and report EXAMPLES USED 5 / 138 ( 3.62% )
 and report END - OF - PROCESS
 and confidence = 91.7
if   diff3 is ok
 and p3 is ok
 and diff1 is left
then outcome is end
 and report EXAMPLES USED 4 / 138 ( 2.90% )
 and report END - OF - PROCESS
 and confidence = 87.5
if   diff3 is ok
 and p3 is ok
 and diff1 is right
 and r2 is right
then outcome is empty
if   diff3 is ok
 and p3 is ok
 and diff1 is right
 and r2 is ok
 and diff4 is right
then outcome is empty
if   diff1 is right
then outcome is carry - on
 and report EXAMPLES USED 2 / 138 ( 1.45% )
 and confidence = 75
```

```
if   diff3 is ok
 and p3 is ok
 and diff1 is ok
 and r2 is left
then outcome is empty
if   diff3 is ok
 and p3 is ok
 and diff1 is ok
 and r2 is right
 and p2 is left
then outcome is empty
if   diff7 is right
then outcome is carry - on
 and report EXAMPLES USED 2 / 138 ( 1.45% )
 and confidence = 86.5
if   diff3 is ok
 and p3 is ok
 and diff1 is ok
 and r2 is right
 and p2 is right
 and diff7 is left
then outcome is empty
if   diff3 is ok
 and p3 is ok
 and diff1 is ok
 and r2 is ok
 and p2 is left
then outcome is empty
if   p3 is ok
 and r2 is ok
then outcome is end
 and report EXAMPLES USED 2 / 138 ( 1.45% )
 and report END - OF - PROCESS
 and confidence = 78.8
if   diff3 is ok
 and p3 is ok
 and diff1 is ok
 and r2 is ok
 and p2 is ok
 and r1 is right
then outcome is empty
if   r2 is ok
 and diff7 is ok
then outcome is end
 and report EXAMPLES USED 12 / 138 ( 8.70% )
 and report END - OF - PROCESS
 and confidence = 68.2
if   diff3 is ok
 and p3 is ok
 and diff1 is ok
```

```
 and r2 is ok
 and p2 is ok
 and r1 is ok
 and diff7 is left
then outcome is empty
if   p1 is ok
then outcome is end
 and report EXAMPLES USED 1 / 138 ( 0.72 % )
 and report END - OF - PROCESS
 and confidence = 0.60
if   p2 is ok
then outcome is end
 and report EXAMPLES USED 1 / 138 ( 0.72 % )
 and report END - OF - PROCESS
 and confidence = 0.54
if   diff7 is ok
then outcome is end
 and report EXAMPLES USED 1 / 138 ( 0.72 % )
 and report END - OF - PROCESS
 and confidence = 0.56
if   peak1 >= 1.397156
 and peak1 <= 1.398220
then p1 is ok
if   peak1 < 1.397156
then p1 is left
if   peak1 > 1.398220
then p1 is right
if   d3 >= 0.008236
 and d3 <= 0.032018
then diff3 is ok
if   d3 < 0.008236
then diff3 is left
if   d3 > 0.032018
then diff3 is right
if   peak2 >= 1.399327
 and peak2 <= 1.399546
then p2 is ok
if   peak2 < 1.399327
then p2 is left
if   peak2 > 1.399546
then p2 is right
if   peak3 >= 1.404048
 and peak3 <= 1.405241
then p3 is ok
if   peak3 < 1.404048
then p3 is left
if   peak3 > 1.405241
then p3 is right
if   peak4 >= 1.406214
 and peak4 <= 1.429488
```

```
then p4 is ok
if   peak4 < 1.406214
then p4 is left
if   peak4 > 1.429488
then p4 is right
if   level1 >= 56
 and level1 <= 64
then r1 is ok
if   level1 < 56
then r1 is left
if   level1 > 64
then r1 is right
if   level2 >= 58
 and level2 <= 65
then r2 is ok
if   level2 < 58
then r2 is left
if   level2 > 65
then r2 is right
if   d1 >= 0.001188
 and d1 <= 0.002309
then diff1 is ok
if   d1 < 0.001188
then diff1 is left
if   d1 > 0.002309
then diff1 is right
if   d2 >= 0.006105
 and d2 <= 0.007807
then diff2 is ok
if   d2 < 0.006105
then diff2 is left
if   d2 > 0.007807
then diff2 is right
if   d4 >= 0.004522
 and d4 <= 0.005894
then diff4 is ok
if   d4 < 0.004522
then diff4 is left
if   d4 > 0.005894
then diff4 is right
if   d5 >= 0.006767
 and d5 <= 0.030062
then diff5 is ok
if   d5 < 0.006767
then diff5 is left
if   d5 > 0.030062
then diff5 is right
if   d6 >= 0.001616
 and d6 <= 0.024798
then diff6 is ok
```

```
  if   d6 < 0.001616
  then diff6 is left
  if   d6 > 0.024798
  then diff6 is right
  if   d7 >= 0
   and d7 <= 6
  then diff7 is ok
  if   d7 < 0
  then diff7 is left
  if   d7 > 6
  then diff7 is right
  if   level1 > level2
  then d7 = level1 - level2
query   outcome is
 query options auto
```

Appendix Six

Listing of knowledge-base of the stopband (search 2)

```
fact     lim1 = 0.0001
fact     lim2 = 0.001
question   peak1 =
    1 to 100 , unknown
  question text enter the value of the first peak
question   peak2 =
    1 to 100 , unknown
  question text enter the value of the second peak
question   peak3 =
    1 to 100 , unknown
  question text enter the value of the third peak
question   peak4 =
    1 to 100 , unknown
  question text enter the value of the fourth peak
question   level1 =
    1 to 100 , unknown
  question text enter the value of the first return level
question   level2 =
    1 to 100 , unknown
  question text enter the value of the second return level
question   observe is
    yes ,
    no
  question text Would you like to see the results ?
   when componen is X
   then report Component to be used is [X]
    and command reset peak1
    and command reset peak2
    and command reset peak3
    and command reset peak4
    and command reset level1
    and command reset level2
    and command reset p1
    and command reset p2
    and command reset p3
    and command reset p4
    and command reset r1
    and command reset diff1
    and command reset diff2
    and command reset diff5
    and command reset diff3
    and command reset diff4
    and command reset diff6
    and command reset diff7
    and command reset r2
    and command reset outcome
    and command reset componen
    and command load c:\newmarket\search1
   when peak1 = unknown
   then force p1 is absent
```

```
when peak2 = unknown
then force p2 is absent
when peak3 = unknown
then force p3 is absent
when peak4 = unknown
then force p4 is absent
when level1 = unknown
then force r1 is absent
when level2 = unknown
then force r2 is absent
if  p1 is left
 and p2 is left
then componen is empty
if  p1 is left
 and p2 is right
then componen is c4a
if  p1 is left
 and p2 is ok
 and r1 is left
then componen is empty
if  p1 is left
 and p2 is ok
 and r1 is right
then componen is empty
if  p1 is left
 and p2 is ok
 and r1 is ok
then componen is c4c
if  p1 is left
 and p2 is ok
 and r1 is absent
then componen is empty
if  p1 is left
 and p2 is ok
 and r1 is fleft
then componen is empty
if  p1 is left
 and p2 is ok
 and r1 is fright
then componen is empty
if  p1 is left
 and p2 is ok
 and r1 is cleft
then componen is c7c
if  p1 is left
 and p2 is ok
 and r1 is cright
then componen is empty
if  p1 is left
 and p2 is absent
```

then componen is empty
if p1 is left
and p2 is fleft
then componen is empty
if p1 is left
and p2 is fright
then componen is empty
if p1 is left
and p2 is cleft
then componen is c7c
if p1 is left
and p2 is cright
then componen is c4c
if p1 is right
and p2 is left
then componen is empty
if p1 is right
and p2 is right
and p4 is left
then componen is c7a
if p1 is right
and p2 is right
and p4 is ok
then componen is c7a
if p1 is right
and p2 is right
and p4 is absent
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is left
then componen is c4a
if p1 is right
and p2 is right
and p4 is fleft
and r1 is right
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is ok
then componen is c4a
if p1 is right
and p2 is right
and p4 is fleft
and r1 is absent
then componen is empty
if p1 is right
and p2 is right

and p4 is fleft
and r1 is fleft
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is fright
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is cleft
and r2 is left
then componen is c7a
if p1 is right
and p2 is right
and p4 is fleft
and r1 is cleft
and r2 is right
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is cleft
and r2 is ok
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is cleft
and r2 is absent
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is cleft
and r2 is fleft
then componen is c4a
if p1 is right
and p2 is right
and p4 is fleft
and r1 is cleft
and r2 is fright
then componen is empty
if p1 is right
and p2 is right
and p4 is fleft
and r1 is cleft
and r2 is cleft
then componen is clash

```
if   p1 is right
 and p2 is right
 and p4 is fleft
 and r1 is cleft
 and r2 is cright
then componen is empty
if   p1 is right
 and p2 is right
 and p4 is fleft
 and r1 is cright
then componen is empty
if   p1 is right
 and p2 is right
 and p4 is cleft
then componen is empty
if   p1 is right
 and p2 is ok
then componen is c4a
if   p1 is right
 and p2 is absent
then componen is empty
if   p1 is right
 and p2 is fleft
then componen is empty
if   p1 is right
 and p2 is fright
then componen is empty
if   p1 is right
 and p2 is cleft
then componen is empty
if   p1 is right
 and p2 is cright
then componen is c7a
if   p1 is ok
 and r2 is left
then componen is c4a
if   p1 is ok
 and r2 is right
then componen is empty
if   p1 is ok
 and r2 is ok
then componen is c7a
if   p1 is ok
 and r2 is absent
 and p2 is left
then componen is c4c
if   p1 is ok
 and r2 is absent
 and p2 is right
then componen is empty
```

```
if   p1 is ok
 and r2 is absent
 and p2 is ok
then componen is c7c
if   p1 is ok
 and r2 is absent
 and p2 is absent
then componen is empty
if   p1 is ok
 and r2 is absent
 and p2 is fleft
then componen is empty
if   p1 is ok
 and r2 is absent
 and p2 is fright
then componen is empty
if   p1 is ok
 and r2 is absent
 and p2 is cleft
then componen is empty
if   p1 is ok
 and r2 is absent
 and p2 is cright
then componen is empty
if   p1 is ok
 and r2 is fleft
 and r1 is left
then componen is c7a
 and report This was an empty rule before
if   p1 is ok
 and r2 is fleft
 and r1 is right
then componen is empty
if   p1 is ok
 and r2 is fleft
 and r1 is ok
 and p2 is left
then componen is empty
if   p1 is ok
 and r2 is fleft
 and r1 is ok
 and p2 is right
then componen is c7a
if   p1 is ok
 and r2 is fleft
 and r1 is ok
 and p2 is ok
then componen is empty
if   p1 is ok
 and r2 is fleft
```

and r1 is ok
and p2 is absent
then componen is empty
if p1 is ok
and r2 is fleft
and r1 is ok
and p2 is fleft
then componen is empty
if p1 is ok
and r2 is fleft
and r1 is ok
and p2 is fright
then componen is empty
if p1 is ok
and r2 is fleft
and r1 is ok
and p2 is cleft
then componen is empty
if p1 is ok
and r2 is fleft
and r1 is ok
and p2 is cright
then componen is clash
and report component previously used
and report c4c (1 examples during elicitation)
and report c7a (1 examples during elicitation)
and report c7a (1 examples during testing)
if p1 is ok
and r2 is fleft
and r1 is absent
then componen is empty
if p1 is ok
and r2 is fleft
and r1 is fleft
then componen is empty
if p1 is ok
and r2 is fleft
and r1 is fright
then componen is empty
if p1 is ok
and r2 is fleft
and r1 is cleft
then componen is c4a
if p1 is ok
and r2 is fleft
and r1 is cright
then componen is empty
if p1 is ok
and r2 is fright
then componen is empty

if p1 is ok
and r2 is cleft
and p2 is left
then componen is empty
if p1 is ok
and r2 is cleft
and p2 is right
then componen is c4a
if p1 is ok
and r2 is cleft
and p2 is ok
then componen is c7a
if p1 is ok
and r2 is cleft
and p2 is absent
then componen is empty
if p1 is ok
and r2 is cleft
and p2 is fleft
then componen is empty
if p1 is ok
and r2 is cleft
and p2 is fright
then componen is empty
if p1 is ok
and r2 is cleft
and p2 is cleft
then componen is empty
if p1 is ok
and r2 is cleft
and p2 is cright
then componen is c4a
if p1 is ok
and r2 is cright
and p2 is left
then componen is empty
if p1 is ok
and r2 is cright
and p2 is right
then componen is empty
if p1 is ok
and r2 is cright
and p2 is ok
then componen is c4c
if p1 is ok
and r2 is cright
and p2 is absent
then componen is empty
if p1 is ok
and r2 is cright

and p2 is fleft
then componen is empty
if p1 is ok
and r2 is cright
and p2 is fright
then componen is empty
if p1 is ok
and r2 is cright
and p2 is cleft
then componen is empty
if p1 is ok
and r2 is cright
and p2 is cright
then componen is clash
if p1 is absent
and p2 is left
and p3 is left
then componen is empty
if p1 is absent
and p2 is left
and p3 is right
then componen is c4c
if p1 is absent
and p2 is left
and p3 is ok
then componen is empty
if p1 is absent
and p2 is left
and p3 is absent
then componen is clash
and report components previously used
and report c4c (2 examples during elicitation)
and report c7c (1 examples during elicitation)
and report c7c (1 examples during testing)
and report c4c (2 examples during testing)
if p1 is absent
and p2 is left
and p3 is fleft
then componen is empty
if p1 is absent
and p2 is left
and p3 is fright
then componen is clash
and report component previously used
and report c7c (4 examples during elicitation)
and report c4c (4 examples during elicitation)
and report c4c (1 examples during testing)
if p1 is absent
and p2 is left
and p3 is cleft

then componen is empty
if p1 is absent
and p2 is left
and p3 is cright
then componen is empty
if p1 is absent
and p2 is right
and p3 is left
then componen is clash
and report component previously used
and report c4a (1 examples during elicitation)
and report c7c (1 examples during elicitation)
and report c4c (2 examples during testing)
and report c4a (2 examples during testing)
if p1 is absent
and p2 is right
and p3 is right
then componen is empty
if p1 is absent
and p2 is right
and p3 is ok
then componen is empty
if p1 is absent
and p2 is right
and p3 is absent
then componen is empty
if p1 is absent
and p2 is right
and p3 is fleft
then componen is clash c4a c4c
and report THERE IS A CLASH
and report CLASH OCCURS AT THE START OF THE PROCESS
and report components used : C4A (2.75) ex.12
and report : C4C (2.50) ex.82
if p1 is absent
and p2 is right
and p3 is fright
then componen is empty
if p1 is absent
and p2 is right
and p3 is cleft
then componen is empty
if p1 is absent
and p2 is right
and p3 is cright
then componen is c7c
and report This was an empty rule before
if p1 is absent
and p2 is ok
then componen is c7c

```
if   p1 is absent
 and p2 is absent
then componen is c4c
if   p1 is absent
 and p2 is fleft
then componen is c4c
if   p1 is absent
 and p2 is fright
then componen is empty
if   p1 is absent
 and p2 is cleft
then componen is clash
 and report component previously used
 and report c4c ( 1 examples during elicitation )
 and report c7c ( 1 examples during elicitation )
 and report c4c ( 1 examples during testing )
if   p1 is absent
 and p2 is cright
then componen is c7c
 and report This was an empty rule before
if   p1 is fleft
 and r2 is left
then componen is empty
if   p1 is fleft
 and r2 is right
then componen is empty
if   p1 is fleft
 and r2 is ok
then componen is c4a
if   p1 is fleft
 and r2 is absent
 and p3 is left
then componen is empty
if   p1 is fleft
 and r2 is absent
 and p3 is right
then componen is c7c
if   p1 is fleft
 and r2 is absent
 and p3 is ok
then componen is empty
if   p1 is fleft
 and r2 is absent
 and p3 is absent
then componen is clash
if   p1 is fleft
 and r2 is absent
 and p3 is fleft
then componen is empty
if   p1 is fleft
```

```
 and r2 is absent
 and p3 is fright
then componen is c7c
if  p1 is fleft
 and r2 is absent
 and p3 is cleft
then componen is empty
if  p1 is fleft
 and r2 is absent
 and p3 is cright
then componen is empty
if  p1 is fleft
 and r2 is fleft
then componen is empty
if  p1 is fleft
 and r2 is fright
then componen is empty
if  p1 is fleft
 and r2 is cleft
then componen is c7c
if  p1 is fleft
 and r2 is cright
then componen is empty
if  p1 is fright
 and r2 is left
 and r1 is left
then componen is empty
if  p1 is fright
 and r2 is left
 and r1 is right
then componen is empty
if  p1 is fright
 and r2 is left
 and r1 is ok
then componen is clash
 and report THERE IS A CLASH
 and report CLASH OCCURS AT THE START OF THE PROCESS
 and report components used : C7A ( 1.50 ) ex.46
 and report : C4A ( 1.75 ) ex.53
 and report : C7A ( 0.75 ) ex.57
if  p1 is fright
 and r2 is left
 and r1 is absent
then componen is empty
if  p1 is fright
 and r2 is left
 and r1 is fleft
then componen is empty
if  p1 is fright
 and r2 is left
```

and r1 is fright
then componen is empty
if p1 is fright
and r2 is left
and r1 is cleft
then componen is c4a
if p1 is fright
and r2 is left
and r1 is cright
then componen is empty
if p1 is fright
and r2 is right
then componen is empty
if p1 is fright
and r2 is ok
then componen is c4a
if p1 is fright
and r2 is absent
then componen is c4a
if p1 is fright
and r2 is fleft
and p2 is left
then componen is empty
if p1 is fright
and r2 is fleft
and p2 is right
then componen is c4a
if p1 is fright
and r2 is fleft
and p2 is ok
then componen is empty
if p1 is fright
and r2 is fleft
and p2 is absent
then componen is empty
if p1 is fright
and r2 is fleft
and p2 is fleft
then componen is empty
if p1 is fright
and r2 is fleft
and p2 is fright
then componen is empty
if p1 is fright
and r2 is fleft
and p2 is cleft
then componen is empty
if p1 is fright
and r2 is fleft
and p2 is cright

```
then componen is c7a
if   p1 is fright
 and r2 is fright
then componen is empty
if   p1 is fright
 and r2 is cleft
then componen is c7a
if   p1 is fright
 and r2 is cright
then componen is clash
 and report THERE IS A CLASH
 and report CLASH OCCURS AT THE START OF THE PROCESS
 and report components used : C4A ( 2.50 ) ex.4
 and report : C7A ( 2.00 ) ex.8
 and report : C7A ( 1.50 ) ex.33
 and report : C4A ( 2.25 ) ex.41
 and report : C4A ( 2.75 ) ex.96
if   p1 is cleft
then componen is c4a
if   p1 is cright
 and p2 is left
then componen is empty
if   p1 is cright
 and p2 is right
then componen is empty
if   p1 is cright
 and p2 is ok
then componen is c4a
if   p1 is cright
 and p2 is absent
then componen is empty
if   p1 is cright
 and p2 is fleft
then componen is empty
if   p1 is cright
 and p2 is fright
then componen is empty
if   p1 is cright
 and p2 is cleft
then componen is empty
if   p1 is cright
 and p2 is cright
then componen is c7a
if   peak1 >= 1.397516
 and peak1 <= 1.39822
then p1 is ok
if   peak1 < 1.397516
 and 1.397516 - peak1 < lim1
then p1 is cleft
if   peak1 < 1.397516
```

```
 and 1.397516 - peak1 >= lim1
 and 1.397516 - peak1 < lim2
then p1 is left
if   peak1 < 1.397516
 and 1.397516 - peak1 >= lim2
then p1 is fleft
if   peak1 > 1.39822
 and peak1 - 1.39822 < lim1
then p1 is cright
if   peak1 > 1.39822
 and peak1 - 1.39822 >= lim1
 and peak1 - 1.39822 < lim2
then p1 is right
if   peak1 > 1.39822
 and peak1 - 1.39822 >= lim2
then p1 is fright
if   peak2 >= 1.399327
 and peak2 <= 1.399546
then p2 is ok
if   peak2 < 1.399327
 and 1.399327 - peak2 < lim1
then p2 is cleft
if   peak2 < 1.399327
 and 1.399327 - peak2 >= lim1
 and 1.399327 - peak2 < lim2
then p2 is left
if   peak2 < 1.399327
 and 1.399327 - peak2 >= lim2
then p2 is fleft
if   peak2 > 1.399546
 and peak2 - 1.399546 < lim1
then p2 is cright
if   peak2 > 1.399546
 and peak2 - 1.399546 >= lim1
 and peak2 - 1.399546 < lim2
then p2 is right
if   peak2 > 1.399546
 and peak2 - 1.399546 >= lim2
then p2 is fright
if   peak3 >= 1.404048
 and peak3 <= 1.405241
then p3 is ok
if   peak3 < 1.404048
 and 1.404048 - peak3 < lim1
then p3 is cleft
if   peak3 < 1.404048
 and 1.404048 - peak3 >= lim1
 and 1.404048 - peak3 < lim2
then p3 is left
if   peak3 < 1.404048
```

```
 and 1.404048 - peak3 >= lim2
then p3 is fleft
if   peak3 > 1.405241
 and peak3 - 1.405241 < lim1
then p3 is cright
if   peak3 > 1.405241
 and peak3 - 1.405241 >= lim1
 and peak3 - 1.405241 < lim2
then p3 is right
if   peak3 > 1.405241
 and peak3 - 1.405241 >= lim2
then p3 is fright
if   peak4 >= 1.406214
then p4 is ok
if   peak4 < 1.406214
 and 1.406214 - peak4 < lim1
then p4 is cleft
if   peak4 < 1.406214
 and 1.406214 - peak4 >= lim1
 and 1.406214 - peak4 < lim2
then p4 is left
if   peak4 < 1.406214
 and 1.406214 - peak4 >= lim2
then p4 is fleft
if   level1 >= 56
 and level1 <= 64
then r1 is ok
if   level1 < 56
 and level1 >= 49
then r1 is cleft
if   level1 < 49
 and level1 >= 42
then r1 is left
if   level1 < 42
then r1 is fleft
if   level1 > 64
 and level1 <= 71
then r1 is cright
if   level1 > 71
 and level1 <= 78
then r1 is right
if   level1 > 78
then r1 is fright
if   level2 >= 58
 and level2 <= 65
then r2 is ok
if   level2 < 58
 and level2 >= 52
then r2 is cleft
if   level2 < 52
```

```
 and level2 >= 46
then r2 is left
if  level2 < 46
then r2 is fleft
if  level2 > 65
 and level2 <= 72
then r2 is cright
if  level2 > 72
 and level2 <= 78
then r2 is right
if  level2 > 78
then r2 is fright
query  componen is
 query options auto
query  componen
```

Appendix Seven

Listing of knowledge-base of the passband (search 1)

```
when done process
then command load c:\newmarket\pamod
when process is end
then report Please connect another filter
 and command reset data
 and command load c:\newmarket\search1
if  vripple >= 0.0
 and vripple <= 1.0
then ripple is ok
if  vripple > 1
 and vripple <= 2.60
then ripple is closerigt
if  vripple > 2.60
 and vripple <= 4.20
then ripple is right
if  vripple > 4.20
then ripple is farright
if  vinloss >= 0.0
 and vinloss < 0.5
then inloss is closeleft
if  vinloss >= 0.50
 and vinloss <= 5.0
then inloss is ok
if  vinloss > 5.0
 and vinloss <= 5.33
then inloss is closerigt
if  vinloss > 5.33
 and vinloss <= 5.66
then inloss is right
if  vinloss > 5.66
then inloss is farright
if  vlowpb >= 0.0
 and vlowpb <= 4.0
then lowpb is ok
if  vlowpb > 4.0
 and vlowpb <= 5.05
then lowpb is closerigt
if  vlowpb > 5.05
 and vlowpb <= 6.1
then lowpb is right
if  vlowpb > 6.1
then lowpb is farright
if  vhighpb >= 0.0
 and vhighpb <= 4.0
then highpb is ok
if  vhighpb > 4.0
 and vhighpb <= 6.93
then highpb is closerigt
if  vhighpb > 6.93
 and vhighpb <= 9.86
```

then highpb is right
if vhighpb > 9.86
then highpb is farright
if vcarret < 5.8
then carret is farleft
if vcarret >= 5.8
and vcarret < 7.9
then carret is left
if vcarret >= 7.9
and vcarret < 10
then carret is closeleft
if vcarret >= 10
and vcarret <= 12.10
then carret is closeok
if vcarret > 12.1
and vcarret <= 14.20
then carret is ok
if vcarret > 14.20
then carret is farok
if vlowsb < 32.6
then lowsb is farleft
if vlowsb >= 32.6
and vlowsb < 38.8
then lowsb is left
if vlowsb >= 38.8
and vlowsb < 45
then lowsb is closeleft
if vlowsb >= 45
and vlowsb <= 51.2
then lowsb is closeok
if vlowsb > 51.2
and vlowsb <= 57.4
then lowsb is middleok
if vlowsb > 57.4
then lowsb is farok
if vhighsb < 27.5
then highsb is farleft
if vhighsb >= 27.5
and vhighsb < 36.25
then highsb is left
if vhighsb >= 36.25
and vhighsb < 45.0
then highsb is closeleft
if vhighsb >= 45.0
and vhighsb <= 53.75
then highsb is closeok
if vhighsb > 53.75
and vhighsb <= 62.5
then highsb is middleok
if vhighsb > 62.5

```
then highsb is farok
if   vlowsbret < 33.76
then lowsbret is farleft
if   vlowsbret >= 33.76
 and vlowsbret < 39.38
then lowsbret is left
if   vlowsbret >= 39.38
 and vlowsbret < 45.0
then lowsbret is closeleft
if   vlowsbret >= 45.00
 and vlowsbret <= 50.62
then lowsbret is closeok
if   vlowsbret > 50.62
 and vlowsbret <= 56.24
then lowsbret is middleok
if   vlowsbret > 56.24
then lowsbret is farok
if   vhighsbret < 35.96
then highsbret is farleft
if   vhighsbret >= 35.96
 and vhighsbret < 40.48
then highsbret is left
if   vhighsbret >= 40.48
 and vhighsbret < 45.00
then highsbret is closeleft
if   vhighsbret >= 45.00
 and vhighsbret <= 49.52
then highsbret is closeok
if   vhighsbret > 49.52
 and vhighsbret <= 54.04
then highsbret is middleok
if   vhighsbret > 54.04
then highsbret is farok
if   ripple is ok
 and inloss is ok
 and lowpb is ok
 and highpb is ok
 and carret is ok or middleok or farok or closeok
 and lowsb is ok or middleok or farok or closeok
 and highsb is ok or middleok or farok or closeok
 and lowsbret is ok or middleok or farok or closeok
 and highsbret is ok or middleok or farok or closeok
 and difference <= 3.0
then process is end
if   vlowsbret >= vhighsbret
then difference = vlowsbret - vhighsbret
if   vlowsbret < vhighsbret
then difference = vhighsbret - vlowsbret
if   check process
then check ripple
```

```
    and check inloss
    and check lowpb
    and check highpb
    and check carret
    and check lowsb
    and check highsb
    and check lowsbret
    and check highsbret
    and check difference
default     process is noend
query   process
  query options auto
```

Appendix Eight

Listing of knowledge-base of the passband (search 2)

```
when tuner is X
 and process is Y
then report component to be used [tuner]
 and report process to be used [Y]
 and report "" " end of testing " ""
 and command reset data
 and command load a:\numbers
if  ripple is ok
 and lowsbret is farleft
then tuner is clash
 and report components previously used
 and report t2c ( 1 example during elicitation )
 and report t1c ( 1 example during elicitation )
if  ripple is ok
 and lowsbret is left
 and lowsb is farleft
then tuner is empty
if  ripple is ok
 and lowsbret is left
 and lowsb is left
then tuner is empty
if  ripple is ok
 and lowsbret is left
 and lowsb is closeleft
 and highsb is farleft
then tuner is empty
if  ripple is ok
 and lowsbret is left
 and lowsb is closeleft
 and highsb is left
then tuner is empty
if  ripple is ok
 and lowsbret is left
 and lowsb is closeleft
 and highsb is closeleft
then tuner is clash
 and report components previously used
 and report c7c ( 1 examples during elicitation )
 and report t2a ( 1 examples during elicitation )
 and report t1a ( 1 examples during elicitation )
 and report t3a ( 1 examples during elicitation )
if  ripple is ok
 and lowsbret is left
 and lowsb is closeleft
 and highsb is closeok
then tuner is c4a
if  ripple is ok
 and lowsbret is left
 and lowsb is closeleft
 and highsb is middleok
```

then tuner is empty
if ripple is ok
and lowsbret is left
and lowsb is closeleft
and highsb is farok
then tuner is empty
if ripple is ok
and lowsbret is left
and lowsb is closeok
then tuner is empty
if ripple is ok
and lowsbret is left
and lowsb is middleok
and highpb is ok
then tuner is c7a
if ripple is ok
and lowsbret is left
and lowsb is middleok
and highpb is closerigt
then tuner is t2c
if ripple is ok
and lowsbret is left
and lowsb is middleok
and highpb is right
then tuner is empty
if ripple is ok
and lowsbret is left
and lowsb is middleok
and highpb is farright
then tuner is empty
if ripple is ok
and lowsbret is left
and lowsb is farok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is left
and highsbret is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is left
and highsbret is left
then tuner is c7a
if ripple is ok
and lowsbret is closeleft

and highsb is left
and highsbret is closeleft
then tuner is c7a
if ripple is ok
and lowsbret is closeleft
and highsb is left
and highsbret is closeok
then tuner is clash
and report components previously used
and report c4c (1 examples during elicitation)
and report c7a (1 examples during elicitation)
if ripple is ok
and lowsbret is closeleft
and highsb is left
and highsbret is middleok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is left
and highsbret is farok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is closeleft
and highsbret is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is closeleft
and highsbret is left
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is closeleft
and highsbret is closeleft
then tuner is c4a
if ripple is ok

and lowsbret is closeleft
and highsb is closeleft
and lowsb is closeleft
and highsbret is closeok
then tuner is t2a
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is closeleft
and highsbret is middleok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is closeleft
and highsbret is farok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is closeok
then tuner is c4a
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is middleok
and highsbret is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is middleok
and highsbret is left
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is middleok
and highsbret is closeleft
then tuner is clash
and report components previously used
and report c7a (1 examples during elicitation)
and report c7c (1 examples during elicitation)
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is middleok
and highsbret is closeok
then tuner is clash
and report components previously used

and report c7a (1 example during elicitation)
and report c7c (1 example during elicitation)
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is middleok
and highsbret is middleok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is middleok
and highsbret is farok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is farok
and highpb is ok
then tuner is clash
and report components previously used
and report c7c (2 examples during elicitation)
and report t2a (1 examples during elicitation)
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is farok
and highpb is closerigt
then tuner is t3c
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is farok
and highpb is right
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeleft
and lowsb is farok
and highpb is farright
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeok
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeok
and lowsb is left

then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is closeok
and lowsb is closeleft
then tuner is t3c
if ripple is ok
and lowsbret is closeleft
and highsb is closeok
and lowsb is closeok
then tuner is c4a
if ripple is ok
and lowsbret is closeleft
and highsb is closeok
and lowsb is middleok
then tuner is clash
and report components previously used
and report c4a (1 example during elicitation)
and report c7a (1 example during elicitation)
and report c4a (1 example during testing)
if ripple is ok
and lowsbret is closeleft
and highsb is closeok
and lowsb is farok
then tuner is c4a
if ripple is ok
and lowsbret is closeleft
and highsb is middleok
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is middleok
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is middleok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is middleok
and lowsb is closeok
then tuner is clash
and report components previously used
and report t2a (1 example during elictation)
and report c4a (1 example during elictation)
and report c7a (2 example during testing)
and report c4a (1 example during testing)

if ripple is ok
and lowsbret is closeleft
and highsb is middleok
and lowsb is middleok
then tuner is c4a
if ripple is ok
and lowsbret is closeleft
and highsb is middleok
and lowsb is farok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is farok
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is farok
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is farok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is farok
and lowsb is closeok
then tuner is empty
if ripple is ok
and lowsbret is closeleft
and highsb is farok
and lowsb is middleok
then tuner is c7a
if ripple is ok
and lowsbret is closeleft
and highsb is farok
and lowsb is farok
then tuner is c7c
if ripple is ok
and lowsbret is closeok
and highsbret is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is left
then tuner is empty
if ripple is ok
and lowsbret is closeok

and highsbret is closeleft
then tuner is c4c
if ripple is ok
and lowsbret is closeok
and highsbret is closeok
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is closeok
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is closeok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is closeok
and lowsb is closeok
then tuner is c4a
and report This was an empty rule before
if ripple is ok
and lowsbret is closeok
and highsbret is closeok
and lowsb is middleok
then tuner is c4a
if ripple is ok
and lowsbret is closeok
and highsbret is closeok
and lowsb is farok
then tuner is clash
and report components previously used
and report c7c (1 example during elicitation)
and report c4a (1 example during elicitation)
and report c7a (1 example during elicitation)
and report c7a (1 example during testing)
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is left
then tuner is empty
if ripple is ok
and lowsbret is closeok

and highsbret is middleok
and highsb is closeleft
then tuner is c7c
and report This was an empty rule before
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is closeok
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is closeok
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is closeok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is closeok
and lowsb is closeok
then tuner is c4a
and report This was an empty rule before
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is closeok
and lowsb is middleok
then tuner is c4a
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is closeok
and lowsb is farok
then tuner is C4a
and report previously component used
and report t2c (2 examples during elicitation)
and report t2a (1 examples during elicitation)
and report c4a (1 examples during elicitation)
and report c4a (3 examples during testing)
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is middleok

and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is middleok
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is middleok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is middleok
and lowsb is closeok
then tuner is t3a
and report This was an empty rule before
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is middleok
and lowsb is middleok
then tuner is clash
and report components previously used
and report c7a (1 examples during elicitation)
and report t2a (1 examples during elicitation)
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is middleok
and lowsb is farok
then tuner is clash
and report components previously used
and report t3a (1 example during elicitation)
and report t2a (2 example during elicitation)
if ripple is ok
and lowsbret is closeok
and highsbret is middleok
and highsb is farok
then tuner is c4a
and report This was an empty rule before
if ripple is ok
and lowsbret is closeok
and highsbret is farok
and lowsb is farleft
then tuner is empty

```
if  ripple is ok
 and lowsbret is closeok
 and highsbret is farok
 and lowsb is left
then tuner is empty
if  ripple is ok
 and lowsbret is closeok
 and highsbret is farok
 and lowsb is closeleft
then tuner is empty
if  ripple is ok
 and lowsbret is closeok
 and highsbret is farok
 and lowsb is closeok
then tuner is t2a
if  ripple is ok
 and lowsbret is closeok
 and highsbret is farok
 and lowsb is middleok
then tuner is c4a
 and report This was an empty rule before
if  ripple is ok
 and lowsbret is closeok
 and highsbret is farok
 and lowsb is farok
then tuner is c4a
if  ripple is ok
 and lowsbret is middleok
 and highsbret is farleft
then tuner is c7c
 and report This was an empty rule before
if  ripple is ok
 and lowsbret is middleok
 and highsbret is left
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is closeleft
then tuner is c7a
if  ripple is ok
 and lowsbret is middleok
 and highsbret is closeok
 and lowsb is farleft
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is closeok
 and lowsb is left
then tuner is empty
if  ripple is ok
```

and lowsbret is middleok
and highsbret is closeok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is closeok
and lowsb is closeok
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is closeok
and lowsb is middleok
then tuner is c4c
if ripple is ok
and lowsbret is middleok
and highsbret is closeok
and lowsb is farok
and highsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is closeok
and lowsb is farok
and highsb is left
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is closeok
and lowsb is farok
and highsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is closeok
and lowsb is farok
and highsb is closeok
then tuner is clash
and report components previously used
and report t3a (1 example during elicitation)
and report c7a (2 example during elicitation)
and report t2c (1 example during elicitation)
and report c4c (1 example during elicitation)
and report c4c (1 example during testing)
if ripple is ok
and lowsbret is middleok
and highsbret is closeok
and lowsb is farok
and highsb is middleok
then tuner is c7a

```
if  ripple is ok
 and lowsbret is middleok
 and highsbret is closeok
 and lowsb is farok
 and highsb is farok
then tuner is c7a
 and report This was an empty rule before
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
 and highsb is farleft
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
 and highsb is left
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
 and highsb is closeleft
 and lowsb is farleft
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
 and highsb is closeleft
 and lowsb is left
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
 and highsb is closeleft
 and lowsb is closeleft
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
 and highsb is closeleft
 and lowsb is closeok
then tuner is empty
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
 and highsb is closeleft
 and lowsb is middleok
then tuner is c4c
if  ripple is ok
 and lowsbret is middleok
 and highsbret is middleok
```

and highsb is closeleft
and lowsb is farok
then tuner is t2c
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is closeok
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is closeok
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is closeok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is closeok
and lowsb is closeok
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is closeok
and lowsb is middleok
then tuner is c7c
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is closeok
and lowsb is farok
then tuner is t3c
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is middleok
then tuner is t2a
if ripple is ok
and lowsbret is middleok
and highsbret is middleok
and highsb is farok
then tuner is empty
if ripple is ok

and lowsbret is middleok
and highsbret is farok
and lowsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is left
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is closeleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is closeok
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is middleok
then tuner is c4c
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is farok
and highsb is farleft
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is farok
and highsb is left
then tuner is empty
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is farok
and highsb is closeleft
then tuner is clash
and report components previously used
and report c7c (1 examples during elicitation)
and report t3a (1 examples during elicitation)
if ripple is ok
and lowsbret is middleok
and highsbret is farok
and lowsb is farok
and highsb is closeok

then tuner is clash
 and report components previously used
 and report t3c (1 example during elicitation)
 and report c7c (2 example during elicitation)
 and report c7c (2 example during testing)
 and report t2a (1 example during testing)
if ripple is ok
 and lowsbret is middleok
 and highsbret is farok
 and lowsb is farok
 and highsb is middleok
then tuner is c4a
 and report This was an empty rule before
if ripple is ok
 and lowsbret is middleok
 and highsbret is farok
 and lowsb is farok
 and highsb is farok
then tuner is empty
if ripple is ok
 and lowsbret is farok
 and highsbret is farleft
then tuner is empty
if ripple is ok
 and lowsbret is farok
 and highsbret is left
then tuner is empty
if ripple is ok
 and lowsbret is farok
 and highsbret is closeleft
then tuner is empty
if ripple is ok
 and lowsbret is farok
 and highsbret is closeok
then tuner is c7a
if ripple is ok
 and lowsbret is farok
 and highsbret is middleok
then tuner is t1a
if ripple is ok
 and lowsbret is farok
 and highsbret is farok
then tuner is c7c
if ripple is closerigt
 and lowsbret is farleft
 and lowpb is ok
then tuner is t3c
if ripple is closerigt
 and lowsbret is farleft
 and lowpb is closerigt

then tuner is c7a
if ripple is closerigt
and lowsbret is farleft
and lowpb is right
then tuner is empty
if ripple is closerigt
and lowsbret is farleft
and lowpb is farright
then tuner is empty
if ripple is closerigt
and lowsbret is left
then tuner is t3a
if ripple is closerigt
and lowsbret is closeleft
and highsb is farleft
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is left
then tuner is t3c
if ripple is closerigt
and lowsbret is closeleft
and highsb is closeleft
then tuner is clash t2a / t2c)
and report component previously used
and report t2a (1 examples during elicitation)
and report t2c (1 examples during elicitation)
if ripple is closerigt
and lowsbret is closeleft
and highsb is closeok
and lowsb is farleft
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is closeok
and lowsb is left
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is closeok
and lowsb is closeleft
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is closeok
and lowsb is closeok
then tuner is t3a
if ripple is closerigt
and lowsbret is closeleft
and highsb is closeok

and lowsb is middleok
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is closeok
and lowsb is farok
then tuner is t2a
if ripple is closerigt
and lowsbret is closeleft
and highsb is middleok
and lowsb is farleft
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is middleok
and lowsb is left
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is middleok
and lowsb is closeleft
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is middleok
and lowsb is closeok
then tuner is empty
if ripple is closerigt
and lowsbret is closeleft
and highsb is middleok
and lowsb is middleok
then tuner is t2a
if ripple is closerigt
and lowsbret is closeleft
and highsb is middleok
and lowsb is farok
then tuner is c4a
if ripple is closerigt
and lowsbret is closeleft
and highsb is farok
then tuner is empty
if ripple is closerigt
and lowsbret is closeok
and highpb is ok
and highsbret is farleft
then tuner is empty
if ripple is closerigt
and lowsbret is closeok
and highpb is ok
and highsbret is left

then tuner is empty
if ripple is closerigt
and lowsbret is closeok
and highpb is ok
and highsbret is closeleft
then tuner is empty
if ripple is closerigt
and lowsbret is closeok
and highpb is ok
and highsbret is closeok
then tuner is t2a
if ripple is closerigt
and lowsbret is closeok
and highpb is ok
and highsbret is middleok
then tuner is clash
and report components previously used
and report c7c (1 example during elicitation)
and report t2a (1 example during elicitation)
and report t2c (1 example during testing)
and report t2a (1 example during testing)
if ripple is closerigt
and lowsbret is closeok
and highpb is ok
and highsbret is farok
then tuner is t2a
if ripple is closerigt
and lowsbret is closeok
and highpb is closerigt
then tuner is t3a
and report This was an empty rule before
if ripple is closerigt
and lowsbret is closeok
and highpb is right
then tuner is clash
and report component previously used
and report t2a (1 examples during elicitation)
and report t1a (1 examples during elicitation)
if ripple is closerigt
and lowsbret is closeok
and highpb is farright
then tuner is empty
if ripple is closerigt
and lowsbret is middleok
and highsb is farleft
then tuner is empty
if ripple is closerigt
and lowsbret is middleok
and highsb is left
then tuner is empty

if ripple is closerigt
and lowsbret is middleok
and highsb is closeleft
then tuner is t2c
if ripple is closerigt
and lowsbret is middleok
and highsb is closeok
and highsbret is farleft
then tuner is empty
if ripple is closerigt
and lowsbret is middleok
and highsb is closeok
and highsbret is left
then tuner is empty
if ripple is closerigt
and lowsbret is middleok
and highsb is closeok
and highsbret is closeleft
then tuner is empty
if ripple is closerigt
and lowsbret is middleok
and highsb is closeok
and highsbret is closeok
then tuner is empty
if ripple is closerigt
and lowsbret is middleok
and highsb is closeok
and highsbret is middleok
then tuner is t2a
if ripple is closerigt
and lowsbret is middleok
and highsb is closeok
and highsbret is farok
then tuner is clash
and report component previously used
and report c7a (1 examples during elicitation)
and report t2a (1 examples during elicitation)
and report c7c (1 examples during testing)
and report t2a (1 examples during testing)
if ripple is closerigt
and lowsbret is middleok
and highsb is middleok
then tuner is t2a
if ripple is closerigt
and lowsbret is middleok
and highsb is farok
then tuner is t2a
if ripple is closerigt
and lowsbret is farok
then tuner is t2a

```
if  ripple is right
 and lowsbret is farleft
then tuner is empty
if  ripple is right
 and lowsbret is left
then tuner is empty
if  ripple is right
 and lowsbret is closeleft
then tuner is t2a
if  ripple is right
 and lowsbret is closeok
 and highsb is farleft
then tuner is empty
if  ripple is right
 and lowsbret is closeok
 and highsb is left
then tuner is empty
if  ripple is right
 and lowsbret is closeok
 and highsb is closeleft
then tuner is t1a
if  ripple is right
 and lowsbret is closeok
 and highsb is closeok
then tuner is t3a
if  ripple is right
 and lowsbret is closeok
 and highsb is middleok
then tuner is t3a
if  ripple is right
 and lowsbret is closeok
 and highsb is farok
then tuner is empty
if  ripple is right
 and lowsbret is middleok
 and lowsb is farleft
then tuner is empty
if  ripple is right
 and lowsbret is middleok
 and lowsb is left
then tuner is empty
if  ripple is right
 and lowsbret is middleok
 and lowsb is closeleft
then tuner is empty
if  ripple is right
 and lowsbret is middleok
 and lowsb is closeok
then tuner is empty
if  ripple is right
```

```
 and lowsbret is middleok
 and lowsb is middleok
then tuner is t3a
if  ripple is right
 and lowsbret is middleok
 and lowsb is farok
then tuner is t3a
if  ripple is right
 and lowsbret is farok
then tuner is t3a
if  ripple is farright
 and lowpb is ok
 and highpb is ok
then tuner is t3a
if  ripple is farright
 and lowpb is ok
 and highpb is closerigt
then tuner is t3a
if  ripple is farright
 and lowpb is ok
 and highpb is right
then tuner is t3a
if  ripple is farright
 and lowpb is ok
 and highpb is farright
 and highsb is farleft
then tuner is empty
if  ripple is farright
 and lowpb is ok
 and highpb is farright
 and highsb is left
then tuner is empty
if  ripple is farright
 and lowpb is ok
 and highpb is farright
 and highsb is closeleft
then tuner is empty
if  ripple is farright
 and lowpb is ok
 and highpb is farright
 and highsb is closeok
then tuner is t2c
if  ripple is farright
 and lowpb is ok
 and highpb is farright
 and highsb is middleok
then tuner is t3a
 and report This was an empty rule before
if  ripple is farright
 and lowpb is ok
```

```
  and highpb is farright
  and highsb is farok
 then tuner is t3a
 if  ripple is farright
  and lowpb is closerigt
 then tuner is t2c
 if  ripple is farright
  and lowpb is right
 then tuner is t1a
 if  ripple is farright
  and lowpb is farright
 then tuner is empty
query  tuner
 query options auto
```

Appendix Nine

Listing of HP-Basic program which displays the output of the networks

```
1 ! THIS IS THE LAST VERSION AT 25/6/91
2 ! IT SENDS THE RESULTS TO DISC
3 !
4 !
10  INTEGER Layer,Node,Max_node,Prev_max_node,Prev_node,
        Next_node, Loopx
20  DIM Inputx(57,4),Weight(57,11,3),Threshold(11,3)
40  DIM Frx(50),Mrk(50),Binno(50),
41  DIM M$[40],F$[20],A(100,2),Name$[6]
42  !
43  ASSIGN @Na TO 711
44  ASSIGN @Prt TO 1
45  Prt=1
46  PRINTER IS 1
47  !
48  ASSIGN @Na_nofmt TO 711;FORMAT OFF
49  Meas_complete=4
50   !
51   !
52  CLEAR @Na
53  OUTPUT @Na;"IPR;"
54  OUTPUT @Na;"IAR;IA1;IR1;IB1;"
55  OUTPUT @Na;"BP0;"
56  !
57  OUTPUT @Na;"ST5;SM1;SFR1401500HZ;DF7;DIV1DBR;REF0DBR;"
58  OUTPUT @Na;"SAM+5.8DBM;FM2;"
59  OUTPUT @Na;"RPS50%;BW3;AV0;"
60  !
61  DISP "      Insert S/C and press 'CONT'"
62  PAUSE
63  DISP ""
64  !
65  OUTPUT @Na;"DM1;TRG;"
66  Meascomp
67  ENTER @Na USING "%,2A";Junk$
68  ENTER @Na_nofmt;Sc_ref
69  PRINT "Ref   ";Sc_ref
70  OUTPUT @Na;"DIV5DBR;REF-26DBR;"
71  !
72  DISP "      Insert unit and press 'CONT'"
73  PAUSE
74  DISP ""
75  OUTPUT @Na;"DM1;TRG;"
76  Meascomp
77  ENTER @Na USING "%,2A";Junk$
78  ENTER @Na_nofmt;Ins_loss
79  PRINT "Approx. Insertion loss  ";-Ins_loss-26
80  !
81  Again:  !
82    PRINTER IS 1
```

```
83 Menu1: !
84    OUTPUT @Na;"DF7;SAM+5.8DBM;FM1;ST1;SWT1SEC;
              DIV10DBR; RPS100%;"
85    OUTPUT @Na;"REF-26DBR;"
86    OUTPUT @Na;"FRC1401500HZ;FRS40KHZ;SM1;BW3;"
87    LOCAL 711
88  !
89 Sband: !
90 Menu2: !
91    DISP "PRESS 'CONT' TO MEASURE"
92    PAUSE
93  !
94 Sweep:    !
95    DISP ""
96    FOR I=1 TO 21
97       A(I,1)=0
98       A(I,2)=0
99    NEXT I
100 !------------------------------------- LO
102 I=0
104 F1=1.380000
105 F2=1.400000
106 Screen1=0
107 Screen2=400
108 Screen_step=Screen2/20
110 !
111 OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA"&VAL$(F1)&"MHZ; FRB"
          &VAL$(F2) &"MHZ;FM1;"
112 OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
113 Measready
114 FOR S=Screen1 TO Screen2 STEP Screen_step
116   I=I+1
118   OUTPUT @Na;"MKP"&VAL$(S)&";"
119   OUTPUT @Na;"DM1;"
120   ENTER @Na;Level
121   OUTPUT @Na;"MP1;"
122   ENTER @Na;Freq1
123   A(I,1)=Freq/1000
124   A(I,2)=-Level+Ins_loss+.6
125 NEXT S
126 Loop1=0
128 FOR J=1 TO 19
129   Loop1=Loop1+1
132   Inputx(Loop1,1)=A(J,2)
133 !    PRINT USING Image11;J,A(J,1),Loop1,Inputx(Loop1,1)
135 NEXT J
136 !---------------------------------------------- MID
150 F1=1.400000
151 F2=1.404000
152 Screen1=0
```

```
153 Screen2=400
154 Screen_step=Screen2/20
157 Lmin=+1000
158 !
159 OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA"&VAL$(F1)&"MHZ;FRB"
        &VAL$(F2)&"MHZ;FM1;"
160 OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
161 Measready
162 I=0
164 FOR S=Screen1 TO Screen2 STEP Screen_step
165   I=I+1
168   OUTPUT @Na;"MKP"&VAL$(S)&";"
169   OUTPUT @Na;"DM1;"
170   ENTER @Na;Level
171   OUTPUT @Na;"MP1;"
172   ENTER @Na;Freq1
173   A(I,1)=Freq/1000
174   A(I,2)=-Level+Ins_loss+.6
175 NEXT S
177 ! PRINT
178 FOR J=1 TO 19
179    Loop1=Loop1+1
182    Inputx(Loop1,1)=A(J,2)
183 ! PRINT USING Image11;J,A(J,1),Loop1,Inputx(Loop1,1)
185 NEXT J
186 !------------------------------------ HI
187 F1=1.404000
188 F2=1.420000
189 Screen1=0
190 Screen2=400
191 Screen_step=Screen2/20
192 Lmin=+1000
193 !
194 OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA"&VAL$(F1)&"MHZ;FRB"
        &VAL$(F2)&"MHZ;FM1;"
195 OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
196 Measready
197 I=0
198 FOR S=Screen1 TO Screen2 STEP Screen_step
199   I=I+1
200   OUTPUT @Na;"MKP"&VAL$(S)&";"
201   OUTPUT @Na;"DM1;"
202   ENTER @Na;Level
203   OUTPUT @Na;"MP1;"
204   ENTER @Na;Freq1
205   A(I,1)=Freq/1000
206   A(I,2)=-Level+Ins_loss+.6
207 NEXT S
208 Image11:IMAGE   DD,2X,DDDD.DDD,2X,DD,2X,DDD.D
210 FOR J=1 TO 19
```

```
211    Loop1=Loop1+1
214    Inputx(Loop1,1)=A(J,2)
215 !  PRINT USING Image11;J,A(J,1),Loop1,Inputx(Loop1,1)
217  NEXT J
218 !
219 !
222 PRINT
223 PRINT
224 INPUT "ENTER FILENAME FOR 57 POINTS",File$
226 CREATE ASCII File$,50
227 ASSIGN @Disk TO File$
228 FOR I=1 TO 57
229 !  PRINT I;Inputx(I,1)
230    OUTPUT @Disk;Inputx(I,1)
232    Inputx(I,1)=((Inputx(I,1)*.01)+0.1)
234  !  PRINT I;Inputx(I,1)
235 NEXT I
236 ASSIGN @Disk TO *1
237  !
238  !
239  !
240 RESTORE
241 !
242 !
243 FOR No_of_out_nodes=1 TO 4
244  FOR Loop1=1 TO 11
245   FOR Loop2=1 TO 57
246      READ Weight(Loop2,Loop1,1)
247!     PRINT Weight(Loop2,Loop1,1)
248    NEXT Loop2
249  NEXT Loop1
250  FOR Loop1=1 TO 10
251   FOR Loop2=1 TO 11
252      READ Weight(Loop2,Loop1,2)
253!     PRINT Weight(Loop2,Loop1,2)
254   NEXT Loop2
255  NEXT Loop1
256  FOR Loop1=1 TO 10
257     READ Weight(Loop1,1,3)
258!    PRINT Weight(Loop1,1,3)
259  NEXT Loop1
260  FOR Loop1=1 TO 11
261     READ Threshold(Loop1,1)
262!    PRINT Threshold(Loop1,1)
263  NEXT Loop1
264  FOR Loop1=1 TO 10
265     READ Threshold(Loop1,2)
266!    PRINT Threshold(Loop1,2)
267  NEXT Loop1
268  READ Threshold(1,3)
```

```
269 FOR Layer=1 TO 3
270 SELECT Layer
271  CASE 1
272     Max_node=11
273     Prev_max_node=571
274  CASE 2
275     Max_node=10
276     Prev_max_node=11
277  CASE 3
278     Max_node=1
279     Prev_max_node=10
280 END SELECT
281 FOR Next_node=1 TO Max_node
284   Sumx=0.
285   FOR Prev_node=1 TO Prev_max_node
286     Sumx= Sumx+Inputx(Prev_node,Layer)*
              Weight(Prev_node,Next_node,Layer)
287   NEXT Prev_node
288   Sumx=Sumx+Threshold(Next_node,Layer)
289   IF Layer<3 THEN
290     Inputx(Next_node,Layer+1)=(1/(1+EXP(-Sumx)))
291   ELSE
292     Inputx(Next_node,Layer+1)=Sumx
293   END IF
294   SELECT Layer
295   CASE 3
296     PRINT USING "10A,D.DDD";"OUTPUT =",Inputx(Next_node,
                                    Layer+1)
297   END SELECT
299 NEXT Next_node
300 NEXT Layer
301 NEXT No_of_out_nodes
302 GOTO Menu2
303 !
304 !
308!
309 ! THIS IS DATA FOR DOING C4C (LEARNED ON C4A)
310 !
311! WEIGHTS FROM 1ST INPUT NODES TO FIRST NODE OF 1ND
LAYER
3          1          2  D          A          T          A
1.7054,1.6862,1.8178,1.656,1.7115,1.5767,1.6291,1.3468,1.5618,1.4923,1.275
6,1.3185,1.3777,1.4325,1.5456,1.52,1.5833,1.5493,-0.86580
3       1       3                    D       A       T       A
1.5812,0.1820,-0.6985,-0.6548,-0.7964,-0.8214,-0.7573,-0.6252,-0.5864,-0.667
,-0.7946,-0.5579,-0.0432,1.5602,1.368,-0.0346,-0.2067
3       1       4                    D       A       T       A
-0.5331,-1.8867,0.2903,0.3879,-1.6108,-2.3090,-1.9934,-4.1147,-3.7425,-1.345
8,-0.5457,0.2033,0.0654,0.0584,0.0998,0.2333,0.1874,0.09160
315 DATA 0.1709,0.0974,0.23860
```

316 ! WEIGHTS FROM 1ST INPUT NODES TO SECOND NODE OF 1ND LAYER
3 1 7 D A T A
2.5158,2.3952,2.3875,2.4159,2.4248,2.4999,2.4166,2.1193,2.2126,2.3709,2.1615,2.1193,1.967,2.2638,2.1276,2.1899,2.0721,1.58628
3 1 8 D A T A
-1.6029,1.8458,0.1827,-0.945,-0.909,-0.8244,-0.9231,-0.9592,-0.8964,-0.8545,-0.7511,-0.8708,-0.6894,0.1577,2.4668,1.9297,-0.457
3 1 9 D A T A
-1.398,-0.979,-2.4885,0.271,0.1042,-2.6323,-3.291,-2.7768,-5.1375,-4.7547,-1.7416,-0.978,-0.0323,-0.0941,-0.0516,0.0489,0.11587
320 DATA 0.1884,0.1211,0.0283,0.1399,0.1733
321 ! WEIGHTS FROM 1ST INPUT NODES TO THIRD NODE OF 1ND LAYER
3 2 2 D A T A
1.2599,1.5166,1.4285,1.7237,0.7171,0.345,0.3048,-0.2053,-0.2489,-0.7161,-0.9294,-1.0911,-1.4691,-1.6038,-1.6178,-0.9569,-0.1923
3 2 3 D A T A
0.7266,-0.7471,1.6313,-0.1382,-1.3494,-1.1822,-1.2035,-1.1812,-1.1775,-1.0249,-0.9898,-0.9562,-1.0977,-1.0489,-0.5762,1.063,1.846
3 2 4 D A T A
2.6369,1.3001,1.6308,0.6172,0.8608,2.4969,-0.0971,-0.6238,0.5247,-1.9494,-1.2785,1.4571,1.8799,2.7754,2.0713,1.8505,1.428,1.2671,1.110
325 DATA 0.9573,0.5478,0.6825,0.4302
326 ! WEIGHTS FROM 1ST INPUT NODES TO FOURTH NODE OF 1ND LAYER
3 2 7 D A T A
0.9649,1.1817,1.2409,1.47,0.5352,0.3692,0.3159,-0.0002,-0.1967,-0.4514,-0.7641,-1.0861,-1.1949,-1.3355,-1.2471,-0.3539,0.9004,2.518
3 2 8 D A T A
2.1994,1.9063,0.0436,-1.0038,-0.9388,-0.8094,-1.0623,-0.8029,-0.9177,-0.8008,-0.716,-0.7886,-0.8073,-0.2346,1.6015,1.6073,2.0828,1.1075
3 2 9 D A T A
1.2788,0.2272,1.1158,2.1758,0.0525,-0.7542,-0.0288,-2.3981,-1.9523,0.2939,0.7205,1.5084,1.2459,0.8794,0.6516,0.6521,0.4521,0.4783
330 DATA 0.175,0.2357,0.1441
331 ! WEIGHTS FROM 1ST INPUT NODES TO FIFTH NODE OF 1ND LAYER
3 3 2 D A T A
1.9901,1.7912,1.8668,1.8065,1.8999,1.7316,1.5983,1.5832,1.548,1.6979,1.5862,1.5071,1.4992,1.5322,1.5582,1.789,1.7337,1.6234,-0.9936
3 3 3 D A T A
1.6702,0.0718,-0.8268,-0.6721,-0.6783,-0.8339,-0.7771,-0.7426,-0.6866,-0.6370,-0.7639,-0.566,-0.0264,1.8716,1.5631,0.0763,-0.3124,-0.6324
3 3 4 D A T A
-2.026,0.1063,0.467,-1.7374,-2.4888,-2.1225,-4.4329,-3.9213,-1.4887,-0.7111,0.0514,0.2415,0.1864,0.0704,0.2061,0.0643,0.1558,0.2082
335 DATA 0.1447,0.2068
336 ! WEIGHTS FROM 1ST INPUT NODES TO SIXTH NODE OF 1ND LAYER

3 3 7 D A T A
1.5819,1.3994,1.5173,1.2064,1.7098,2.0028,1.5771,1.8332,1.9508,2.2222,2.53
38,2.5665,2.6559,2.9872,2.7367,1.5754,-0.3105,-2.7309
3 3 8 D A T A
-6.5742,-0.646,-0.5992,-0.7228,-0.5495,-0.4839,-0.5621,-0.6491,-0.5607,-0.641
6,-0.5584,-0.4884,-0.5631,-0.4482,-0.0655,0.6264,-0.8728
3 3 9 D A T A
-0.7563,-1.0207,-1.7276,-1.1338,-1.1366,-2.2394,-2.5464,-2.3386,-3.7755,-3.55
37,-1.8299,-0.9768,-0.1829,-0.0315,0.0681,-0.0083,0.2607
340 DATA 0.2001,0.1831,0.2386,0.3356,0.5613
341 ! WEIGHTS FROM 1ST INPUT NODES TO SEVENTH NODE OF 1ND
LAYER
3 4 2 D A T A
1.2214,1.5381,1.4747,1.75,0.6924,0.5081,0.4486,0.0486,-0.1136,-0.5959,-0.78
09,-1.0821,-1.3284,-1.6308,-1.5428,-0.5304,0.7299,2.1261
3 4 3 D A T A
1.3985,1.8813,-0.0006,-1.3246,-1.0367,-1.086,-1.1192,-1.0132,-1.0403,-1.0240
,-0.9412,-1.1037,-0.8656,-0.3417,1.5146,1.8954,2.5244
3 4 4 D A T A
1.2452,1.5969,0.7091,1.0204,2.3797,-0.0837,-0.6935,0.0792,-2.2889,-1.8771,0
.5788,1.0701,1.9302,1.4296,1.1358,0.9263,0.8606,0.6705
345 DATA 0.4769,0.3369,0.2451,0.1678
346 ! WEIGHTS FROM 1ST INPUT NODES TO EIGHTH NODE OF 1ND
LAYER
3 4 7 D A T A
1.0236,1.3993,1.3658,1.6092,0.8041,0.4712,0.3225,-0.0785,-0.0848,-0.5058,-0
.793,-1.0316,-1.1897,-1.3756,-1.4596,-0.5494,0.7615,2.3191
3 4 8 D A T A
1.6652,1.8015,-0.0424,-1.111,-0.9029,-0.935,-0.9671,-1.0377,-0.9534,-0.8302,-
0.8324,-0.9532,-0.7524,-0.2031,1.5052,1.7483,2.318,1.22753
3 4 9 D A T A
1.4594,0.4527,1.0866,2.3117,-0.0529,-0.7106,0.0588,-2.2574,-1.7795,0.454,0.
8475,1.6529,1.3089,0.9748,0.8728,0.6793,0.6074,0.4943,0.2523
350 DATA 0.218,0.086
351 ! WEIGHTS FROM 1ST INPUT NODES TO NINETH NODE OF 1ND
LAYER
3 5 2 D A T A
1.2672,1.3552,1.5325,1.6085,0.6318,0.3876,0.3591,-0.0145,-0.2325,-0.4929,-0
.7531,-1.1901,-1.3432,-1.4546,-1.6355,-0.9454,-0.0191,0.9993
3 5 3 D A T A
-0.2964,1.6265,-0.1868,-1.2415,-1.0215,-1.0191,-1.0779,-1.1247,-1.1679,-1.08
23,-1.0482,-1.089,-0.9349,-0.5188,1.2855,1.8188,2.5913
3 5 4 D A T A
1.2689,1.641,0.5635,0.8944,2.4673,-0.1465,-0.5923,0.3818,-1.9583,-1.4859,1.
3146,1.6137,2.432,1.9992,1.6747,1.3028,1.1504,0.8553
355 DATA 0.7551,0.6142,0.4887,0.4092
356 ! WEIGHTS FROM 1ST INPUT NODES TO TENTH NODE OF 1ND
LAYER
3 5 7 D A T A
0.5466,0.7718,0.6676,0.9327,0.3583,0.1768,0.1418,-0.1721,-0.2081,-0.3505,-0

.6532,-0.8608,-0.9692,-0.8972,-0.8145,0.027,1.0047,2.30349
3 5 8 D A T A
2.1322,1.6969,0.2209,-0.7477,-0.7777,-0.6731,-0.6739,-0.7427,-0.725,-0.5871,
-0.6055,-0.5663,-0.5534,-0.213,1.3218,1.2017,1.3019,0.76643
3 5 9 D A T A
0.6119,-0.2955,1.1374,1.7687,0.2226,-0.4985,-0.2279,-2.2104,-1.7272,-0.1902,
0.4874,1.079,0.8319,0.6921,0.4226,0.4785,0.4155,0.28257
360 DATA 0.0433,0.0525,-0.0206
361 ! WEIGHTS FROM 1ST INPUT TO ELEVENTH NODE OF 1ND LAYER
3 6 2 D A T A
0.5865,0.5576,0.6355,0.8511,0.3259,0.1207,0.1536,-0.0052,-0.0984,-0.3556,-0
.454,-0.6591,-0.7156,-0.6406,-0.7134,-0.0677,0.7952,1.8213
3 6 3 D A T A
1.6868,1.4035,0.298,-0.5292,-0.5705,-0.5756,-0.5439,-0.6292,-0.5922,-0.4809,
-0.583,-0.668,-0.4791,-0.1398,0.9935,1.1301,0.9148,0.5805
3 6 4 D A T A
0.4682,-0.3096,1.1135,1.5773,0.3575,-0.3853,-0.439,-2.0493,-1.7594,-0.2621,0
.31,0.8143,0.6283,0.6007,0.3624,0.2647,0.227,0.3153,0.13563
365 DATA 0.0985,0.0927
366 ! WEIGHTS FROM 1ST LAYER NODES TO FIRST NODE OF 2ND LAYER
3 6 7 D A T A
-0.4260,-0.8936,-1.2925,-0.7822,-0.3524,1.4944,-0.9189,-0.8249,-1.2242,-0.64
65,-0.3090
368 ! WEIGHTS FROM 1ST LAYER NODES TO SECOND NODE OF 2ND LAYER
3 6 9 D A T A
-0.3681,-0.5297,-1.3782,-0.6117,-0.3637,1.2812,-0.7503,-0.5911,-1.0642,-0.42
26,-0.2374
370 ! WEIGHTS FROM 1ST LAYER NODES TO THIRD NODE OF 2ND LAYER
3 7 1 D A T A
-0.3703,-0.5028,-1.2933,-0.5446,-0.4311,1.2952,-0.7872,-0.5846,-1.1442,-0.42
56,-0.403
372 ! WEIGHTS FROM 1ST LAYER NODES TO FOURTH NODE OF 2ND LAYER
3 7 3 D A T A
-0.3571,-0.4842,-1.3277,-0.5512,-0.3569,1.2426,-0.7175,-0.5436,-1.105,-0.461
4,-0.4473
374 ! WEIGHTS FROM 1ST LAYER NODES TO FIFTH NODE OF 2ND LAYER
3 7 5 D A T A
-0.386,-0.6443,-1.3222,-0.606,-0.4141,1.3604,-0.7894,-0.6911,-1.2168,-0.3903,
-0.28053
376 ! WEIGHTS FROM 1ST LAYER NODES TO SIXTH NODE OF 2ND LAYER
3 7 7 D A T A
-0.2893,-0.747,-1.2615,-0.5905,-0.3922,1.3591,-0.7614,-0.713,-1.2423,-0.4798,
-0.43833

378 ! WEIGHTS FROM 1ST LAYER NODES TO SEVENTH NODE OF 2ND LAYER
3 7 9 D A T A
-0.273,-0.1772,-1.3838,-0.3308,-0.4379,1.0309,-0.64,-0.5318,-1.2122,-0.2999,-0.5106
380 ! WEIGHTS FROM 1ST LAYER NODES TO EIGHTH NODE OF 2ND LAYER
3 8 1 D A T A
-0.268,-0.9124,-1.3525,-0.7842,-0.3672,1.4543,-0.8172,-0.8422,-1.205,-0.4682,-0.37493
382 ! WEIGHTS FROM 1ST LAYER NODES TO NINETH NODE OF 2ND LAYER
3 8 3 D A T A
-0.3792,-0.7339,-1.2309,-0.6663,-0.4636,1.4558,-0.8864,-0.838,-1.148,-0.5701,-0.39593
384 ! WEIGHTS FROM 1ST LAYER NODES TO TENTH NODE OF 2ND LAYER
3 8 5 D A T A
-0.3314,0.106,-1.4582,-0.3749,-0.3677,0.7514,-0.5944,-0.3731,-1.1957,-0.3436,-0.49773
386 ! WEIGHTS FROM 2ND LAYER NODES TO FIRST NODE OF 3ND LAYER
3 8 7 D A T A
2.1717,2.0299,2.042,2.011,2.0901,2.0886,1.9205,2.1465,2.1407,1.8425
388 ! THRESHOLDS OF 1ST LAYER
3 8 9 D A T A
-5.7925,-7.0155,-9.749,-7.4551,-6.0775,-5.468,-8.7614,-8.1127,-9.4196,-5.3607,-4.6895
390 ! THRESHOLDS OF 2ND LAYER
3 9 1 D A T A
-2.2662,-2.4902,-2.4485,-2.4863,-2.4021,-2.3821,-2.6299,-2.3498,-2.3166,-2.729-
392 ! THRESHOLDS OF 3ND LAYER
393 DATA 0.0393
394 !
395 ! THIS IS DATA FOR DOING C4A (LEARNED ON C4C)
396 !
397 ! WEIGHTS FROM INPUT LAYER TO FIRST NODE OF 1ST LAYER
3 9 8 D A T A
-0.5326,-0.4641,-0.3858,-0.5713,-0.5249,-0.5759,-0.519,-0.6862,-0.5337,-0.7065,-0.7948,-0.7885,-0.7038,-0.9384,-0.8808,-1.1824,-1.349
3 9 9 D A T A
-2.2014,-2.2053,0.7106,1.0498,0.4152,0.1029,0.1971,0.1146,-0.0151,0.073,0.0914,-0.0054,-0.03,0.1736,-0.3038,-1.3411,-0.5053,1.7871,0.5646
4 0 0 D A T A
-0.7283,2.5025,1.9539,2.8437,-1.0475,-0.1296,1.6436,2.2876,1.5952,-0.2911,-0.8008,-0.3687,-0.1418,0.3828,0.3909,0.326,0.4262,0.5074
401 DATA 0.5397,0.3043,0.62429
402 WEIGHTS FROM INPUT LAYER TO SECOND NODE OF 1ST LAYER.
4 0 3 D A T A
-0.0156,-0.0662,-0.1083,-0.0965,-0.1372,0.0124,-0.0442,-0.2562,-0.2206,-0.16

09,-0.2517,-0.3201,-0.4272,-0.3864,-0.4978,-0.5273,-0.61276
4 0 4 D A T A
-1.1385,-0.0751,-0.5378,-0.281,-0.2515,-0.0751,0.0345,-0.0467,-0.19,-0.1683,-0.1608,-0.0845,-0.0645,0.0709,0.2902,0.2282,0.5861,0.33836
4 0 5 D A T A
-1.071,-2.338,-1.7791,-0.3594,0.0113,-0.3735,-0.2001,0.8384,1.4309,1.1215,-0.1026,-0.4446,-0.0423,0.0379,0.4695,0.5496,0.4504,0.67798
406 DATA 0.7637,0.6533,0.5304,0.8088
407 WEIGHTS FROM INPUT LAYER TO THIRD NODE OF 1ST LAYER
4 0 8 D A T A
-0.4098,-0.2758,-0.3983,-0.3596,-0.3766,-0.4773,-0.4199,-0.5553,-0.4596,-0.6499,-0.5813,-0.5456,-0.7233,-0.7536,-0.7135,-0.8807
4 0 9 D A T A
-1.0333,-1.7112,-1.4021,0.3085,0.4775,-0.0348,-0.0642,0.0594,0.0713,-0.1167,0.0081,0.0274,0.0175,0.0117,0.0694,-0.1716,-0.719,0.01738
4 1 0 D A T A
1.5695,0.1179,-0.8575,1.2039,0.6631,1.5948,-0.8081,0.1849,1.5235,1.8821,1.4287,-0.1598,-0.5391,-0.1113,-0.0022,0.5076,0.3472,0.2551
411 DATA 0.4374,0.5731,0.3794,0.4081,0.48361
412! WEIGHTS FROM INPUT LAYER TO FOURTH NODE OF 1ST LAYER8
4 1 3 D A T A
0.2749,0.3397,0.3018,0.2705,0.0529,0.2349,0.1193,0.1067,0.0316,0.0139,-0.0646,-0.2138,-0.2066,-0.2887,-0.2646,-0.4372,-0.7195,-1.3677
4 1 4 D A T A
0.278,-0.7011,-0.6183,-0.4273,-0.0564,0.1107,-0.143,-0.0468,-0.2005,-0.1021,-0.0611,0.046,0.1277,0.4614,0.697,0.5526,0.1028,-1.4833,-3.1431
4 1 5 D A T A
-2.7204,-0.7787,-0.3658,-0.5334,-0.9808,0.908,1.4979,1.245,-0.4564,-0.9148,-0.4047,-0.0301,0.5188,0.5333,0.5361,0.7638,1.233,0.9995
416 DATA 0.7796,1.3581
417! WEIGHTS FROM INPUT LAYER TO FIFTH NODE OF 1ST LAYER
4 1 8 D A T A
0.1698,0.0275,0.0619,-0.0168,0.043,-0.0342,-0.1489,-0.0825,-0.1714,-0.093,-0.116,-0.2271,-0.2142,-0.4397,-0.4385,-0.3944,-0.6056,-1.13214
4 1 9 D A T A
-0.028,-0.5537,-0.4796,-0.3545,0.0567,0.0923,-0.0745,-0.1305,-0.142,-0.1166,-0.0912,-0.0664,0.1726,0.3232,0.4199,0.6243,0.3158,-1.28942
4 2 0 D A T A
-2.7291,-2.0121,-0.6189,-0.034,-0.445,-0.4828,0.9135,1.5243,1.231,-0.348,-0.6433,-0.3037,0.1298,0.6035,0.5145,0.4979,0.5783,0.90692
421 DATA 0.9259,0.6028,1.03081
422! WEIGHTS FROM INPUT LAYER TO SIXTH NODE OF 1ST LAYER
4 2 3 D A T A
-0.0875,-0.0378,0.0738,-0.0998,-0.0653,0.0055,-0.2018,-0.1212,-0.1955,-0.1827,-0.1806,-0.3219,-0.3358,-0.4082,-0.4222,-0.5398,-0.5209
4 2 4 D A T A
-0.9498,-0.102,-0.4682,-0.2871,-0.3704,-0.012,0.0743,-0.0329,-0.1413,-0.0679,-0.1646,-0.0854,0.0383,0.1307,0.2465,0.2097,0.4868,0.13582
4 2 5 D A T A
-1.0954,-2.31,-1.5477,-0.2604,0.1336,-0.3097,-0.225,0.6996,1.1924,0.9046,-0.2

234,-0.4206,-0.1335,0.1281,0.5281,0.447,0.4547,0.55341
426 DATA 0.6782,0.6076,0.5109,0.8672
427! WEIGHTS FROM INPUT LAYER TO SEVENTH NODE OF 1ST LAYER
4 2 8 D A T A
0.1572,0.2916,0.1445,0.1172,-0.0388,0.1299,0.0561,0.0025,-0.0226,-0.2077,-0
.1438,-0.2164,-0.297,-0.4696,-0.371,-0.4143,-0.5833,-1.25092
4 2 9 D A T A
0.143,-0.5725,-0.4267,-0.51,-0.019,-0.0251,-0.0468,-0.1058,-0.1676,-0.1701,-0.
1363,-0.1242,0.1819,0.3923,0.5006,0.6726,0.1198,-1.4248
4 3 0 D A T A
-2.9161,-2.287,-0.6503,-0.2219,-0.5227,-0.6062,0.9047,1.5765,1.2941,-0.3935,
-0.7445,-0.272,-0.0844,0.5016,0.5601,0.5352,0.7701,0.9858
431 DATA 0.9752,0.6361,1.16750
432! WEIGHTS FROM INPUT LAYER TO EIGHTH NODE OF 1ST LAYER,
4 3 3 D A T A
-0.3254,-0.1115,-0.2156,-0.2542,-0.2014,-0.1991,-0.3525,-0.4048,-0.2809,-0.39
67,-0.3986,-0.3974,-0.368,-0.3959,-0.4309,-0.4571,-0.36882
4 3 4 D A T A
-0.3643,-0.181,0.341,0.3785,0.0768,-0.0159,0.1037,0.0913,-0.1661,-0.1132,0.0
093,-0.0350,-0.0734,0.0306,-0.3257,-0.8837,0.3807,1.52818
4 3 5 D A T A
0.2107,-0.6054,1.0688,1.967,1.7485,0.195,0.1963,0.9246,1.0195,0.9631,0.344
7,0.1201,0.3338,0.4218,0.495,0.5844,0.4614,0.5894,0.65298
436 DATA 0.5654,0.4654,0.56558
437! WEIGHTS FROM INPUT LAYER TO NINETH NODE OF 1ST LAYER3
4 3 8 D A T A
0.4098,0.3752,0.4829,0.2944,0.1844,0.2507,0.2499,0.1964,0.1135,0.0875,0.05
09,-0.2053,-0.1966,-0.286,-0.4151,-0.5957,-0.8385,-1.8335
4 3 9 D A T A
0.3622,-0.7584,-0.6493,-0.5039,0.0924,0.1294,0.0543,-0.1606,-0.2418,-0.1786,
-0.1906,-0.0166,0.3264,0.6636,1.0511,0.9389,0.2585,-1.77132
4 4 0 D A T A
-3.6126,-3.2705,-0.8187,-0.5098,-0.9113,-1.3164,1.1003,2.1909,1.7122,-0.7046
,-1.3023,-0.68,-0.1774,0.7294,0.7107,0.5689,0.8784,1.3602
441 DATA 1.3411,0.8554,1.64651
442! WEIGHTS FROM INPUT LAYER TO TENTH NODE OF 1ST LAYER
4 4 3 D A T A
0.1671,0.3047,0.1039,0.1909,0.0772,0.1578,0.0661,-0.0354,0.0021,-0.0044,-0.
1382,-0.2716,-0.3226,-0.2841,-0.3311,-0.3851,-0.7302,-1.4478
4 4 4 D A T A
0.0452,-0.6138,-0.5008,-0.449,-0.1111,0.0359,0.0122,-0.2012,-0.222,-0.1018,-
0.1507,0.054,0.2,0.399,0.6769,0.5519,0.1469,-1.3691,-3.1044
4 4 5 D A T A
-2.516,-0.6089,-0.262,-0.5085,-0.7724,0.9636,1.5773,1.3221,-0.5286,-0.7090,-
0.2781,-0.0226,0.644,0.5462,0.5618,0.8575,1.1261,0.8955 446 DATA
0.6466,1.1687
447! WEIGHTS FROM INPUT LAYER TO ELEVENTH NODE OF 1ST
LAYER
4 4 8 D A T A
0.3928,0.3108,0.295,0.3698,0.1567,0.1686,0.1451,0.1413,0.0938,-0.085,-0.063

4,-0.2354,-0.2494,-0.25,-0.439,-0.5266,-0.7649,-1.5522
4 4 9 D A T A
0.2352,-0.6836,-0.4494,-0.3954,0.01,0.0473,0.0402,-0.1804,-0.1806,-0.0873,-0.
2113,-0.1296,0.246,0.5765,0.768,0.7136,0.0733,-1.574
4 5 0 D A T A
-3.3648,-2.8066,-0.6694,-0.3297,-0.5991,-1.0349,0.8207,1.8019,1.3402,-0.5662
,-0.8967,-0.462,-0.1508,0.693,0.6144,0.4403,0.7863,1.3153
451 DATA 1.0977,0.7681,1.42266>452
WEIGHTS FROM 1ND LAYER NODES TO FIRST NODE OF 2ND LAYER
4 5 3 D A T A
-0.6997,-0.7052,-0.53,-0.8284,-0.6257,-0.4957,-0.7538,-0.9141,-1.2098,-0.8673
,-0.8674
454! WEIGHTS FROM 1ST LAYER NODES TO SECOND NODE OF 2ND
LAYER
4 5 5 D A T A
-0.6751,-0.5507,-0.6291,-0.9774,-0.657,-0.6176,-0.8019,-0.9977,-1.0779,-0.839
9,-0.841
456! WEIGHTS FROM 1ST LAYER NODES TO THIRD NODE OF 2ND
LAYER
4 5 7 D A T A
-0.6516,-0.547,-0.5104,-0.8806,-0.7,-0.5273,-0.812,-0.9887,-1.1175,-0.8106,-0.
95148
458! WEIGHTS FROM 1ST LAYER NODES TO FOURTH NODE OF 2ND
LAYER
4 5 9 D A T A
-0.6236,-0.6264,-0.5007,-0.9323,-0.6357,-0.5308,-0.758,-0.9689,-1.0525,-0.859
5,-0.97620
460! WEIGHTS FROM 1ST LAYER NODES TO FIFTH NODE OF 2ND
LAYER
4 6 1 D A T A
-0.6901,-0.601,-0.5791,-0.8536,-0.7098,-0.5652,-0.7668,-0.985,-1.2262,-0.7416
,-0.8723
462! WEIGHTS FROM 1ST LAYER NODES TO SIXTH NODE OF 2ND
LAYER
4 6 3 D A T A
-0.55,-0.672,-0.4992,-0.8046,-0.6347,-0.5274,-0.7132,-1.0383,-1.2134,-0.7899,-
0.97583
464! WEIGHTS FROM 1ST LAYER NODES TO SEVENTH NODE OF 2ND
LAYER
4 6 5 D A T A
-0.5554,-0.6472,-0.4716,-0.8255,-0.7974,-0.5622,-0.7302,-1.0351,-1.1127,-0.73
47,-0.9875
466! WEIGHTS FROM 1ST LAYER NODES TO EIGHTH NODE OF 2ND
LAYER
4 6 7 D A T A
-0.5508,-0.6301,-0.6329,-0.8615,-0.6152,-0.6101,-0.6812,-1.0152,-1.1999,-0.72
21,-0.9752
468! WEIGHTS FROM 1ST LAYER NODES TO NINETH NODE OF 2ND
LAYER
4 6 9 D A T A

-0.5986,-0.5131,-0.4663,-0.8055,-0.7053,-0.5212,-0.793,-1.0249,-1.1406,-0.857
3,-0.97052
470! WEIGHTS FROM 1ST LAYER NODES TO TENTH NODE OF 2ND
LAYER
4 7 1 D A T A
-0.6881,-0.5584,-0.5557,-0.9544,-0.7613,-0.6177,-0.7256,-0.9783,-1.0858,-0.79
11,-0.9124
472! WEIGHTS FROM 2ND LAYER NODES TO FIRST NODE OF 3RD
LAYER
4 7 3 D A T A
1.4987,1.4891,1.4934,1.4942,1.4948,1.4988,1.495,1.5009,1.4998,1.48758
474! THRESHOLDS OF 1ST LAYER
4 7 5 D A T A
0.1063,-1.1184,-0.7053,-1.4004,-1.247,-1.1718,-1.2812,-0.733,-1.4439,-1.2136,
-1.32971
476! THRESHOLDS OF 2ND LAYER
4 7 7 D A T A
-1.592,-1.4998,-1.5475,-1.5828,-1.5153,-1.5493,-1.5548,-1.5442,-1.554,-1.5244
478! THRESHOLDS OF 3RD LAYER
479 DATA -0.1369
480!
481! THIS IS DATA FOR DOING C7C (LEARNED ON C7A)
482 !
483! WEIGHTS FROM INPUT NODES TO FIRST NODE OF 1ST LAYER
4 8 4 D A T A
-2.4549,-2.3229,-1.7672,-2.0566,-0.7999,-0.071,0.0644,0.8988,1.0743,1.5834,1
.7837,1.97,1.9369,1.8503,1.6613,0.8948,0.0603,-0.2645
4 8 5 D A T A
-1.4781,-2.2384,-1.7873,-1.3837,-0.6214,-0.3756,-0.457,-0.7985,-0.6755,-0.889
5,-0.7821,-0.7362,-0.4108,0.0223,0.357,0.7908,1.2379
4 8 6 D A T A
1.3089,0.8573,0.9444,0.9907,1.7586,0.4414,-0.2865,0.4228,2.1211,0.6179,0.2
744,-0.4988,-1.827,-0.4668,-0.1557,0.1135,0.0363,0.4529,0.76554
487 DATA 0.8196,1.0777,1.3797
488! WEIGHTS FROM INPUT NODES TO SECOND NODE OF 1ST LAYER
4 8 9 D A T A
-2.0541,-2.106,-1.5474,-1.8677,-0.7691,-0.369,-0.1108,0.4884,0.655,1.1395,1.2
752,1.3716,1.3908,1.5975,1.267,0.6413,-0.1168,-0.58480
4 9 0 D A T A
-1.4955,-1.6476,-1.147,-0.7323,-0.3485,-0.0558,-0.2692,-0.4032,-0.46,-0.3872,-
0.3747,-0.4,-0.0822,0.2756,0.5947,0.7053,0.9488,0.83967
4 9 1 D A T A
0.2414,0.0677,-0.0524,0.3022,-0.2929,-0.533,0.4376,2.0498,0.8888,0.4058,-0.
4799,-1.5975,-0.379,-0.0126,0.1803,0.2904,0.385,0.6175,0.888
492 DATA 0.9841,1.1557
493! WEIGHTS FROM INPUT NODES TO THIRD NODE OF 1ST LAYER
4 9 4 D A T A
-2.2205,-2.177,-1.6166,-1.8605,-0.9124,-0.1141,-0.0908,0.5961,0.7769,1.4067,
1.479,1.6885,1.6628,1.7774,1.5696,0.8319,0.1908,-0.0558
4 9 5 D A T A

-1.1411,-2.1293,-1.6146,-1.3761,-0.5961,-0.386,-0.6235,-0.7741,-0.8038,-0.798
1,-0.784,-0.7281,-0.4498,-0.0232,0.2436,0.6066,1.10098
4 9 6 D A T A
1.2828,1.0072,1.169,1.1696,1.9512,0.5972,-0.1957,0.607,2.1647,0.605,0.3949,
-0.3808,-1.7226,-0.4611,-0.1145,0.0666,0.1752,0.4665,0.62588
497 DATA 0.8465,1.0651,1.49331
498! WEIGHTS FROM INPUT NODES TO FOURTH NODE OF 1ST LAYER
4 9 9 D A T A
-2.3365,-2.1853,-1.7966,-2.0085,-0.7929,-0.2523,0.0974,0.7532,0.822,1.2038,1
.5779,1.5976,1.6935,1.6399,1.3763,0.5414,-0.3404,-1.18315
5 0 0 D A T A
-2.1653,-1.492,-0.8904,-0.6549,-0.1681,0.188,0.0385,-0.3335,-0.3476,-0.4173,-
0.2531,-0.0991,0.1081,0.45,0.82,1.0522,1.0021,0.7233
5 0 1 D A T A
-0.1828,-0.5565,-0.6757,-0.478,-0.7705,-0.6349,0.4558,2.1462,0.83,0.4568,-0.5
369,-1.5014,-0.4109,-0.1177,0.0812,0.2625,0.4183,0.58665
502 DATA 0.9168,0.8719,1.15527
503! WEIGHTS FROM INPUT NODES TO FIFTH NODE OF 1ST LAYER
5 0 4 D A T A
-2.3222,-2.1726,-1.7156,-2.0303,-0.8267,-0.1403,0.0439,0.6137,0.9064,1.3188,
1.6533,1.6931,1.7687,1.7163,1.3851,0.8667,-0.0694,-0.3665
5 0 5 D A T A
-1.5029,-1.962,-1.4459,-1.1398,-0.4466,-0.2435,-0.432,-0.585,-0.6589,-0.6034,-
0.6864,-0.5277,-0.3235,0.0679,0.3627,0.8292,1.1703,1.1352
5 0 6 D A T A
0.6329,0.6329,0.5708,1.0458,0.0594,-0.2808,0.589,2.3975,0.8521,0.5238,-0.50
27,-1.6903,-0.4719,-0.1352,0.0368,0.1447,0.4171,0.7336
507 DATA 0.8309,0.9825,1.3243
508! WEIGHTS FROM INPUT NODES TO SIXTH NODE OF 1ST LAYER
5 0 9 D A T A
-2.0594,-2.0273,-1.6004,-1.7756,-0.9104,-0.2596,-0.1565,0.4882,0.7258,1.0906
,1.227,1.3038,1.3378,1.4895,1.2035,0.7819,0.0554,0.1624
5 1 0 D A T A
-0.8118,-2.056,-1.5817,-1.1657,-0.5425,-0.2208,-0.5213,-0.7342,-0.7013,-0.722
6,-0.7162,-0.5851,-0.4758,-0.0239,0.1489,0.7276,1.0109
5 1 1 D A T A
1.0914,0.8567,0.8182,0.9086,1.8964,0.4671,-0.1147,0.7187,2.3509,0.8245,0.4
296,-0.3751,-1.7182,-0.3116,0.014,0.034,0.1347,0.4271,0.82452
512 DATA 0.8845,1.0677,1.4463
513! WEIGHTS FROM INPUT NODES TO SEVENTH NODE OF 1ST
LAYER
5 1 4 D A T A
-2.3966,-2.2665,-1.8559,-2.0819,-0.7752,-0.336,0.029,0.7857,0.9069,1.4918,1.
696,1.8527,1.7349,1.7966,1.5595,0.7518,-0.4661,-1.44398
5 1 5 D A T A
-2.5469,-1.5388,-0.9324,-0.6174,-0.191,0.1977,-0.0891,-0.1604,-0.2451,-0.344
9,-0.2647,-0.1701,0.1665,0.5167,0.9031,1.1728,1.1858,0.7324
5 1 6 D A T A
-0.1584,-0.621,-0.8985,-0.6585,-0.8554,-0.7915,0.3019,2.1923,0.8148,0.3069,-
0.6116,-1.5131,-0.4926,-0.1195,0.1761,0.2013,0.4422,0.73144 517 DATA

0.8374,0.961,1.2423
518! WEIGHTS FOM INPUT NODES TO EIGHTH NODE OF 1ST LAYER
5 1 9 D A T A
-2.27,-2.1913,-1.7951,-1.8872,-0.7494,-0.3147,0.0712,0.5673,0.785,1.245,1.36
31,1.5176,1.656,1.6464,1.4321,0.7296,-0.2586,-0.9207,-1.726
5 2 0 D A T A
-1.7326,-1.077,-0.8729,-0.3666,0.1304,-0.2524,-0.2846,-0.4304,-0.5409,-0.488
3,-0.2885,-0.0793,0.3347,0.6603,0.9439,1.0866,0.862,0.06776
5 2 1 D A T A
-0.1728,-0.287,0.0996,-0.3965,-0.5418,0.5162,2.1123,0.8764,0.3014,-0.4558,-1
.6334,-0.3393,-0.1511,-0.0012,0.2441,0.5583,0.7366,0.8619^Y522 DATA
0.975,1.16158
523! WEIGHTS FROM INPUT NODES TO NINETH NODE OF 1ST LAYER.
5 2 4 D A T A
-2.0763,-1.9956,-1.5501,-1.8702,-0.8286,-0.3852,-0.1505,0.5199,0.7156,0.984,
1.2007,1.3342,1.3994,1.4234,1.0896,0.4892,-0.3623,-0.8381
5 2 5 D A T A
-1.5269,-1.3491,-0.826,-0.5802,-0.1197,0.0171,-0.0419,-0.198,-0.3489,-0.4277,
-0.3102,-0.1332,0.0733,0.3651,0.5798,0.7989,0.9625,0.52996
5 2 6 D A T A
-0.1982,-0.5123,-0.6372,-0.2606,-0.5259,-0.5704,0.5704,2.0671,0.9193,0.4963,
-0.5671,-1.4984,-0.4451,0.0318,0.1229,0.2422,0.4742,0.70396
527 DATA 0.6845,0.8876,1.16853
528! WEIGHTS FROM INPUT NODES TO TENTH NODE OF 1ST LAYER
5 2 9 D A T A
-2.3691,-2.4287,-1.6102,-1.9907,-0.9324,0.1755,0.1388,0.9689,1.1364,2.021,2.
1585,2.2389,2.0843,2.2313,2.0233,1.4583,1.0039,1.1644
5 3 0 D A T A
-0.6229,-2.9251,-2.4377,-2.1163,-1.1042,-0.7743,-0.9173,-1.2808,-1.2675,-1.34
53,-1.3317,-1.1432,-0.9855,-0.5851,-0.075,0.7128,1.75313
5 3 1 D A T A
2.5853,2.6601,3.6818,3.3754,4.8412,1.7359,0.0584,0.227,1.5901,-0.04047,-0.0
754,-0.6222,-2.2901,-0.7356,-0.316,-0.2509,-0.1646,0.23133(532 DATA
0.6245,0.7099,1.0856,1.5502
533! WEIGHTS FROM INPUT NODES TO ELEVENTH NODE OF 1ST LAYER
5 3 4 D A T A
-2.049,-2.0172,-1.5801,-1.9052,-0.8557,-0.3014,-0.1578,0.4403,0.7366,1.0346,
1.3028,1.5043,1.4334,1.3765,1.2798,0.7018,-0.2366,-0.6496
5 3 5 D A T A
-1.4868,-1.5719,-1.0311,-0.717,-0.3585,0.0755,-0.1657,-0.3153,-0.3121,-0.534
7,-0.3902,-0.2527,-0.0415,0.2946,0.4348,0.7717,1.0438
5 3 6 D A T A
0.7211,0.0378,-0.1041,-0.1625,0.2107,-0.2577,-0.4444,0.4441,2.0777,0.8093,0
.4254,-0.5664,-1.5171,-0.2804,-0.0315,0.0678,0.1856,0.4705(537 DATA
0.7647,0.8119,0.9431,1.13699
538! WEIGHTS FROM 1ST LAYER NODES TO FIRST OF 2ND LAYER
5 3 9 D A T A
-0.8475,-0.585,-0.731,-0.6812,-0.6928,-0.7487,-0.8038,-0.6847,-0.689,-1.8663,-
0.55615

```
540! WEIGHTS FROM 1ST LAYER NODES TO SECOND OF 2ND LAYER
5          4          1          D          A          T          A
-0.8412,-0.7079,-0.882,-0.7897,-0.5838,-0.6889,-0.8388,-0.6237,-0.6062,-
.8384,-0.61299
542! WEIGHTS FROM 1ST LAYER NODES TO THIRD OF 2ND LAYER
5          4          3          D          A          T          A
-0.9104,-0.685,-0.8591,-0.6715,-0.7306,-0.6571,-0.7861,-0.7696,-0.6587,-1.766
9,-0.61166
544! WEIGHTS FROM 1ST LAYER NODES TO FOURTH OF 2ND LAYER
5          4          5          D          A          T          A
-0.8436,-0.5366,-0.7936,-0.6537,-0.674,-0.7443,-0.6561,-0.7003,-0.6281,-1.851
1,-0.542969
546! WEIGHTS FROM 1ST LAYER NODES TO FIFTH OF 2ND LAYER
5          4          7          D          A          T          A
-0.8472,-0.5668,-0.7739,-0.6989,-0.7306,-0.818,-0.6649,-0.6287,-0.5587,-1.801
4,-0.69
548! WEIGHTS FROM 1ST LAYER NODES TO SIXTH OF 2ND LAYER
5          4          9          D          A          T          A
-0.8641,-0.4888,-0.8474,-0.5939,-0.6401,-0.6992,-0.6494,-0.5855,-0.6507,-1.84
76,-0.69365
550! WEIGHTS FROM 1ST LAYER TO SEVENTH OF 2ND LAYER
5          5          1          D          A          T          A
-0.7469,-0.7109,-0.8796,-0.7143,-0.624,-0.6439,-0.7248,-0.7509,-0.6212,-1.871
,-0.67454
552! WEIGHTS FROM 1ST LAYER TO EIGHTH OF 2ND LAYER
5          5          3          D          A          T          A
-0.7396,-0.5717,-0.8012,-0.7935,-0.6616,-0.7657,-0.7144,-0.6518,-0.6867,-1.88
72,-0.58044
554! WEIGHTS FROM 1ST LAYER TO NINETH OF 2ND LAYER
5          5          5          D          A          T          A
-0.9384,-0.6431,-0.741,-0.648,-0.6272,-0.704,-0.7319,-0.7290,-0.6116,-1.8253,-
0.732283
556! WEIGHTS FROM 1ST LAYER TO TENTH OF 2ND LAYER
5          5          7          D          A          T          A
-0.7544,-0.7124,-0.7308,-0.7603,-0.7567,-0.7526,-0.7934,-0.6061,-0.6788,-1.86
65,-0.65493
558! WEIGHTS FROM 2ND LAYER TO FIRST OF 3RD LAYER
5          5          9          D          A          T          A
2.6083,2.6096,2.603,2.6003,2.5978,2.5988,2.6103,2.6091,2.605,2.6115
560! THRESHOLDS FOR 1ST LAYER
5          6          1          D          A          T          A
-7.0838,-4.0887,-7.3075,-2.9892,-5.9418,-6.7417,-2.8443,-3.8748,-2.9295,-11.9
581,-3.8993
562! THRESHOLDS FOR 2ND LAYER
5          6          3          D          A          T          A
-3.0054,-2.9972,-2.9975,-3.0242,-3.0239,-3.0283,-3.0007,-3.0058,-3.0059,-2.99
45
564! THESHOLDS FOR 2RD LAYER
565     DATA -0.2864
566!
```

```
567! THIS IS DATA FOR DOING C7A (LEARNED ON C7C)
568!
569! WEIGHTS FROM INPUT NODES TO FIRST OF 1ST LAYER
5          7          0          D          A          T          A
0.0245,-0.012,0.143,-0.0152,0.0779,-0.0341,0.0989,-0.0829,0.1114,0.1167,-0.1
021,-0.0673,0.129,-0.0702,-0.021,-0.05,0.1476,0.8239,1.1887
5          7          1          D          A          T          A
-0.7831,0.1762,0.3917,-0.2555,-0.3097,-0.2127,-0.1903,-0.1175,-0.0993,-0.223
3,-0.3718,-0.6457,-1.3497,-2.8493,-3.3415,-0.6554,0.03848
5          7          2          D          A          T          A
-0.6746,-0.3119,0.1448,1.2777,2.0556,0.1991,-0.3452,-0.1753,-0.9051,-1.2434,
-0.6402,-0.2249,-0.1245,-0.2233,-0.1151,0.2727,0.3044
573     DATA 0.2578,0.47,0.2784,0.63436
574! WEIGHTS FROM INPUT NODES TO SECOND OF 1ST LAYER
5          7          5          D          A          T          A
-0.0125,-0.109,-0.1087,-0.0686,-0.0506,0.0553,0.0524,-0.1369,-0.0861,0.0839,
-0.0652,-0.1006,-0.1497,-0.0711,-0.1997,-0.0647,0.07538
5          7          6          D          A          T          A
0.3684,0.657,-0.5451,0.2609,0.4094,-0.1704,-0.087,-0.0407,-0.1083,-0.0912,-0.
0775,-0.0237,-0.1318,-0.4361,-0.7741,-1.7463,-2.1906
5          7          7          D          A          T          A
-0.6078,0.4031,-0.381,-0.3545,0.096,0.8345,1.1991,0.0125,-0.3962,-0.2076,-0.
5859,-0.8013,-0.7454,-0.4805,-0.441,-0.4812,-0.34358
578     DATA -0.1361,0.0393,-0.0146,0.0696,0.0957,0.22235
579! WEIGHTS FROM INPUT NODES TO THIRD OF 1ST LAYER2
5          8          0          D          A          T          A
0.0434,0.1005,0.0405,0.996,0.115,-0.041,0.0779,-0.0652,0.0678,0.0156,-0.032
,0.0352,-0.0309,-0.0748,-0.04,0.0263,0.1655,0.7185,1.22658
5          8          1          D          A          T          A
-0.7209,0.1365,0.1842,-0.4024,-0.3423,-0.1730,-0.2489,-0.1409,-0.1178,-0.150
4,-0.2922,-0.7206,-1.4163,-2.7997,-3.0903,-0.5363,-0.2007
5          8          2          D          A          T          A
-0.6491,-0.3695,0.2329,1.4138,2.0489,0.2162,-0.2292,-0.2076,-0.784,-1.2382,-
0.6269,-0.1089,-0.0487,-0.0454,-0.1215,0.1833,0.3279,0.3701
583     DATA 0.3287,0.3766,0.527326
584! WEIGHTS FROM INPUT NODES TO FOURTH OF 1ST LAYER
5          8          5          D          A          T          A
0.092,0.0837,0.1127,0.1066,-0.0154,0.093,0.0542,0.0727,0.0304,0.1647,0.021
7,-0.0942,0.0406,-0.0086,-0.0109,0.0158,0.096,0.7265,1.1663
5          8          6          D          A          T          A
-0.7372,0.0704,0.3224,-0.347,-0.1379,-0.2465,-0.0758,-0.2385,-0.1296,-0.1105
,-0.1895,-0.7081,-1.319,-2.5585,-3.2702,-0.5217,0.09696
5          8          7          D          A          T          A
-0.55,-0.3968,-0.0083,1.19,1.9717,0.0667,-0.2771,-0.3436,-0.8617,-1.1017,-0.7
612,-0.323,-0.048,-0.2612,-0.2368,0.1745,0.1862,0.3786
588     DATA 0.3408,0.3319,0.595435
589! WEIGHTS FROM INPUT NODES TO FIFTH OF 1ST LAYER,
5          9          0          D          A          T          A
-0.0386,-0.2248,-0.1484,-0.1883,-0.0519,-0.1683,-0.222,-0.1231,-0.1972,0.002
1,-0.0613,-0.1292,-0.0623,-0.214,-0.2223,-0.0074,0.0275
```

5 9 1 D A T A
0.3419,0.3392,-0.235,0.2874,0.453,0.0862,0.0545,0.0189,0.0525,0.0406,0.071
7,0.0712,-0.0242,-0.2056,-0.4936,-1.0006,-1.5154,-0.23425
5 9 2 D A T A
0.8986,-0.1837,-0.3076,-0.205,0.5636,0.7807,-0.3094,-0.5001,-0.3569,-0.5769,
-0.8743,-0.839,-0.7563,-0.4327,-0.5179,-0.5653,-0.284351
593 DATA -0.3014,-0.1765,0.0529,-0.058,0.0968
594! WEIGHTS FROM INPUT NODES TO SIXTH OF 1ST LAYER
5 9 5 D A T A
0.064,-0.0175,0.153,-0.0111,0.0684,0.0699,-0.0861,0.0028,-0.047,0.0901,-0.01
65,-0.146,-0.0543,-0.1661,-0.2121,-0.1465,0.1596,0.8206
5 9 6 D A T A
1.3595,-0.8483,0.1541,0.1844,-0.3923,-0.2206,-0.2088,-0.2671,-0.2127,-0.3005
,-0.2379,-0.2792,-0.7322,-1.5127,-3.0642,-3.2363,-0.81533
5 9 7 D A T A
-0.4224,-0.8158,-0.3201,0.3242,1.697,2.3992,0.4064,-0.2883,-0.1965,-1.0112,-
1.4027,-0.6572,-0.1271,0.2110,0.1041,0.0579,0.4539,0.4409
598 DATA 0.4365,0.483,0.4331,0.776837
599! WEIGHTS FROM INPUT NODES TO SEVENTH OF 1ST LAYER
6 0 0 D A T A
-0.0998,-0.0856,-0.1548,-0.194,-0.2965,-0.2465,-0.2582,-0.3129,-0.3448,-0.356
8,-0.4875,-0.555,-0.5793,-0.782,-0.6767,-0.448,-0.1297
6 0 1 D A T A
0.8478,1.9259,-0.8578,0.2959,0.1291,-0.7311,-0.5154,-0.4217,-0.4512,-0.545,-
0.5328,-0.5081,-0.7359,-1.1587,-2.2433,-4.1042,-3.32257
6 0 2 D A T A
-0.8171,-0.8644,-0.7449,-0.6617,0.4182,2.3219,4.041,1.063,-0.0127,-0.0747,-1.
4394,-1.9763,-0.3535,0.8045,1.1853,1.0692,1.0007,1.42489
603 DATA 1.7559,1.0772,1.0325,0.6618,1.153546
604! WEIGHTS FROM INPUT NODES TO EIGHTH OF 1ST LAYER
6 0 5 D A T A
-0.4203,-0.2698,-0.3265,-0.2915,-0.1777,-0.1976,-0.3208,-0.3167,-0.0864,-0.04
81,-0.155,-0.1317,0.1765,0.0123,-0.2389,-0.1276,0.0037
6 0 6 D A T A
0.6832,0.1658,0.6967,2.0149,1.574,-0.009,0.1473,0.3127,0.1496,0.1726,0.300
7,0.2196,-0.0018,-0.637,-2.1045,-4.484,-0.8386,2.8081,3.29761
6 0 7 D A T A
3.1827,1.8214,3.0407,3.4279,1.4383,-1.9137,-2.2437,-1.0049,-0.6885,-1.1091,-
0.6502,-0.0257,-0.2596,-1.3008,-1.3405,-1.003,-0.6684
608 DATA -0.7209,-0.7334,-0.8471,-0.41776
609! WEIGHTS FROM INPUT NODES TO NINETH OF 1ST LAYER
6 1 0 D A T A
-0.0399,-0.2013,0.0012,-0.1922,-0.2003,-0.2624,-0.1673,-0.2038,-0.2801,-0.10
44,-0.2987,-0.5289,-0.4259,-0.476,-0.5903,-0.4715,-0.1232
6 1 1 D A T A
0.8217,1.8245,-0.8881,0.1635,0.229,-0.6114,-0.3822,-0.3123,-0.4737,-0.5781,-
0.4994,-0.5227,-0.6184,-1.0763,-2.5185,-3.8384,-3.36982
6 1 2 D A T A
-0.6727,-0.8107,-0.723,-0.7565,0.4388,2.279,3.6228,0.9014,-0.0389,-0.0222,-1.
3556,-1.7643,-0.4167,0.5833,1.0759,0.9547,0.8008,1.15522

```
613     DATA 0.9138,0.9485,0.9962,0.6273,1.088135
614! WEIGHTS FROM INPUT NODES TO TENTH OF 1ST LAYER
6          1          5          D          A          T          A
0.0335,0.0834,-0.432,0.0577,.0263,0.021,0.0047,-0.0767,-0.0091,0.1423,-0.09
02,-0.1978,-0.1206,-0.0772,-0.1455,0.0232,0.0778,0.77275
6          1          6          D          A          T          A
1.3712,-0.8199,0.1232,0.2317,-0.547,-0.3027,-0.1871,-0.339,-0.3765,-0.2443,-
0.3111,-0.3103,-0.8083,-1.6881,-3.1039,-3.5085,-0.63095
6          1          7          D          A          T          A
-0.2613,-0.7899,-0.4202,0.2656,1.5939,2.6045,0.4085,-0.2042,-0.2717,-1.0349,
-1.5266,-0.5652,0.0085,0.25,0.1609,0.0629,0.5192,0.5626
618     DATA 0.567,0.4868,0.3797,0.679528
619! WEIGHTS FROM INPUT NODES TO ELEVENTH OF 1ST LAYER
6          2          0          D          A          T          A
0.1804,0.0187,0.0755,0.1723,0.0603,-0.0076,0.0529,0.0795,0.062,0.0436,-0.00
51,-0.1409,-0.0322,0.0163,-0.1906,-0.0607,0.102,0.7005
6          2          1          D          A          T          A
1.2103,-0.7533,0.2083,0.3352,-0.3232,-0.2378,-0.0945,-0.2374,-0.2468,-0.1428
,-0.2874,-0.4031,-0.6521,-1.3216,-2.6667,-3.1689,-0.71532
6          2          2          D          A          T          A
-0.1007,-0.7077,-0.4002,0.2231,1.4014,2.1345,0.2327,-0.3838,-0.1674,-0.9147,
-1.1823,-0.6173,-0.2674,-0.0742,-0.0728,-0.146,0.1058
623     DATA 0.2092,0.4318,0.4171,0.3094,0.612203
624! WEIGHTS FROM 1ST LAYER TO FIRST OF 2ND LAYER
6          2          5          D          A          T          A
-0.6856,-0.655,-0.5701,-0.5135,-0.5899,-0.624,-1.1072,-0.4819,-1.0934,-0.7625
,-0.50824
626! WEIGHTS FROM 1ST LAYER TO SECOND OF 2ND LAYER
6          2          7          D          A          T          A
-0.6415,-0.511,-0.6629,-0.6789,-0.6377,-0.7609,-1.1707,-0.5919,-0.9819,-0.758
5,-0.4999.3
628! WEIGHTS FROM 1ST LAYER TO THIRD OF 2ND LAYER
6          2          9          D          A          T          A
-0.6167,-0.5128,-0.5556,-0.5993,-0.6883,-0.6747,-1.1941,-0.5544,-1.0422,-0.73
88,-0.62614
630! WEIGHTS FROM 1ST LAYER TO FOURTH OF 2ND LAYER
6          3          1          D          A          T          A
-0.5993,-0.5907,-0.5529,-0.6479,-0.6188,-0.6775,-1.1341,-0.5158,-0.972,-0.784
2,-0.647513
632! WEIGHTS FROM 1ST LAYER TO FIFTH OF 2ND LAYER
6          3          3          D          A          T          A
-0.6721,-0.5612,-0.6241,-0.5618,-0.6884,-0.7122,-1.1498,-0.5768,-1.1434,-0.66
62,-0.53573
634! WEIGHTS FROM 1ST LAYER TO SIXTH OF 2ND LAYER
6          3          5          D          A          T          A
-0.5415,-0.6448,-0.5655,-0.5454,-0.6292,-0.6935,-1.1252,-0.5951,-1.1731,-0.74
36,-0.67345
636! WEIGHTS FROM 1ST LAYER TO SEVENTH OF 2ND LAYER
6          3          7          D          A          T          A
-0.5922,-0.6236,-0.5718,-0.5682,-0.7875,-0.7465,-1.1615,-0.5842,-1.0865,-0.7,-
```

```
0.689574
638! WEIGHTS FROM 1ST LAYER TO EIGHTH OF 2ND LAYER
6          3          9          D          A          T          A
-0.547,-0.5956,-0.6823,-0.5825,-0.5996,-0.7686,-1.0753,-0.5888,-1.1352,-0.660
1,-0.655444
640! WEIGHTS FROM 1ST LAYER TO NINETH OF 2ND LAYER
6          4          1          D          A          T          A
-0.638,-0.4725,-0.5509,-0.5237,-0.6798,-0.6825,-1.1983,-0.6198,-1.0891,-0.795
4,-0.6553
642! WEIGHTS FROM 1ST LAYER TO TENTH OF 2ND LAYER
6          4          3          D          A          T          A
-0.6326,-0.533,-0.5945,-0.6768,-0.759,-0.7709,-1.1034,-0.5356,-1.0046,-0.7246
,-0.59033
644! WEIGHTS FROM 2ND LAYER TO FIRST OF 3RD LAYER
6          4          5          D          A          T          A
1.8659,1.88,1.878,1.8692,1.8841,1.8863,1.8927,.8818,1.8922,1.8759
646! THRESHOLDS OF 1ST LAYER
6          4          7          D          A          T          A
-1.7985,-0.4371,-2.1599,-1.7573,0.4019,-2.5127,-4.6869,1.837,-4.4738,-2.6227
,-1.94113
648! THRESHOLDS OF 2ND LAYER
6          4          9          D          A          T          A
-2.6343,-2.55,-2.5803,-2.5984,-2.5636,-2.5535,-2.5385,-2.5551,-2.5454,-2.5689
650! THRESHOLDS OF 3RD LAYER
651    DATA -0.2069
652 END
653!
654 Measready:
655    SUB Measready
656      REPEAT
657        Stat=SPOLL(711)
658      UNTIL BINAND(Stat,16)
659    SUBEND
660!
661 Measscomp:
662    SUB Measscomp
663      REPEAT
664        Stat=SPOLL(711)
665      UNTIL BINAND(Stat,4)
666    SUBEND
```

Appendix Ten

Listing of HP-Basic program which displays the values of the attributes of the stopband and passband regions

```
1000    DIM Frx(50),Mrk(50),Binno(50)
1010    DIM M$[40],F$[20],A(100,2),Name$[6],File_name$[20]
1020    !
1021!   MASS STORAGE IS ":CS80,700,0"
1030    ASSIGN @Na TO 711
1040    ASSIGN @Prt TO 1
1050    Prt=1
1060    PRINTER IS 1
1070 !
1080    ASSIGN @Na_nofmt TO 711;FORMAT OFF
1090    Meas_complete=4
1100 !
1110 !
1120 !  GOTO Pband
1130 !
1140 !
1150    CLEAR @Na
1160    OUTPUT @Na;"IPR;"
1170    OUTPUT @Na;"IAR;IA1;IR1;IB1;"
1180    OUTPUT @Na;"BP0;"
1190 !
1200   OUTPUT @Na;"ST5;SM1;SFR1401500HZ;DF7;DIV1DBR;REF0DBR;"
1210    OUTPUT @Na;"SAM+5.8DBM;FM2;"
1220    OUTPUT @Na;"RPS50%;BW3;AV0;"
1230 !
1240    DISP "       Insert S/C and press 'CONT'"
1250    PAUSE
1260    DISP ""
1270 !
1280    OUTPUT @Na;"DM1;TRG;"
1290    Meascomp
1300    ENTER @Na USING "%,2A";Junk
1310    ENTER @Na_nofmt;Sc_ref
1320    PRINT "Ref   ";Sc_reff
1330    OUTPUT @Na;"DIV5DBR;REF-26DBR;"
1340 !
1350    DISP "       Insert unit and press 'CONT'"
1360    PAUSE
1370    DISP ""
1380    OUTPUT @Na;"DM1;TRG;"
1390    Meascomp
1400    ENTER @Na USING "%,2A";Junk$
1410    ENTER @Na_nofmt;Ins_loss
1420    PRINT "Approx. Insertion loss  ";-Ins_loss-26
1430 !
1440 !     GOTO Singles
1450 Again:!
1460    PRINTER IS 1
1461    Cnt=0
1470 Menu1:!
```

```
1471      INPUT "ENTER SERIAL STRING (5 CHRS MAX.)  ",Name
1472        MASS STORAGE IS ":CS80,700,1"
1474        ON ERROR GOTO Jumpa
1475        PURGE Name$
1476 Jumpa:
1477        OFF ERROR
1478        CREATE ASCII Name$,100
1479        ASSIGN @Disk TO Name$
1480   Flagp=0
1490   Flags=0
1500   OUTPUT @Na; "DF7; SAM+ 5.8DBM; FM1; ST1; SWT1SEC;
             DIV10DBR ;RPS100%;"
1510   OUTPUT @Na;"REF-26DBR;"
1520   OUTPUT @Na;"FRC1401500HZ;FRS40KHZ;SM1;BW3;"
1530   LOCAL 711
1540 !
1550   OFF KEY
1560   DISP "SELECT DISPLAY AREA....."
1570   ON KEY 5 LABEL "PASSBAND" GOTO Pband
1580   ON KEY 9 LABEL "STOPBAND" GOTO Sband
1590 Idle1:GOTO Idle1L
1600 !
1610 !
1620 !
1630 !
1640 Sband:!
1650   Flags=0
1660   DISP ""
1670   OFF KEY
1680 Menu2:!
1690   DISP "SELECT POINT WITH MARKER....."
1700   OUTPUT @Na;"SM2;"
1710   LOCAL 711
1720   ON KEY 0 LABEL "PEAK 1" GOTO P1
1730   ON KEY 1 LABEL "PEAK 2" GOTO P2
1740   ON KEY 2 LABEL "PEAK 3" GOTO P3
1750   ON KEY 3 LABEL "PEAK 4" GOTO P4
1760   ON KEY 5 LABEL "RTN  1" GOTO R1
1770   ON KEY 6 LABEL "RTN  2" GOTO R2
1780   ON KEY 7 LABEL "SKIP" GOTO Blank
1781   ON KEY 8 LABEL "SWEEP" GOTO Sweep
1790   ON KEY 9 LABEL "MENU" GOTO Lf
1800 Idle2:GOTO Idle2
1810 !
1820 P1:!
1830   IF Flags=0 THEN PRINT "Stopband"
1840   PRINTER IS Prt
1850   Flags=1
1860   OUTPUT @Na;"DM1;"
1870   ENTER @Na;Level
```

```
1880    OUTPUT @Na;"MP1;"
1890    ENTER @Na;Freq
1900 Image1:IMAGE "Peak 1 Freq   ",DDDDD.DDD," kHz    Level     " ,
             DDDD.D ," dB"
1910    PRINT USING Image1;Freq/1000,-Level+Ins_loss
1911    OUTPUT 703 USING Image1;Freq/1000,-Level+Ins_loss
1920    GOTO Menu2
1930 P2:!
1940    IF Flags=0 THEN PRINT "Stopband"
1950    PRINTER IS Prt
1960    Flags=1
1970    OUTPUT @Na;"DM1;"
1980    ENTER @Na;Level
1990    OUTPUT @Na;"MP1;"
2000    ENTER @Na;Freq
2010 Image2:IMAGE "Peak 2 Freq   ",DDDDD.DDD," kHz    Level    ",
             DDDD.D," dB"
2020    PRINT USING Image2;Freq/1000,-Level+Ins_loss
2021    OUTPUT 703 USING Image2;Freq/1000,-Level+Ins_loss
2030    GOTO Menu2
2040 P3:!
2050    IF Flags=0 THEN PRINT "Stopband"
2060    PRINTER IS Prt
2070    Flags=1
2080    OUTPUT @Na;"DM1;"
2090    ENTER @Na;Level
2100    OUTPUT @Na;"MP1;"
2110    ENTER @Na;Freq
2120 Image3:IMAGE "Peak 3 Freq   ",DDDDD.DDD," kHz    Level    ",
             DDDD.D," dB"
2130    PRINT USING Image3;Freq/1000,-Level+Ins_loss
2131    OUTPUT 703 USING Image3;Freq/1000,-Level+Ins_loss
2140    GOTO Menu2
2150 P4:!
2160    IF Flags=0 THEN PRINT "Stopband"
2170    PRINTER IS Prt
2180    Flags=1
2190    OUTPUT @Na;"DM1;"
2200    ENTER @Na;Level;
2210    OUTPUT @Na;"MP1;"
2220    ENTER @Na;Freq
2230 Image4:IMAGE "Peak 4 Freq   ",DDDDD.DDD," kHz    Level    ",
             DDDD.D," dB"
2240    PRINT USING Image4;Freq/1000,-Level+Ins_loss
2241    OUTPUT 703 USING Image4;Freq/1000,-Level+Ins_loss
2250    GOTO Menu2
2260 R1:!
2270    IF Flags=0 THEN PRINT "Stopband"
2280    PRINTER IS Prt
2290    Flags=1
```

```
2300    OUTPUT @Na;"DM1;"
2310    ENTER @Na;Level;
2320    OUTPUT @Na;"MP1;"
2330    ENTER @Na;Freq
2340 Image5:IMAGE "Return 1 Freq ",DDDDD.DDD," kHz    Level    ",
              DDDD.D," dB"
2350    PRINT USING Image5;Freq/1000,-Level+Ins_loss
2351    OUTPUT 703 USING Image5;Freq/1000,-Level+Ins_loss
2360    GOTO Menu2
2370 R2:!
2380    IF Flags=0 THEN PRINT "Stopband"
2390    PRINTER IS Prt
2400    Flags=1
2410    OUTPUT @Na;"DM1;"
2420    ENTER @Na;Level
2430    OUTPUT @Na;"MP1;"
2440    ENTER @Na;Freq
2450 Image6:IMAGE "Return 2 Freq ",DDDDD.DDD," kHz    Level    ",
              DDDD.D," dB"
2460    PRINT USING Image6;Freq/1000,-Level+Ins_loss
2461    OUTPUT 703 USING Image6;Freq/1000,-Level+Ins_loss
2470    GOTO Menu2
2471 !
2473 Sweep:  !
2474         OFF KEY
2475         DISP ""
2477         Cnt=Cnt+1
2478         File_name$=Name$&VAL$(Cnt)
2479         PRINT "<"&File_name$&">"
2487    FOR I=1 TO 21
2488      A(I,1)=0
2489      A(I,2)=0
2490    NEXT I
2491    F1=1.380000
2492    F2=1.400000
2493    Screen1=0
2494    Screen2=400
2495    Screen_step=Screen2/20
2497 !
2498   OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA"&VAL$(F1)&" MHZ; FRB"
              &VAL$(F2) &"MHZ;FM1;"
2499    OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
2500    Measready
2501    I=0
2502    FOR S=Screen1 TO Screen2 STEP Screen_step;
2503      I=I+1
2504      OUTPUT @Na;"MKP"&VAL$(S)&";"
2505      OUTPUT @Na;"DM1;"
2506      ENTER @Na;Level
2507      OUTPUT @Na;"MP1;"
```

```
2508      ENTER @Na;Freq
2509      A(I,1)=Freq/1000
2510      A(I,2)=-Level+Ins_loss+.6
2511    NEXT S
2512    PRINT "Lo Freq returns"
2513 !  OUTPUT 703;"Lo Freq returns"
2515 !  OUTPUT 703;CHR$(10)
2516    FOR J=1 TO 19 STEP 3
2517      PRINT USING Image11;A(J,1);A(J,2),A(J+1,1); A(J+1,2), A(J+2,1);
                A(J+2,2)
2518 !OUTPUT 703 USING Image11;A(J,1);A(J,2),A(J+1,1); A(J+1,2),
                A(J+2,1); A(J+2,2)
2519    NEXT J
2520    OUTPUT @Disk;File_name$
2521    OUTPUT @Disk;"        "
2523    FOR J=1 TO 21
2524    OUTPUT @Disk;A(J,2)
2525    NEXT J
2526!
2527 PRINT
2528 PRINT
2529    F1=1.404000
2530    F2=1.420000
2531    Screen1=0
2532    Screen2=400
2533    Screen_step=Screen2/20
2534    Lmin=+1000
2535 !
2536    OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA" ,&VAL$(F1)&"MHZ;FRB"
            &VAL$(F2)&"MHZ;FM1;"
2537    OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
2538    Measready
2539    I=0
2540    FOR S=Screen1 TO Screen2 STEP Screen_step;
2541      I=I+1
2542      OUTPUT @Na;"MKP"&VAL$(S)&";"
2543      OUTPUT @Na;"DM1;"
2544      ENTER @Na;Level
2545      OUTPUT @Na;"MP1;"
2546      ENTER @Na;Freq1
2547      A(I,1)=Freq/1000
2548      A(I,2)=-Level+Ins_loss+.6
2549    NEXT S
2550 Image11:IMAGE 3(DDDD.DDD,2X,DDD.D,3X)
2551    PRINT "Hi Freq returns"
2552 !  OUTPUT 703;"Hi Freq returns"
2553 !  OUTPUT 703;CHR$(10)
2554    PRINT
2555    FOR J=1 TO 19 STEP 3
2556      PRINT USING Image11;A(J,1);A(J,2),A(J+1,1); A(J+1,2),A(J+2,1);
```

```
                    A(J+2,2)
2557 !OUTPUT 703 USING Image11;A(J,1);A(J,2), A(J+1,1); A(J+1,2),
                    A(J+2,1);A(J+2,2)
2558   NEXT J
2559   OUTPUT @Disk;"        "
2560   FOR J=1 TO 21
2561   OUTPUT @Disk;A(J,2)
2562   NEXT J
2563 !
2564 !
2565 !
2567   F1=1.400000
2568   F2=1.404000
2569   Screen1=0
2570   Screen2=400
2571   Screen_step=Screen2/20
2572   Lmin=+1000
2573 !
2574   OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA"&VAL$(F1) &"MHZ;FRB"
            &VAL$(F2)&"MHZ;FM1;")
2575   OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
2576   Measready
2577   I=0
2578   FOR S=Screen1 TO Screen2 STEP Screen_step;
2579     I=I+1
2580     OUTPUT @Na;"MKP"&VAL$(S)&";"
2581     OUTPUT @Na;"DM1;"
2582     ENTER @Na;Level
2583     OUTPUT @Na;"MP1;"
2584     ENTER @Na;Freq1
2585     A(I,1)=Freq/1000
2586     A(I,2)=-Level+Ins_loss+.6
2587   NEXT S
2589   PRINT "Pass band levels"
2590 ! OUTPUT 703;"Passband levels"
2591 ! OUTPUT 703;CHR$(10)
2592   PRINT
2593   FOR J=1 TO 19 STEP 3
2594     PRINT USING Image11;A(J,1);A(J,2),A(J+1,1);A(J+1,2), A(J+2,1);
                    A(J+2,2)
2595 !OUTPUT 703 USING Image11;A(J,1);A(J,2),A(J+1,1);A(J+1,2),A(J+2,1);
                    A(J+2,2)
2596   NEXT J
2597   OUTPUT @Disk;"        "
2598   FOR J=1 TO 21
2599   OUTPUT @Disk;A(J,2)
2600   NEXT J
2603   GOTO Menu2
2604 !
2605 Blank:!
```

```
2606    PRINTER IS Prt
2607    IF Flags=0 THEN PRINT "Stopband"
2608    Flags=1
2609    PRINT "     -         -          -"
2610    PRINTER IS 1
2611    GOTO Menu2
2612!
2613!
2614 !
2615 Lf:!
2616    PRINTER IS Prt
2617    PRINT CHR$(15)
2618    PRINT CHR$(15)
2619    PRINT CHR$(15)
2620    PRINT CHR$(15)
2621    PRINTER IS 1
2622    ASSIGN @Disk TO *
2624    GOTO Again
2625 !
2626 !-----------------------------------------
2627 !
2628 A:!
2629 Pband:  !
2630 !
2631 !GOTO Singles
2632 !
2633    OFF KEY
2634    PRINTER IS Prt
2635    DISP ""
2636    Bw=30000
2637    Bw2=Bw/2
2638    F1=1400000
2639    F2=1403000
2640    F9=(F2-F1)/50
2641    Screen1=0
2642    Screen2=400
2643    Screen_step=Screen2/50
2644    Lmax=-1000
2645    Lmin=+1000
2646 !
2647    OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA1.4MHZ;FRB1.403MHZ;
            FM1;"
2648    OUTPUT @Na;"REF-26DBR;DIV2DBR;TRG;"
2649    WAIT 1.5
2650    I=0
2651    FOR S=Screen1 TO Screen2 STEP Screen_step1
2652      I=I+1
2653      OUTPUT @Na;"MKP"&VAL$(S)&";"
2654      OUTPUT @Na;"DM1;"
2655      ENTER @Na;Level
```

```
2656      OUTPUT @Na;"MP1;"
2657      ENTER @Na;Freq1
2658      A(I,1)=Freq/1000
2659      A(I,2)=-Level
2660      IF S=0 THEN Carrier=-Level
2661      IF -Level<Lmin THEN Lmin=-Level
2662      IF S>=104 AND S<=272 AND -Level>Lmax THEN Lmax=-Level
2663 !  PRINT S;Freq/1000;-Level;Lmin;Lmax
2664    NEXT S
2665    Insert_loss=Lmin-26
2666    Ripple=Lmax-Insert_loss-26
2667    FOR J=1 TO 51 STEP 1
2668! PRINT J;A(J,1);
2669      A(J,2)=A(J,2)-Insert_loss-26
2670! PRINT A(J,2)
2671    NEXT J
2672 Image10:IMAGE 3(DDDD.DDD,2X,DD.D,3X)
2673 !  FOR J=1 TO 51 STEP 3
2674 !     PRINT USING Image10;A(J,1);A(J,2),A(J+1,1);A(J+1,2),A(J+2,1);
                    A(J+2,2)
2675 !  NEXT J
2676    PRINT
2677    PRINT USING "20A,DDD.D,3A";"Insertion loss   ",Insert_loss," dB"
2678    PRINT USING "20A,DDD.D,3A";"Ripple",Ripple," dB"
2679    OUTPUT 703 USING "20A,DDD.D,3A";"Insertion loss",Insert_loss,"
                dB"
2680    OUTPUT 703 USING "20A,DDD.D,3A";"Ripple",Ripple," dB"
2681    Offset=Insert_loss+26
2682 !
2683 B:!
2684 Singles:!
2685 !
2686    OUTPUT @Na;"IPR;"
2687    OUTPUT @Na;"IAR;IA1;IR1;IB1;"
2688    OUTPUT @Na;"BP0;SAM5.8DBM;"
2689    OUTPUT @Na;"ST5;SM1;SFR1400500HZ;"
2690    OUTPUT @Na;"DF7;DIV1DBR;REF-26DBR;"
2691    OUTPUT @Na;"RPS50%;BW3;AV0;FM2;TKM;"
2692    Measready
2693    OUTPUT @Na;"DM1;TRG;"
2694    Meascomp
2695    ENTER @Na USING "%,2A";Junk$
2696    ENTER @Na_nofmt;Level"
2697    PRINT USING "20A,DDD.D,3A";"Lo P/B",-Level-Offset," dB"
2698    OUTPUT 703 USING "20A,DDD.D,3A";"Lo P/B",-Level-Offset," dB"
2699 !
2700    OUTPUT @Na;"SFR1402500HZ;TKM;"
2701    Measready
2702    OUTPUT @Na;"DM1;TRG;"
2703    Meascomp
```

```
2704   ENTER @Na USING "%,2A";Junk$
2705   ENTER @Na_nofmt;Level"
2706   PRINT USING "20A,DDD.D,3A";"Hi P/B",-Level-Offset," dB"
2707   OUTPUT 703 USING "20A,DDD.D,3A";"Hi P/B",-Level-Offset," dB"
2708 !
2709   OUTPUT @Na;"SFR1400000HZ;TKM;"
2710   Measready
2711   OUTPUT @Na;"DM1;TRG;"
2712   Meascomp
2713   ENTER @Na USING "%,2A";Junk$
2714   ENTER @Na_nofmt;Level"
2715   PRINT USING "20A,DDD.D,3A";"C/R   ",-Level-Offset," dB"
2716   OUTPUT 703 USING "20A,DDD.D,3A";"C/R   ",-Level-Offset," dB"
2717 !
2718   OUTPUT @Na;"SFR1399300HZ;TKM;"
2719   Measready
2720   OUTPUT @Na;"DM1;TRG;"
2721   Meascomp
2722   ENTER @Na USING "%,2A";Junk$
2723   ENTER @Na_nofmt;Level"
2724   PRINT USING "20A,DDD.D,3A";"Lo S/B",-Level-Offset," dB""
2725   OUTPUT 703 USING "20A,DDD.D,3A";"Lo S/B",-Level-Offset," dB"
2726 !
2727   OUTPUT @Na;"SFR1405000HZ;TKM;"
2728   Measready
2729   OUTPUT @Na;"DM1;TRG;"
2730   Meascomp
2731   ENTER @Na USING "%,2A";Junk$
2732   ENTER @Na_nofmt;Level"
2733   PRINT USING "20A,DDD.D,3A";"Hi S/B",-Level-Offset," dB"
2734   OUTPUT 703 USING "20A,DDD.D,3A";"Hi S/B",-Level-Offset," dB"
2735 !
2736   FOR I=1 TO 51
2737     A(I,1)=0
2738     A(I,2)=0
2739   NEXT I
2740   F1=1.380000
2741   F2=1.398000
2742   Screen1=0
2743   Screen2=400
2744   Screen_step=Screen2/30
2745   Lmin=+1000
2746 !
2747  OUTPUT @Na;"ST1;SM2; SWT1SEC;FRA" &VAL$(F1) &"MHZ; FRB"
            &VAL$(F2) &"MHZ;FM1;")
2748   OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
2749   Measready
2750   I=0
2751   FOR S=Screen1 TO Screen2 STEP Screen_step;
2752     I=I+1
```

```
2753      OUTPUT @Na;"MKP"&VAL$(S)&";"
2754      OUTPUT @Na;"DM1;"
2755      ENTER @Na;Level
2756      OUTPUT @Na;"MP1;"
2757      ENTER @Na;Freq1
2758      A(I,1)=Freq/1000
2759      A(I,2)=-Level-Offset
2760      IF -Level-Offset<Lmin THEN Lmin=-Level-Offset
2761    NEXT S
2762    Return_level=Lmine
2763 !  FOR J=1 TO 31 STEP 3
2764 !    PRINT USING Image11;A(J,1);A(J,2),A(J+1,1);A(J+1,2),A(J+2,1);
                    A(J+2,2)
2765 !  NEXT J
2766 !  PRINT
2767    PRINT USING "20A,DDD.D,3A";"Lo return level  ",Lmin," dB"
2768    OUTPUT 703 USING "20A,DDD.D,3A";"Lo return level  ",Lmin," dB"
2769!
2770    F1=1.406000
2771    F2=1.420000
2772    Screen1=0
2773    Screen2=400
2774    Screen_step=Screen2/30
2775    Lmin=+1000
2776 !
2777    OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA" &VAL$(F1)&"MHZ;FRB"
            &VAL$(F2)&"MHZ;FM1;"
2778    OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
2779    Measready
2780    I=0
2781    FOR S=Screen1 TO Screen2 STEP Screen_step;
2782      I=I+1
2783      OUTPUT @Na;"MKP"&VAL$(S)&";"
2784      OUTPUT @Na;"DM1;"
2785      ENTER @Na;Level
2786      OUTPUT @Na;"MP1;"
2787      ENTER @Na;Freq1
2788      A(I,1)=Freq/1000
2789      A(I,2)=-Level-Offset
2790      IF -Level-Offset<Lmin THEN Lmin=-Level-Offset
2791    NEXT S
2792    Return_level=Lmine
2793 !  FOR J=1 TO 31 STEP 3
2794 !    PRINT USING Image11;A(J,1);A(J,2),A(J+1,1);A(J+1,2),A(J+2,1);
                    A(J+2,2)
2795 !  NEXT J
2796 !  PRINT
2797    PRINT USING "20A,DDD.D,3A";"Hi return level  ",Lmin," dB"
2798    OUTPUT 703 USING "20A,DDD.D,3A";"Hi return level  ",Lmin," dB"
2799    GOTO Menu1
```

```
2800 !
2801   ENDO
2802 !
2803 !
2804 Measready:!
2805   SUB Measready
2806     REPEAT
2807       Stat=SPOLL(711)
2808     UNTIL BINAND(Stat,16)
2809   SUBEND
2810 !
2811 Meascomp:!
2812   SUB Meascomp
2813     REPEAT
2814       Stat=SPOLL(711)
2815     UNTIL BINAND(Stat,4)
2816   SUBEND
```

Appendix Eleven

Listing of HP-Basic program which samples the magnitude response

```
1000   DIM Frx(50),Mrk(50),Binno(50)
1010   DIM M$[40],F$[20],A(100,2),Name$[6],File_name$[20]
1020   !
1030   ASSIGN @Na TO 711
1040   ASSIGN @Prt TO 1
1050   Prt=1
1060   PRINTER IS 1
1070 !
1080   ASSIGN @Na_nofmt TO 711;FORMAT OFF
1090   Meas_complete=4t
1100 !
1110 !
1150   CLEAR @Nal
1160   OUTPUT @Na;"IPR;"
1170   OUTPUT @Na;"IAR;IA1;IR1;IB1;"
1180   OUTPUT @Na;"BP0;"
1190 !
1200  OUTPUT @Na;"ST5;SM1;SFR1401500HZ;DF7;DIV1DBR;REF0DBR;"
1210   OUTPUT @Na;"SAM+5.8DBM;FM2;"
1220   OUTPUT @Na;"RPS50%;BW3;AV0;"
1230 !
1240   DISP "      Insert S/C and press 'CONT'"
1250   PAUSE
1260   DISP ""
1270 !
1280   OUTPUT @Na;"DM1;TRG;"
1290   Meascomp
1300   ENTER @Na USING "%,2A";Junk$
1310   ENTER @Na_nofmt;Sc_ref
1320   PRINT "Ref   ";Sc_reff
1330   OUTPUT @Na;"DIV5DBR;REF-26DBR;"
1340 !
1350   DISP "      Insert unit and press 'CONT'"
1360   PAUSE
1370   DISP ""
1380   OUTPUT @Na;"DM1;TRG;"
1390   Meascomp
1400   ENTER @Na USING "%,2A";Junk$
1410   ENTER @Na_nofmt;Ins_loss
1420   PRINT "Approx. Insertion loss  ";-Ins_loss-26R
1430 !
1450 Again:!
1460   PRINTER IS 1
1470 Menu1:!
1500  OUTPUT @Na;"DF7;SAM+5.8DBM;FM1; ST1; SWT1SEC; DIV10DBR;
     RPS100%;"
1510   OUTPUT @Na;"REF-26DBR;"
1520   OUTPUT @Na;"FRC1401500HZ;FRS40KHZ;SM1;BW3;"
1530   LOCAL 711
1540 !
```

```
1640 Sband:!
1680 Menu2:!
1781    ON KEY 8 LABEL "SWEEP" GOTO Sweep;
1800 Idle2:GOTO Idle2
1810 !
2473 Sweep:   !
2474          OFF KEY
2475          DISP ""
2487    FOR I=1 TO 21
2488       A(I,1)=02
2489       A(I,2)=02
2490    NEXT I
2491    F1=1.380000
2492    F2=1.400000
2493    Screen1=0
2494    Screen2=400
2495    Screen_step=Screen2/20
2497 !
2498    OUTPUT @Na; "ST1;SM2;SWT1SEC;FRA"&VAL$(F1)&"MHZ; FRB
 "&VAL$(F2)&"MHZ;FM1;"
2499    OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
2500    Measready
2501    I=0
2502    FOR S=Screen1 TO Screen2 STEP Screen_step;
2503       I=I+1
2504       OUTPUT @Na;"MKP"&VAL$(S)&";"
2505       OUTPUT @Na;"DM1;"
2506       ENTER @Na;Level
2507       OUTPUT @Na;"MP1;"
2508       ENTER @Na;Freq
2509       A(I,1)=Freq/1000
2510       A(I,2)=-Level+Ins_loss+.6
2511    NEXT S
2512    PRINT "Lo Freq Returns"
2516    FOR J=1 to 19 STEP 3
2517       PRINT USING Image11;A(J,1);A(J,2);A(J+1,1);A(J+1,2),A(J+2,2)
2519    NEXT J
2526 !
2527  PRINT
2528  PRINT
2529     F1=1.404000
2530     F2=1.420000
2531     Screen1=0
2532     Screen2=400
2533     Screen_step=Screen2/20
2534     Lmin=+1000
2535 !
2536    OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA" &VAL$(F1) &"MHZ; FRB
              "& VAL$(F")&"MHX;FM1;"
2537     OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
```

```
2538   Measready
2539   I=0
2540   FOR S=Screen1 TO Screen2 STEP Screen_step;
2541     I=I+1
2542     OUTPUT @Na;"MKP"&VAL$(S)&";"
2543     OUTPUT @Na;"DM1;"
2544     ENTER @Na;Level
2545     OUTPUT @Na;"MP1;"
2546     ENTER @Na;Freq
2547     A(I,1)=Freq/1000
2548     A(I,2)=-Level+Ins_loss+.6
2549   NEXT S
2550 Image11:IMAGE 3 (DDDD.DDD,2X,DDD.D,3X)
2551   PRINT "Hi Freq Returns"
2554   PRINT
2555  FOR J=1 to 19 STEP 3
2556     PRINT USING Image11;A(J,1);A(J,2);A(J+1,1);A(J+1,2),A(J+2,2)
2558   NEXT J
2565 !
2567  F1=1.400000
2568  F2=1.404000
2569    Screen1=0
2570    Screen2=400
2571    Screen_step=Screen2/20
2572    Lmin=+1000
2573 !
2574   OUTPUT @Na;"ST1;SM2;SWT1SEC;FRA" &VAL$(F1) &"MHZ; FRB
            "& VAL$(F")&"MHX;FM1;"
2575    OUTPUT @Na;"RPS50%;REF-45DBR;DIV10DBR;TRG;"
2576   Measready
2577   I=0
2578   FOR S=Screen1 TO Screen2 STEP Screen_step;
2579     I=I+1
2580     OUTPUT @Na;"MKP"&VAL$(S)&";"
2581     OUTPUT @Na;"DM1;"
2582     ENTER @Na;Level
2583     OUTPUT @Na;"MP1;"
2584     ENTER @Na;Freq
2585     A(I,1)=Freq/1000
2586     A(I,2)=-Level+Ins_loss+.6
2587   NEXT S
2589   PRINT "Pass band levels"
2592   PRINT
2593   FOR J=1 to 19 STEP 3
2594     PRINT USING Image11;A(J,1);A(J,2);A(J+1,1);A(J+1,2),A(J+2,2)
2596   NEXT J
2603   GOTO Menu2
2800 !
2801   END
2802 !
```

```
2804 Measready:!
2805    SUB Measready
2806       REPEAT
2807         Stat=SPOLL(711)
2808       UNTIL BINAND(Stat,16)
2809    SUBEND
2810 !
2811 Meascomp:!
2812    SUB Meascomp
2813       REPEAT
2814         Stat=SPOLL(711)
2815       UNTIL BINAND(Stat,4)
2816    SUBEND
```

LEARNING BY ANALYTICAL METHODS OR INDUCTION : A CASE STUDY

D. Tsaptsinos, B. W. Jervis and A. R. Mirzai

Abstract

An expert system is under construction for the application of tuning electronic filters. Machine learning techniques were used to enhance knowledge elicitation and experiences gained in implementing them are presented in terms of learning, testing, and learning refinement.

The application

The tuning of an electronic filter is typically performed manually. Initial knowledge elicitation using protocol analysis revealed the nature of the problem domain. An operator monitored the magnitude response of the filter and when tuning was required a set of tunable components was adjusted. This was followed by determining a number of frequency and attenuation points.

Initial knowledge elicitation

The tuning of two filters of the same characteristics was video-taped. The analysis of the transcripts resulted in the identification of the activity classes, a domain dictionary, the reasoning process of tuning and the order of specification checking. It is interesting to note that the justifications for the actions taken comprised of a mixture of symbols (eg. the left peak), abstractions (eg. the peak is too far out) and numerical parameters (especially during testing). When the operator was prompted for further elaboration (eg. can you define the value, or range of values, where the peak should be) the answers did not provide any further information. This made the knowledge elicitation even more difficult. It was found necessary to separate the process into two tasks (stopband and passband tuning) and to recognise which components to employ for each task.

Machine learning techniques

To aid the knowledge elicitation for the task of stopband tuning, for which two tuning components were applicable, three machine learning techniques were applied, namely, ID3 [1], an adaptive combiner [2] and a neural network [3]. A set of examples was generated. Each example was described in terms of six attributes derived from the magnitude response. Each attribute, in turn, contained a numerical value with six significant figures. In a series of experiments, the three techniques were compared and ID3 was elected for further use [4]. Moreover, a number of experiments took place which compared the ID3 performance using different numbers of attributes in different formats (logical or numerical). The results are reported in reference [5].

D. Tsaptsinos and B.W. Jervis are with the School of Engineering IT, Sheffield City Polytechnic. A.R. MIrzai with the School of Mechanical Engineering, Polytechnic of Central London.

Problems encountered

The presentation of attributes holding numerical values to ID3 introduced the following problem. The enlargement of the learning set resulted, at best, in a slight change of the threshold values of the decision tree. This led to different classifications of a number of testing examples. In addition, new attributes were introduced or old attributes were excluded from the newly generated decision tree. The grouping of numerical values into logical ranges [5] resulted in more stable decision trees in terms of attributes and thresholds. The introduction of further examples to the adaptive combiner or the neural net did not alter the architectural structure but strengthened or weakened the individual nodes.

The task of stopband tuning was divided into three searches. Each search had a different goal. Search one contained rules on how to recognise a tuned state (ie. 2 classes). Search two incorporated rules for which component to tune and in which direction (ie. 4 classes). Search three determined the amount of turn (ie. 11 classes). Unlike adaptive combiners or neural networks which can provide continuous output ID3 had to be presented with examples covering all eleven classes. The large number of classes meant that the examples were less representative with the consequence of poor performance. This deficiency was the reason for dividing the task into three searches in the first place.

To improve the adaptive combiner performance it was necessary to manipulate the attribute set. The manipulation took the form of scaling the attribute values and/or the introduction of second order features. The scaling of the attribute values was important. Without a proper scaling an ill-conditioned problem was created in terms of the auto-correlation matrix in the RLS algorithm [6]. It was possible to find if the problem was ill-conditioned by using eigenvalue analysis [7]. The adaptive combiners are linear structures and they cannot directly model non-linearity. Therefore it was necessary to introduce second order features (ie. attribute squared) to handle the non-linearities. Scaling was also required for the neural network implementation. All three techniques produced comparable results but the use of ID3 was less time consuming. The actual learning time for the combiner was small but some extra time was required to identify the right format of the attributes and their values. Neural networks were also time consuming. A lot of time was spend in investigating different architectures and the learning time for some architectures ran into hours.

The structure of the learning set was of some importance. Initial work with neural networks and ID3 employed examples generated while tuning a number of filters. When both techniques were tested using unseen examples ID3 performed better. Neural nets failed to classify correctly a number of testing examples. Those examples contained at least one attribute with a value previously found in an example with a different classification. Because of the large range of numerical values each attribute can take, a different set of learning examples was required which included either all likely values or the extreme values (ie. maximum or minimum). This was also necessary when using the adaptive combiners. The new training set missed out the heuristics employed by the operator but was appropriate for the comparison. Unlike ID3 which learned by considering all examples at once, adaptive combiner and neural network implementations learned in an incremental fashion. The weights were updated with each example presented. For that reason the order of introduction of the examples was critical. Examples with different class were presented alternatively. In this way a better performance was achieved.

The set of rules produced by ID3 can be examined to identify the relationships that exist between the attributes. The outcomes of the adaptive

combiner or neural network (vector of weights) can be also examined. Weights can be transformed into rules but this presents a more difficult challenge [6].

Testing the rules generated by ID3 was more time consuming than testing the two other techniques. More importantly, though, was that in running the ID3 algorithm, examples with unknown attribute values could not be used when numerical attributes were employed. A notation to indicate that the attribute value was not significant was available but not to indicate that an attribute value was unknown. It was then impossible to use both numeric and symbolic descriptions for an attribute. If at any point an attribute value was requested and this value was unknown then the system failed completely. Using the other two techniques this could not happen. Unknown values were represented with a constant. During testing the network might not perform appropriately but it would not fail.

Conclusions

ID3 and adaptive combiners learned faster than neural networks once the structure of the learning set had been established. Testing adaptive combiners and neural nets was quicker than testing ID3. The problems with ID3 testing and refining when numerical attribute values were used were by-passed with the use of logical attribute values. Introducing new examples did not alter the architecture of adaptive combiners or neural nets.

Acknowledgements

The authors wish to thank Dr. Anna Hart from Lancashire Polytechnic for useful discussions and Mr. Lawrence Mack for his continuing help and expertise.

References

1. QUINLAN, J.R.: "Induction of decision trees", Machine Learning, Vol 1, pp 81-106, 1986.

2. GRANT, P.M., MIRZAI A.R., BROWN K.E., CRAWFORD T.M.: "Intelligent techniques for electronic component and system alignment", IEE Electronics and Communication Engineering Journal, pp 23-32, Vol 1, No 1, 1989.

3. LIPMANN, R.P.: "An introduction to computing with neural nets", IEEE ASSP Magazine, pp 4-22, April 1987.

4. TSAPTSINOS, D., MIRZAI, A.R., JERVIS, B.W.: "Comparison of machine learning paradigms in a classification task", Proceedings of the Fifth International Conference on Applications of Artificial Intelligence in Engineering, Computational mechanics publications, July 1990.

5. TSAPTSINOS, D., MIRZAI, A.R., JERVIS, B.W., COWAN C.F.N.: "Comparison of knowledge elicitation techniques in the domain of electronic filter tuning", submitted to IEE Proc. F, Radar and Signal Processing.

6. MIRZAI, A.R., COWAN, C.F.N., CRAWFORD, T.M.: "Learning using pattern recognition and signal processing",In: Artificial Intelligence:Concepts and Applications in Engineering (ed. A.R Mirzai), Chapman and Hall Computing, London, 1990.

7. WIDROW, B., STEARNS, S.D.: "Adaptive signal processing", Prentice Hall, Englewood Cliffs, N.J., 1985.

Comparison of knowledge elicitation techniques in the domain of electronic filter tuning

D. Tsaptsinos
A.R. Mirzai
B.W. Jervis, PhD
C.F.N. Cowan, PhD

Indexing terms: Algorithms, Artificial intelligence, Filters, Manufacturing

Abstract: The work done towards the construction of an expert system to assist an operator in the identification of the corrective action to be applied during tuning of electronic filters is described. The first part of the paper introduces two algorithms for induction by examples (ID3 and adaptive combiner) and their relationship to expert systems. The two algorithms were applied, in a series of tests which involved an incremental presentation of a number of examples, to the task of filter tuning. The reported results suggest the use of ID3 when a small number of classes is present. The second part of the paper presents subsequent work with ID3. Results are reported of using this algorithm for filter tuning with examples containing either numerical or logical attribute values. A comparison of the results showed that improved test performance was achieved by using logical values.

1 Introduction

Artificial-intelligence techniques, such as expert systems and machine learning, have been applied to engineering applications [1]. This paper reports work on the application of the expert system concept in the field of signal processing by electronic filters. In particular, an expert system is being constructed to assist in the tuning of filters after manufacture. The filter employed for this study is an asymmetric bandpass crystal filter, whose top view is shown in Fig. 1. At present, filter tuning is typically a manual process involving an operator who inspects the performance of the filter (e.g. the amplitude response (Fig. 2)) and applies the necessary successive adjustments to a set of tunable components (e.g. coils) until the performance satisfies the specification. Effectively the operators act as signal interpreters who base their skills not on any theory but on the combination of a strong capability for pattern recognition and knowledge acquired from past experience. Although manual tuning is successful the advantages of providing computerised assistance to an operator have been recognised [2, 3]. Initial knowledge elicitation by protocol analysis [4] revealed the problems to be solved by the expert system-builder (e.g. why component X should be adjusted rather than component Y at a particular instant) and

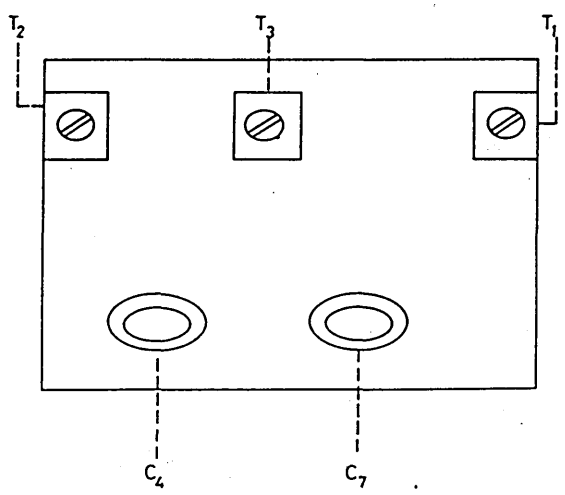


Fig. 1 *Top view of filter*
C_4, C_7 are trimmer capacitors; T_1, T_2, T_3 are inductors

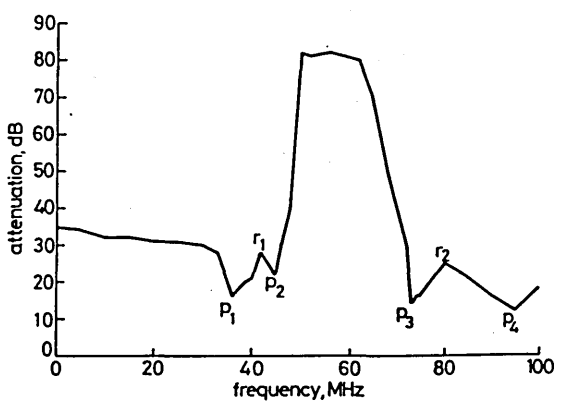


Fig. 2 *Normalised magnitude response*
p_1, p_2, p_3, p_4 are positive peaks; r_1, r_2 are negative peaks

Paper 7459F (C4, E5), first received 8th September 1989 and in revised form 27th June 1990

Mr. Tsaptsinos and Dr. Jervis are with the School of Engineering Information Technology, Sheffield City Polytechnic, Pond Street, Sheffield S1 1WB, United Kingdom

Dr. Cowan is with the Department of Electrical Engineering, University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh EH9 3JL, United Kingdom

Dr. Mirzai was formerly with the Department of Electrical Engineering, University of Edinburgh and is now with the Department of Mechnical Engineering, Polytechnic of Central London, 115 New Cavendish Street, London W1M 8JS, United Kingdom

indicated the need for an alternative to protocol analysis for articulating knowledge. Thus it was found necessary to integrate machine learning into knowledge elicitation. In this paper experiments which were carried out to investigate and compare the applicability of the ID3 and adaptive-combiner techniques to the field of filter tuning are discussed. Furthermore, the results of empirically comparing various decision trees generated using ID3 are reported.

2 Expert and machine-learning systems

Several expert systems have been constructed [5] and their number is rapidly increasing. The purpose of such a system is to incorporate, in an organised way, the substantial knowledge of one or more specialists in a specific field so that the system performs in a similar fashion to the specialists. A classical method for the construction of an expert system involves an iterative interaction between the system builder and the specialist and the encoding of the elicited knowledge in rule form. A number of techniques have been identified as aids to the knowledge elicitation process [6]. Expert systems perform in a deductive format [7, p. 4], i.e. the conclusions always depend on the knowledge supplied. The presence of an incorrect conclusion can generally only be corrected by the builder's interference and not by the system itself. (Even methods for refining existing knowledge bases [8] require additional interaction with the expert.) In contrast, machine-learning systems improve the quality of their performance with time. Three major research paradigms can be identified: neural modelling and decision-theoretic techniques; symbolic concept acquisition (SCA); and knowledge-intensive, domain-specific learning [7, p. 12]. Each paradigm is based upon the same principle, namely that of inferring conclusions given *a priori* knowledge, and differs from others only in the amount of information required and in the way the knowledge is represented and modified.

A number of learning strategies have been documented [7, p. 13] but in the work reported here techniques which learn from data composed of a number of independent examples have been implemented. Each example is described in terms of a number of attribute values, together with an additional attribute, known as the class, which allocates the examples to a particular category (supervised learning). A number of different techniques were available, e.g. neural networks [9], genetic algorithms [10] and AQ11 [11]. The techniques chosen were ID3 and adaptive combiners and these are outlined briefly in the following sections. ID3 is an example of an SCA system and adaptive combiners can be considered as a subset of neural networks. The difference between them, apart from the algorithm employed, lies in the way the knowledge is represented. In ID3 the knowledge is held within a decision tree, in adaptive combiners it is held in a weight matrix. The reasons for choosing these two techniques were more practical than theoretical. Extensive expertise and previous work using adaptive combiners in the field of filter tuning by one of the authors, in addition to the availability of a commercial package implementing ID3, were the main factors behind the decision. It has also been reported that ID3 is faster, in terms of induction, than AQ11 [12] or a genetic algorithm [13] with the same performance rate. Two- and three-layer neural networks using the back-propagation technique [14] have also been considered and the results are reported in Reference 15.

The use of machine learning techniques was intended to enhance the initial knowledge elicitation and to overcome the problem of selecting the appropriate component to tune.

2.1 The ID3 algorithm

ID3 (iterative dichotomiser 3) was developed by Quinlan [16] in 1979. The goal of the algorithm is to induce a decision tree (which can easily be transformed into rules of the form 'if x then y') from a set of examples. The decision tree can then be used to classify an unseen example. A diagrammatic description of the algorithm is shown in Fig. 3. The algorithm selects the most informa-

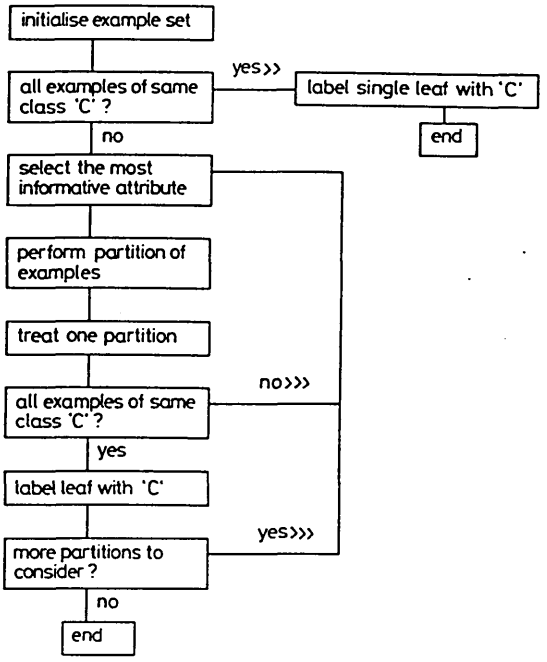


Fig. 3 *ID3 algorithm*

tive attribute (i.e. creates the root of the decision tree) and either forms subsets equal in number to the number of values the attribute takes (i.e. creates the branches of the decision tree) or forms a binary split (cutoff points) when the attribute holds numerical values (e.g. >5, $\leqslant 5$). For each subset the algorithm checks whether all the examples belong to the same class. If they do then the algorithm labels that subset with the name of the class (i.e. creates a leaf of the decision tree) and partitioning stops for that subset; otherwise the algorithm creates further, smaller subsets. The algorithm stops when no more subsets can be created. The key principle underpinning the algorithm lies in the selection of the most informative attribute. This is based on information theory; it may be stated that the most informative attribute is the one that maximises the difference between the expected information of the whole set of examples and the expected information of the whole set of examples when only attribute, X, is considered. A detailed explanation of the formulas used can be found in Reference 17. It is worth noticing that the algorithm may label a leaf as 'empty' or 'clash'. 'Empty' appears when there are no examples that can be used for that particular branch. 'Clash' emerges when there are two (or more) examples covering that specific branch but their classes are distinct.

2.2 Adaptive combiners

In recent years one class of adaptive architectures, linear combiners, has been used for the design of intelligent

systems [18]. Fig. 4 illustrates a simple combiner structure. Each example is presented to the algorithm by a matrix containing the attribute values (the X's) and the class y. The goal of the learning algorithm is to estimate the weight vector (the W's) in such a way that, when the

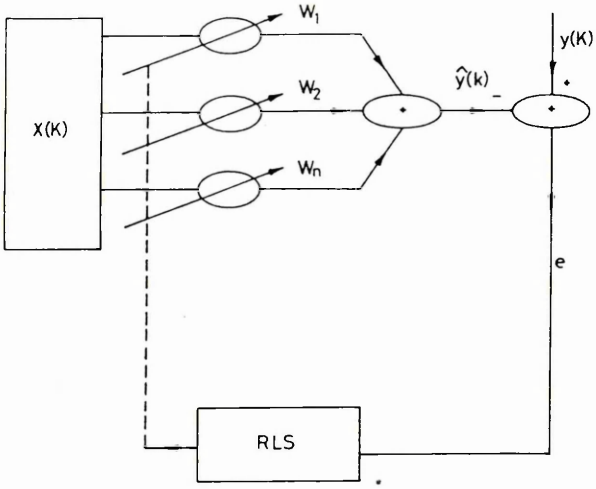


Fig. 4 *Adaptive combiner architecture*

system is presented with a previously unseen example, it can predict the correct class. In other words, the weight vector in the combiner represents the knowledge relating the attributes to the classes. For each example an estimated class is calculated by simply multiplying the attribute values (in a transposed form) and the weight vector. The difference between the estimated and the desired class is the error. The recursive least squares (RLS) algorithm is then used to estimate a new weight vector to minimise the mean squared error. The weights are functions of all the examples in the training set. The formulas used can be found in reference [9]. The RLS algorithm can be slightly modified by introducing a 'forgetting' factor in the range 0.9 to 1.0. This has the effect of giving greater importance to more recent examples than older ones and allows relearning of the same training set. The adaptive combiner structure can be thought of as a one-layer connectionist network. The main difference is that the combiners are linear structures and cannot be directly applied to nonlinear systems. However, the nonlinearity can be treated by manipulating the attributes, i.e. by using second- or third-order attributes depending on the degree of nonlinearity.

3 Comparison of machine learning techniques

The results of experiments to determine the more suitable of the two techniques (ID3 and combiners) for filter tuning are presented in this Section.

3.1 Selection of attributes and generation of examples

To use either technique one has to select a set of attributes and to generate a set of examples.

Attributes can be thought of as those relevant factors (features) used in reaching a decision. As a result of the protocol analysis the tuning of the filter was divided into two primary tasks, namely the tasks of tuning the stopband and the passband regions. Additionally, it was established that only the two trimmer capacitors were to be used for the stopband region. In this paper results are presented for this region only. Six relevant attributes were identified as having strong significance. These were:

(i) locations of sharp positive peaks of the waveform (identified as p_1, p_2, p_3 and p_4 in Fig. 2 and measured in megahertz units)

(ii) relative magnitudes of sharp negative peaks of the waveform (identified as r_1 and r_2 in Fig. 2 and measured in decibels).

This identification was based on the transcripts derived from the protocol analysis. The operator's reasoning was revealed by sentences such as '... arrange these peaks into a more reasonable place' and '... pull that peak out of the screen'. Further discussion with the operator supported the selection.

The second step was to obtain a set of examples. For the purpose of comparing the two machine-learning techniques the detune procedure was employed. This procedure ensured that a set was obtained which covered most of the attribute values likely to arise, a feature which is especially valuable in work in which numerical attributes are used. The process involved an operator tuning the stopband region of a filter and recording the attribute values together with the class 'end-of-process'. This was followed by a systematic detuning in which one of the tunable components was kept constant and the other was misadjusted in a certain direction (i.e. clockwise or anticlockwise) in quarter-turn steps. For each turn the attribute values, together with the component, direction of turn and how far the component was turned, were recorded. The process was then repeated by misadjusting in a different direction and by using the other component. Obviously, the filter was retuned in between. In this way 43 examples were collected for one filter. The tuning of six filters resulted in a total of 258 examples.

3.2 Levels of classification

Two typical recorded examples took the following form

1.39678 1.40234 1.41967 1.42003 45 53 C4a0.5

1.39756 1.40856 1.41325 1.43256 48 51 end-of-process

which can be translated as 'turn the C_4 component anticlockwise by half a turn' (first example) and 'no further tuning is required' (second example) 'when the attributes have the given values'. It is clear that the first example points to another decision, that of continuing the tuning. Previous work resulted in the identification of three search spaces:

(i) search space 1: to carry on or to end the tuning (2 classes)

(ii) search space 2: which component to adjust and which direction (4 classes)

(iii) search space 3: how far to turn (11 classes).

Thus, given a set of attribute values the system is to decide if further tuning is required. If it is, the same attribute values are to be used to define the component, direction and distance to turn; otherwise the process can be repeated with a new set of values of another untuned filter.

3.3 Presentation of examples

The attribute values were used in the two techniques in their recorded format. Initially eight examples were used in the learning set. These comprised four 'end-of-process' examples and four 'carry on' examples for the same filter. The latter included those examples generated when the components were adjusted to their maximum positions in both directions. Four more examples were then introduced, namely those generated when the components

were turned half way. Finally, the four examples which arose when the components were adjusted to their minimal positions were presented. For the second and third search the same examples were presented but with the 'carry on' class replaced by either the component/direction or the distance, respectively. The 'end-of-process' examples were replaced by the examples generated with the minimum turn. At each stage the generated decision tree or weight vector was tested against the training set S_1, the remaining unseen examples of the same filter S_2 and the unseen examples of the rest of the filters S_3. Finally, the total performance was calculated (*Total*).

3.4 Comparison criteria

Machine learning involves generalising from a set of examples and identifying those attributes and attribute values that can be used to discriminate between classes. The quality of generalisation depends heavily on the selected attributes (sufficient or inadequate?) and the number of examples present. At this stage of the work the hypothesis was that the chosen attributes were adequate. However, the number of examples necessary was unknown. The objective of the comparison was to identify the technique which used the least number of examples while giving a satisfactory performance. Note that in using the set of examples, either to learn or to test, the assumption is being made that, given a set of attribute values, the only correct action is the one as defined by the example. By 'correct' action is meant that action which would have resulted in tuning the stopband in the least number of steps.

3.5 Search 1 comparison

The following points can be made regarding the results obtained (Table 1). ID3 is seen always to be capable of predicting accurately those examples presented to it in the training set S_1. Furthermore, by taking into account the percentage success rate it can be concluded that a satisfactory generalisation has been achieved with few examples. Introducing extra examples tends to improve the generalisation even further. Unfortunately, this is misleading. Closer inspection of the test results shows that the high success rate was due to the presence of a large number of 'carry on' examples. ID3 successfully predicted the 'carry on' examples but failed to recognise the 'end-of-process' ones, i.e. there was no true classification.

The only option for improving the ID3 performance was to introduce further 'end-of-process' examples. It was found that by increasing the learning set to 18 the objective was achieved with a 96% success rate (row 4 of Table 1).

The performance (Table 1) of the adaptive combiner with a forgetting factor of 0.9 also tends to improve with the presentation of extra examples, the performance of the training set S_1 being the exception. Unfortunately, like ID3, a large number of 'end-of-process' examples were misclassified. Therefore, experiments were carried out to investigate the effects of varying the forgetting factor and of retraining the combiner with the same training set. Table 2 shows the predictive accuracy of the combiner when the forgetting factor equals 0.9 and the training set was presented to the combiner nine times. Further experiments involved manipulation of the attributes and the presentation of the attribute values. Table 3 shows the results obtained when only four attributes were used (the four positive peaks) with values scaled between 0 and 100. In this case, it is interesting to notice that the combiner performed best when only eight examples were used. This is mainly due to the fact that, when a large number of examples from one filter are shown to the combiner in the training mode, it cannot recognise examples of the other filters (S_3) very well. In both cases (Tables 1, 2) the 'end-of-process' examples were recognised.

Table 2: Combiner predictive accuracy for Search 1 and adjusted parameters (nine relearning loops and forgetting factor = 0.9)

Number of learning examples	% rate of success on			
	S_1	S_2	S_3	*Total*
16	94	100	91	92

Table 3: Combiner predictive accuracy for Search 1 and reduced set of attributes (four positive peaks) and scaled values (0–100)

Number of learning examples	% rate of success on			
	S_1	S_2	S_3	*Total*
8	100	88	90	90
12	100	91	84	85
16	94	98	85	87

3.6 Search 2 comparison

The three learning sets were introduced to the ID3 algorithm. Table 4 shows the results obtained. Note that even when eight examples were used the prediction rate was acceptable and that the performance did not improve with the introduction of further examples. This is probably an indication that further attributes are required if better performance is to be achieved.

Table 4: ID3 Predictive accuracy for Search 2

Number of learning examples	% rate of success on			
	S_1	S_2	S_3	*Total*
8	100	100	88	91
12	100	100	88	91
16	100	100	88	91

Table 1: ID3 and adaptive combiner predictive accuracy for Search 1 (forgetting factor = 0.9)

Number of learning examples	% rate of success on							
	S_1		S_2		S_3		*Total*	
	ID3	combiner	ID3	combiner	ID3	combiner	ID3	combiner
8	100	87	82	57	80	67	81	67
12	100	100	81	77	80	82	81	82
16	100	75	100	100	93	91	94	91
18	100		100		96		97	

As for Search 1, it was found necessary to present the combiner with a reduced number of attributes (the four positive peaks) and to scale the values between 0 and 100 to improve the performance. It was also necessary to include 'end-of-process' examples as well. The first training set contained five examples, i.e. one 'end-of-process' plus four examples when the screws were misadjusted to their maximum positions. Then the examples corresponding to the minimum positions of the screws were added to the learning set (i.e. nine examples in all) and finally the examples corresponding to the half-way misadjustments of the screws plus one more 'end-of-process' example were added, resulting in 14 examples. The performances of the combiners for the three learning sets are summarised in Table 5. Introducing more examples from one filter resulted in an acceptable performance when testing examples from the filter that the training examples were taken from. Instability in the learning occurred for examples generated from different filters. Again, the combiners successfully recognised all the 'end-of-process' examples but their total percentage rate of success was not as high as for the ID3 algorithm.

Table 5: Combiner predictive accuracy for Search 2 and reduced set of attributes (four positive peaks) and scaled values (0–100)

Number of learning examples	% rate of success on			
	S_1	S_2	S_3	*Total*
5	100	93	75	78
9	100	93	81	83
14	93	95	58	64

3.7 Search 3 comparison

Problems arose when ID3 was implemented for Search 3. It is not reasonable to expect a prediction of, say, 2.25 when only examples with 0.25 and 2.75 classes were presented. This implies the necessity of a large training set consisting of all the examples of one filter. However, due to the large number of classes (11), the problem of bushy, unstructured decision trees arose and this resulted in a very poor performance. The inability of ID3 to perform successfully when a large number of classes are present was the main factor in deciding to split the search into three separate searches, as reported previously.

The main advantage of the combiner architecture over that of ID3 is due to its ability to produce continuous output. For this search space, it was decided to train the combiners on the exact values of misadjustments for both screws. Again, the reduced set of attributes and the scaled values were used. Figs. 5*a* and 5*b* show the correct misadjustment levels for screws C_4 and C_7, respectively. Figs. 5*c* and 5*d* illustrate the output of the combiners when five learning examples were used, i.e. one 'end-of-process' and four for the maximum misadjustments of the screws. Figs. 5*e* and 5*f* show the same outputs when nine learning examples were used and, finally, Figs. 5*g* and 5*h* show the outputs with 14 learning examples. Although not one hundred per cent accurate, with this limited number of examples the combiners have managed to track the desired outcomes (Figs. 5*g* and 5*h*) as evidenced by the similarities between Figs. 5*g* and 5*a* and between Figs. 5*h* and 5*b*.

3.8 Discussion of performance

The difference between the two techniques, apart from the algorithmic approach used, is that adaptive combiners learn in an incremental fashion whereas ID3 sees all the examples at the same time. For Search 1 both techniques showed a tendency to improve their total performance when further examples were introduced. ID3 performed as well and sometimes better than the combiner. Also both techniques were plagued with the same problem of not recognising the 'end-of-process' class. Finally, this problem was solved (see Section 3.5) when it was found that ID3 sometimes gave slightly better results with less inconvenience. To produce an acceptable performance with adaptive combiners a considerable amount of time had to be spent manipulating the number of attributes, the format of the attribute values and even the order of introduction of the examples.

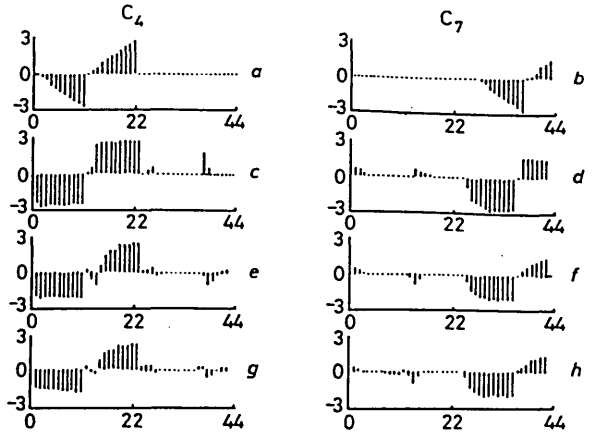


Fig. 5 *Output of combiners (search 3)*

Two interesting points arose for Search 2. Firstly, the performance of ID3 was independent of the number of examples, and secondly ID3 had a higher percentage testing success than the combiner. The reason behind the much lower total performance of the combiner lies in the low percentage rate of success when examples of other filters are tested (S_3 in Table 5). It is known that two filters of the same family are not identical. Tolerancing errors and parasitic effects result in different attribute values. It seems that the combiner could not handle these situations whereas selected cutoff points of ID3 divided the N-dimensional space of the N-attributes properly.

ID3 failed significantly for search 3. This was due to the large number of classes (11) together with the relatively small number of examples (43). Less than four examples contributed to each class. Even the introduction of a larger training set did not ensure success. The ID3 algorithm was originally constructed to deal with binary classification and it seems that better performances are achieved with a low number of classes. On the other hand the ability of adaptive combiners to handle continuous output produced better results with fewer examples. To improve the performance of the combiner for this search space, it would be necessary to include learning examples generated when both screws are misadjusted together. Additionally, combiners can be used to indicate both the magnitude and the direction of the adjustments of both screws, therefore eliminating the need for three search spaces. However, this is not possible if one requires the use of a training set which might be incomplete.

Some general points can be made which apply irrespective of the application. Unlike the combiners, ID3 always gave correct predictions for the examples used in the training set. ID3 also generated decision trees which

could be transformed in the form of IF ... THEN rules. These rules can be used directly to explain the relationships between the attributes and the decisions made. Using adaptive combiners the knowledge is represented in the weights and direct explanation is not feasible. Some work along these lines has been reported [19, chap. 5]. ID3 runs (i.e. learns) faster than the combiners. A drawback in using ID3 is that introducing further examples means the regeneration of a decision tree that may result in changes in the cutoff points and/or the attributes used etc. The experiments reported in the text below attempted to deal with this problem. New examples will not effect the structure of the combiners and only the weights will be updated.

4 Comparison of decision trees generated using ID3

The use of numerical attribute values resulted in a problem associated with the cutoff point. The algorithm produced rules of the form

IF attribute X is less than cutoff point T THEN ..

The cutoff point, which took values such as 1.39765, was calculated using those values that were currently present in the training set. When new examples with previously unseen values were introduced, in most cases the cutoff point changed, resulting in a new set of rules.

In this Section the results obtained in an attempt to identify any advantages in using one attribute presentation form over another are reported. The investigation involved the evaluation and comparison of decision trees produced by using logical and numerical attribute values for the first two searches.

4.1 Further selection of attributes and generation of examples

It was considered that the inclusion of further attributes might be helpful. In total, seven more attributes were introduced. These took the form of the six differences between positive peaks, for example $p_1 - p_2$, $p_3 - p_4$, and the difference between the two negative peaks. A new set of examples was collected. This time the 'tune' procedure was employed. This was necessary since, although the original 'detune' data was applicable, the previous comparison omitted the heuristics (short cuts) of the operator. Therefore, the operator was requested to tune a number of filters and the data were recorded as previously. In this way 34 filters were tuned (only in the stopband) resulting in 138 examples.

4.2 Generation of logical values

Schemes have been proposed [20] which attempt to define supplementary cutpoints for each cutoff point. Producing such confidence intervals enhances the classification of examples with values near the cutoff points. An alternative scheme was followed in our work. Instead of using the raw numerical values a transformation was applied. The numerical values were placed into ranges which were given logical names. Due to the absence of *a priori* knowledge for determining the ranges within which attribute values must lie for the filter to be considered tuned, ranges were calculated by employing the mean m and the standard deviation sd value of each attribute [21] using only those examples with 'end-of-process' as their class. In this way eight (ok, farleft, farright, closeleft, closeright, left, right, absent) or four (ok, left, right, absent) logical values were generated and assigned to each numerical value. Thus, three sets of examples were available for each search space (i.e. numerical, 8-logical, 4-logical). The label 'absent' was used when a value for an attribute could not be determined (i.e. when a peak was absent) and not because it was unknown. The 'ok' label was given to those values which lay within the range $(m - sd)$ to $(m + sd)$. Furthermore, values within the range $(m - 2sd)$ to $(m - sd)$ were labelled 'closeleft', which in the case of 4-logical values were assigned 'left' etc.

4.3 Criteria for the evaluation of decision trees

The evaluation and comparison was based on the following criteria:

(i) percentage errors on classifying unseen examples
(ii) number of branches in the decision tree
(iii) number of rules in the rule base
(iv) number of clash-labelled leaves
(v) number of empty-labelled leaves
(vi) total number of preconditions in the rule base.

The first criterion assessed the performance of a decision tree in terms of accuracy in classifying unseen examples. This indicated how good the generalisation was. The rest of the criteria are of secondary importance and can be applied to determine the complexity and intelligibility of a decision tree. Fig. 6 displays a decision tree and Table 6

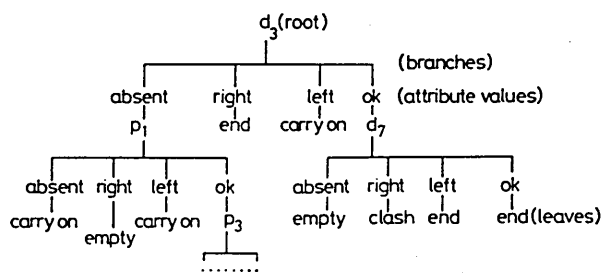


Fig. 6 *Subset of a decision tree generated with four-valued logical attributes*

Table 6: Set of rules produced using the decision tree of Fig. 6

IF	d_3 is absent	IF	d_3 is left
AND	p_1 is absent	THEN	class is carry-on
THEN	class is carry-on		
IF	d_3 is absent	IF	d_3 is ok
AND	p_1 is left	AND	d_7 is left
THEN	class is carry-on	THEN	class is end
IF	d_3 is right	IF	d_3 is ok
THEN	class is end	AND	d_7 is ok
		THEN	class is end

the equivalent set of rules. They both illustrate the terms used in the criteria.

4.4 Presentation of tuned examples

The objective of this part of the work was the identification of the 'best' configuration for the two first search spaces. By 'configuration' is meant the choice of attributes to be used and their format (e.g. numerical, logical). The six configurations used are summarised in Table 7. To test how well the six configurations measured up to the criteria, the available examples were divided into three randomly chosen batches. The first batch included 42 examples, the second 43 and the third 53. Initially, the first batch was used as the training set and the other two

Table 7: Configurations key

Configuration number	Description	Number of attributes
F_1	numerical attributes	13
F_2	4 logical-value attributes	13
F_3	8 logical-value attributes	13
F_4	numerical attributes	6
F_5	4 logical-value attributes	6
F_6	8 logical-value attributes	6

as the testing set (Test 1). This was followed by introduction of the second batch into the training set, which was then tested against the third batch (Test 2). Both tests were evaluated for all configurations for each search space. In total, 54 decision trees were generated, i.e. 18 per search space (Fig. 7).

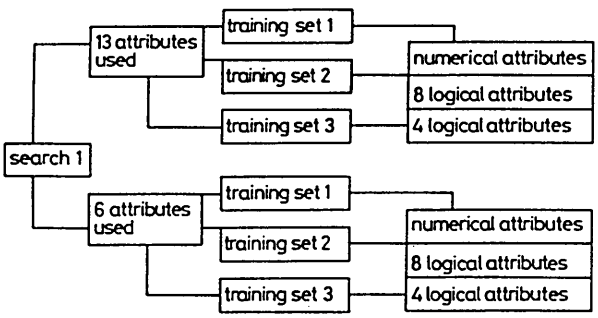


Fig. 7 *Configuration of generated decision trees*

4.5 Evaluation of results and discussion (Search 1)

Table 8 shows the results for each configuration for both tests, expressed as the percentage error of misclassification. From Table 8 the following can be established:

Table 8: Misclassification errors

Configuration	Test 1 % error	Test 2 % error	Classification improvement, %
F_1	42.7	41.5	1.2
F_2	31.3	22.6	8.7
F_3	20.8	28.3	−7.5
F_4	42.7	41.5	1.2
F_5	31.3	26.4	4.9
F_6	27.1	22.6	4.5

(i) All performances but one improve as the size of the training set increases.

(ii) The amount of classification improvement varies between configurations. Trees generated using logical-value attributes seem to perform better than those produced using numerical attributes. The drawback of numerical-value decision trees is their inability to handle examples with absent attribute values.

(iii) Upon increasing the number of attributes no major differences are seen with configurations F_5 and F_6 in terms of improvements in their classification capabilities.

(iv) With 13 attributes it can be seen that the performance improves further with the F_2 configuration.

The results suggest the use of logical values for Search 1. Furthermore, 13 attributes, each one expressed with four logical values, tend to produce better results.

For the algorithm to be effective, the number of situations in which knowledge does not appear to have been learned (empty leaves) or there are contradictions (clash leaves) must be kept to a minimum. If either number is large a poor performance during testing results.

Table 9 shows the results obtained using these two criteria. It is worth noting that when numerical attributes are used there are neither empty nor clash situations.

Table 9: Number of leaves with 'empty' or 'clash' label

Configuration	Test 1/ Batch 1 number of leaves		Test 2/ Batch 1 + 2 number of leaves		Test 3/Batch 1 + 2 + 3 number of leaves	
	empty	clash	empty	clash	empty	clash
F_1	0	0	0	0	0	0
F_2	6	0	15	3	29	2
F_3	16	0	66	3	100	2
F_4	0	0	0	0	0	0
F_5	6	0	14	4	24	8
F_6	20	0	58	4	69	6

This is to be expected. With numerical values the algorithm branches using cutoff points which inevitably cover every example. It is also unlikely for a clash to occur when numerical values with six significant figures are employed. The conclusion that can be made here is that by increasing either the number of attributes (from 6 to 13) or the number of logical descriptors (from 4 to 8) an increasing number of empty situations is generated. This is because the use of a large number of attributes or attribute values renders the training set less representative. This is influenced by a recognised drawback of the algorithm [22]: the algorithm has no means of determining if it is necessary to branch for all defined values of an attribute. Attributes and attribute values are sometimes, but not always, relevant. When the number of attributes was six the number of clashes increased as further examples were introduced. With a larger set of attributes the number of clashes tended to stabilise, irrespective of the number of logical values.

The analysis of the results obtained for the remaining criteria (number of rules, branches, preconditions) were also complicated by the problem of irrelevance branching. Decision trees generated using numerical attributes produce a smaller number of branches, less preconditions and fewer rules.

4.6 Selection of configuration for Search 1

Taking into account all the criteria, with equal weighting attached to each, suggested the use of numerical attributes since they produced smaller trees etc. However, the most important criterion, namely the percentage of errors in the classification of unseen examples, showed the use of numerical values to be unsatisfactory. The misclassification error of approximately 42% was too large to be ignored. The use of logical values resulted in a more acceptable error rate. It was necessary to select between the choice of 6 or 13 attributes. There was not much difference between their performances as far as the secondary criteria were concerned, but the use of F_2 almost doubled the classification improvement. Therefore F_2 was selected as the most promising configuration. Further work with the chosen configuration resulted in an improvement in the performance of the secondary criteria. By taking into account *a priori* knowledge, the clashes were eliminated and the number of leaves was greatly reduced. Pruning the rule base [23] resulted in a reduction of preconditions and rules while maintaining the same performance.

4.7 Selection of configuration for Search 2

A similar analysis took place for the second search space and configuration F_6 was found to perform best.

5 Conclusions

Advances have been made in applying the techniques of expert systems with rule induction by the ID3 algorithm and of adaptive combiners to the tuning of the stopband of crystal filters.

ID3 was chosen as the preferred technique for the first two searches. The main advantages of the ID3 algorithm over adaptive combiners were faster learning, the generation of better results, and less manipulation of the attributes. Additionally, with ID3, decision tree rules were generated which made the relationships between the attributes more visible, the order of introduction of examples was not critical, and, finally, a 100% correct prediction for the training set was obtained. These advantages did not hold for the third search, where a larger number of classes was present. Further work with ID3 indicated that it was more efficeint to use logical than numerical attribute values when testing unseen examples. Logical configurations were identified for each search space. It seems that ID3 is a valuable aid to the knowledge elicitation stage and, particularly, when a small number of classes and logical values are present.

6 Acknowledgment

The authors are grateful to expert operator Lawrence Mack of Newmarket Microsystems and to Dr. Anne Hart of Lancashire Polytechnic for helpful discussions on the use of the ID3 algorithm.

7 References

1 *IEEE Computer*, 1986, **1**, (7)
2 MIRZAI, A.R., COWAN, C.F.N., and CRAWFORD, T.M.: 'Intelligent alignment of waveguide filters using a machine learning approach', *IEEE Trans.*, 1989, **MTT-37**, (1), pp. 166–173
3 NAZEMI, J., and FIDLER, J.K.: 'Filter tuning using a microprocessor based heuristic algorithm'. Proc. 1985 European Conf. on Circuit theory and design, pp. 101–104
4 ERICSSON, K.A., and SIMON, H.A.: 'Protocol analysis: verbal reports as data' (Bradford Books/MIT Press, 1984)
5 REDDY, R.: 'Foundations and grand challenges of artificial intelligence', *AI Mag.*, 1988, **9**, (4), pp. 9–21
6 GAMMACK, J.G., and YOUNG, R.M.: 'Psychological techniques for eliciting knowledge', *in* BRAMER, M.A. (Ed.): 'Research and development on expert systems' (Cambridge University Press, 1985)
7 MICHALSKI, R.S.: 'Understanding the nature of learning', *in* MICHALSKI, R.S., CARBONELL, J.G., and MITCHELL, T.M. (Eds.): 'Machine learning: an artificial intelligence approach. Vol. 2' (Morgan Kaufmann, 1986)
8 POLITAKIS, P.G.: 'Empirical analysis for expert systems' (Pitman, 1985)
9 LIPMANN, R.P.: 'An introduction to computing with neural nets', *IEEE ASSP Mag.*, April 1987, pp. 4–22
10 GOLDBERG, D.E.: 'Genetic algorithms in search, optimisation and machine learning' (Addison-Wesley, 1989)
11 MICHALSKI, R.S., and CHILAUSKY, R.L.: 'Knowledge acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology', *Int. J. Man-Mach. Stud.*, 1980, **12**, pp. 63–87
12 NEWSTEAD, M.A. and PETTIPHER, R.: 'Knowledge acquisition for expert systems', *Electr. Commun.*, 1986, **60**, (2), pp. 115–121
13 QUINLAN, J.R.: An empirical comparison of genetic and decision tree classifiers'. Proc. Fifth Int. Conf. on Machine learning, Morgan Kaufmann, 1988, pp. 135–141
14 RUMELHART, W.E., HINTON, G.E., and WILLIAMS, R.J. (Ed.): 'Parallel distributed processing: explorations in the microstructure for cognition. Vol. 1' (MIT Press, 1986)
15 TSAPTSINOS, D., MIRZAI, A.R., and JERVIS, B.W.: 'Comparison of machine learning paradigms in a classification task'. Proc. Fifth Int. Conf. on Applications of artificial intelligence in engineering, Computational Mechanics Publications, July, 1990
16 QUINLAN, J.R.: 'Discovering rules from a large collection of examples: a case study', *in* MICHIE, D. (Ed.): 'Expert systems in the micro-electronic age' (Edinburgh University Press, 1979)
17 QUINLAN, J.R.: 'Induction of decision trees', *Mach. Learning*, 1986, **1**, pp. 81–106
18 BROWN, K.E., COWAN, C.F.N., CRAWFORD, T.M., and GRANT, P.M.: 'Knowledge-based techniques for fault detection in digital microwave radio communication equipment', *IEEE J. Sel. Areas in Commun.*, 1988, **6**, (5), pp. 819–827
19 MIRZAI, A.R. (Ed.): 'Artificial intelligence: concepts and applications in engineering' (Chapman & Hall, 1990)
20 QUINLAN, J.R.: 'Decision trees as probabilistic classifiers'. Proc. Fourth Int. Workshop on Machine learning, 1987, pp. 31–37
21 GIORDANA, A., and SAITTA, L.: 'An expert system oriented to complex pattern recognition problems', *Inf. Sciences*, 1985, **36**, pp. 157–177
22 CHENG, J., FAYYAD, U.M., IRANI, K.B., and QIAN, Z.: 'Improved decision trees: a generalised version of ID3'. Proc. Fifth Int. Conf. on Machine learning, Morgan Kaufmann, 1988, pp. 100–106
23 QUINLAN, J.R.: 'Simplifying decision trees', *Knowledge-based Syst.*, 1988, **1**, pp. 241–254

PRACTICAL ASPECTS OF USING AN EXPERT SYSTEM-NEURAL NETWORK HYBRID SYSTEM FOR TUNING CRYSTAL FILTERS

D. Tsaptsinos†, B.W. Jervis†, A.R. Mirzai

Sheffield City Polytechnic†, Polytechnic of Central London, UK

INTRODUCTION

A knowledge-based system in a rule format, has been developed in order to help an operator during the post-assembly tuning of crystal filters. The generation of the rules was accomplished using the ID3 learning by examples algorithm. A consultation with the system provides the operator with advice as to whether the filter is tuned or as to which screw to turn and in which direction. Unfortunately it was not possible to use ID3 to generate rules for the distance to turn. The distance can be any value in the range of 0 to 2.5 revolutions inclusive and ID3 cannot handle such a large number of classes (Tsaptsinos et al (1)). It is therefore left to the operator to judge how far to turn the screw. Initial testing results were obtained with an experienced operator who had some idea of how far to turn the screw. The objective though was to construct a system which could be used by anyone irrespective of his level of experience and proficiency. An inexperienced operator would probably turn the screws too far or too little. This could result in a larger number of iterations and while the tuning would eventually be done it would take longer. For this reason, neural networks were investigated in order to provide the operator with an indication of how far to turn the screws. The results below are for one sub-process of tuning, namely the stopband tuning of the filter. For this sub-process two adjustable components are used, C_4 and C_7.

GENERATION OF THE EXAMPLE SET

A number of filters were de-tuned. De-tuning means moving from a tuned response to an untuned one. The examples were created by having C_7 either at its optimum position (i.e. where it was placed when the expert finished the tuning) or maladjusted in steps of half a revolution up to 1.75 revolutions in a clockwise direction or up to 2.5 revolutions in an anti-clockwise direction. At each position of C_7 the other component C_4 was maladjusted in steps of half a revolution from its optimum position up to 1.5 revoulutions in a clockwise direction or up to 2.5 revolutions in an anti-clockwise direction. In total 358 examples were generated for each filter. Each example consisted of fifty seven sampled values of the amplitude response (dB) plus the class to which it belongs (i.e. the distance turned).

Representing the examples

The 57 sampled amplitude data were presented to the neural network without transformation. The class value for each example was coded with a value between 0 and 1. For example, real values of 0, 0.25, 0.50 became 0, 0.1, 0.2 respectively.

Determining the learning set

Various questions arose prior to the neural network implementation concerning the size of the learning set. For example, should the learning set include examples generated from different filters, and/or should it include examples covering de-tuning of both components etc.? This section deals with these questions. The variety of the position of the responses which can be considered as tuned created an overlapping of classes. For this reason, it was decided to employ the de-tune data of just one filter. This will force the tuning of other filters towards the model

'solution'. Additionally, in some cases, maladjustment by more than two revolutions caused negligible changes in the response. Overlapping of classes similarly occurred when the complete learning set was used. Responses generated using the left component (C_4) with, say, 0.5 turns resembled the ones generated using the right component (C_7) with 1.25 turns. For this reason it was thought appropriate to break the learning set into four sets (Table 1).

TABLE 1 - Learning sets

Learning maladjustment	Number of examples
C_4 anti-clockwise	215
C_4 clockwise	144
C_7 anti-clockwise	216
C_7 clockwise	178

NEURAL NETWORK ARCHITECTURE

Software from a commercially available package was used to simulate the learning algorithm on a 80386 based computer.

A three layer feedforward network (57-11-10-1) using the generalised delta learning rule (Rumelhart et al (2)) was employed for each of the four learning sets. The learning rate and the momentum term were set at 0.9 and 0.6 respectively. The number of processing units in the input layer was set to 57, thus each sampled point was assigned to one and only one unit. The input of each input unit was subjected to a simple linear transformation using the software package, of the following form

Transformed input = Input value * Scale factor + Offset

where the values of 0.01 and 0.1 were used for the scale factor and offset respectively. The offset was used to avoid having any zero inputs.

The number of processing units in the output layer was set to 1. The output of the single unit is simply the summation of all its inputs, multiplied by their associate weights, from the second hidden layer. The obtained result was limited to both an upper (1.0) and lower (0.0) bound and then compared to the desired output (i.e. how far to turn). Using the software package learning was inhibited when the error was lower than a pre-set value.

The sigmoid function was used as the transfer function for the two hidden layers. The selection of the number of processing units for each hidden layer was not as natural and effortless as for the other layers. Their numbers were determined empirically (11 and 10 for the first and second hidden layer respectively) and no claim is made that they are the most appropriate. Initial connection weights were set to small random values and they were updated after each presentation of an example.

NEURAL NETWORK IN LEARNING MODE

The stop learning criterion

For each learning set the network was executed for 75000 runs. Every 1000 runs the learning was momentarily paused and the total sum of the squared errors was calculated. The error was the absolute difference between the desired and the obtained value. Those weights which generated the smallest error were selected for the network. For example, when using the learning set generated with C_4 maladjusted anti-clockwise from the tuned position the smallest error occurred in run 56000. One of the criteria used to test the suitability of the above network based on anti-clockwise maladjustments was to test the network with de-tuned examples obtained by maladjusting C_4 in a clockwise direction. The expected

outcome was 0 (since C_4 does not need maladjustment in an anti-clockwise direction) and for the majority of the test cases a value close to zero was produced.

NEURAL NETWORK IN TEST MODE

Definition of test cases

Testing of the tuning of a number of filters was undertaken using three different systems.

System 1: Knowledge-based system plus user
The knowledge-based system provided advice on when to stop the tuning of the stopband, otherwise which component to turn and the direction to turn. The user had to decide on how far to turn.

System 2: Hybrid system As for system 1 but the distance to turn was indicated by the appropriate net. For example, if the expert system indicated C_4 clockwise, then the C_4 clockwise neural network would be used.

System 3: Neural network Because each component/direction combination had a net associated with it then the outcome of each net was used to define all decision levels. For example, if the output of the four networks were:
C_4 anti-clockwise network : 0.1 (i.e. 0.25 in real turns)
C_4 clockwise network : 0.3 (i.e. 0.75 in real turns)
C_7 anti-clockwise network : 0.6 (i.e. 1.50 in real turns)
C_7 clockwise network : 0.5 (i.e. 1.25 in real turns)
then the largest of each component was selected. In this example, that would had meant turn C_4 clockwise 0.75 turns and C_7 anti-clockwise 1.50 turns.

Test evaluation criteria

The following criteria were employed to compare the various systems.
(i) The average number of turns required for the entire tuning
(ii) The number of successful tunings
(iii) The number of unsuccessful tunings
In this paper the term tuning refers to the stopband tuning. It is worth noticing that the final result (i.e. the tuning) was examined rather than the intermediate actions. This was due to the fact that there exist numerous paths to the tuning and only the prominently wrong actions could be identified.

PRESENTATION AND DISCUSSION OF RESULTS

Table 2 shows the number of attempts made for tuning the stopband. Table 3 shows the comparison of the systems in terms of the number of turns required. The table compares the performances of the systems and of the human operator.
The comparison shows that the use of any system did not necessarily reduced the number of steps but the expected benefit will be a reduction of the time an operator spends learning about the tuning procedure. This is apparent when comparing system 1 and system 2. The results are comparable and encouraging. There is no need to have an experienced operator. At this stage it is preferable to use system 2 rather than system 3. The latter system seems to require more steps. There are two probable reasons for this. Firstly the shortcomings of the C_7 anti-clockwise network as experienced during all testing and secondly the attempt of each network to target to a single model solution. This resulted in oscillating outputs.
The neural networks were also tested with data where the desired outcome was known beforehand. Observing the output of the neural networks the following points were made:

(i) The networks for learning C_4 and C_7 clockwise both gave correct estimates
(ii) The network for learning C_4 anti-clockwise tends to underestimate for values greater than 0.5
(iii) The network for learning C_7 anti-clockwise did not perform well in general except in one case in which it worked correctly for values up to 0.7 but for greater values it provided conservative estimates
(iv) All networks recognise a tuned state
(v) Networks for learning the clockwise maladjustment for both components operated better. Both had fewer examples in their learning sets and less classes represented than the ones with anti-clockwise maladjustments.

CONCLUSIONS

From the test results shown above, it should be noted that it is possible for the hybrid system (system 2) and the connective equivalent (system 3) to tune the stopband region of the magnitude response. A decrease in the training of operators can be achieved with either system.

However, each system has its own advantages. The case 2 system can generate basic explanations of its reasoning whereas the networks have a faster execution time despite the larger number of steps taken. Both systems are then promising but an extensive testing period would be required before they be introduced in the production line.

REFERENCES

1. Tsaptsinos, D., Mirzai, A.R., Jervis, B.W., and Cowan, C.F.N, 1990, "Comparison of knowledge elicitation techniques in the domain of electronic tuning", IEE Proceedings, 137, 5, 337-344.

2. Rumelhart, D.E., McClelland, J.L., and the PDP Research group, 1986, "Parallel distributed processing: Explorations in the Microstructure of cognition", MIT Press, Cambridge, USA.

TABLE 2 - Number of test filters

	Manual	System 1	System 2	System 3
Number of attempts	34	21	19	3
Successful tunings	34	18	15	2
Unsuccessful tunings	0	3	4	1

TABLE 3 - Comparison of performances

	Manual	System 1	System 2	System 3
Average number of turns	3.67	3.22	3.53	10.50
Minimum number of turns	1	1	1	7
Maximum number of turns	9	7	8	14

COMPARISON OF MACHINE LEARNING PARADIGMS IN A CLASSIFICATION TASK

D. TSAPTSINOS , A.R MIRZAI[1] , B.W JERVIS
Department of Electrical and Electronic Engineering, Sheffield City Polytechnic, Sheffield S1 1WB, UK
[1]School of Mechanical Engineering, Polytechnic of Central London, London W1M 8J5, UK

ABSTRACT

Filters can be used in a variety of applications. In practice, manual tuning of the components is required in order to achieve a specified performance. An expert system is being constructed to assist the operator during the tuning phase. Protocol analysis was employed originally but failed to provide the whole spectrum of the operator's knowledge. Experiments were carried out to investigate and compare the applicability of three machine learning paradigms (ID3, adaptive combiners, neural nets) as the means of automated knowledge elicitation. A brief description of the techniques, a comprehensive analysis of the experiments and the reasons behind ID3's selection are described in this paper.

INTRODUCTION TO THE DOMAIN

At present, the tuning of electronic filters is performed manually. The objective of our work is to develop the expert system paradigm in this domain in order to partly or completely automate the process.

Electronic filters are available in various types but irrespective of the type the function of an electronic filter remains the same. That is, to retain all frequencies within certain limits (passband regions), while rejecting all other frequencies (stopband regions).

A produced filter will not always meet the desired specification, thus manual tuning is required to adjust component values to achieve the required performance. There does not appear to be a theory of the practical tuning of filters. Through an initial training and with acquired experience the operator is transformed into a skilled operator. Despite the variations between operators, which can be found in

detail, the general pattern is the same. An operator checks the performance of the filter (eg. magnitude response [Figure 1]), decides if tuning is required and, if so, performs the necessary adjustments to a tunable component. These steps are then repeated as many times as necessary until the performance satisfies the requirements. Effectively the operators act as signal interpreters and perform a human real-time optimisation attempting to reduce the total and individual errors of the features of interest.

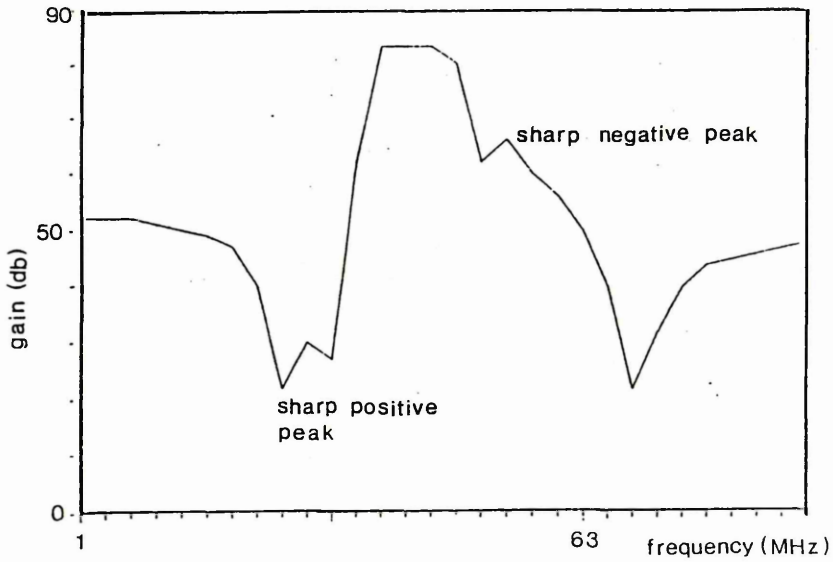


Figure 1 Normalised magnitude response

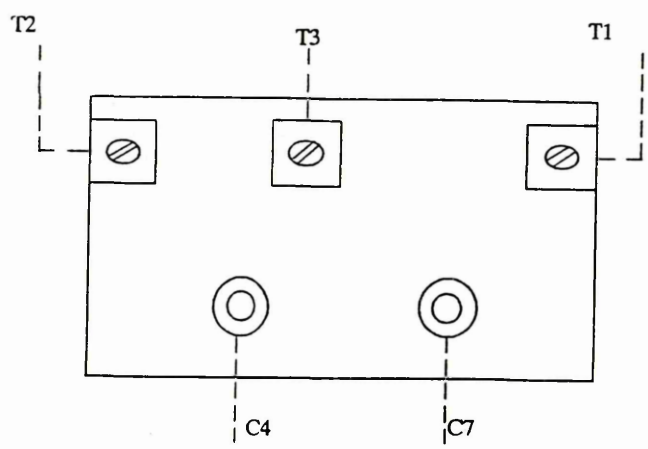


Figure 2 Top level view of filter

The filter employed as the test benchmark is an asymmetric bandpass crystal filter [Figure 2]. The filter consists of two types of adjustable components, namely trimmer capacitors (C4 and C7) and inductors (T1, T2, T3).

INITIAL KNOWLEDGE ELICITATION

Several expert systems have been constructed (Bramer [1]), and their number is rapidly increasing for engineering domains (Computer [2]). The construction of an expert system is a painstaking process. Eliciting, analysing, interpreting, representing, administering and utilising the knowledge that a human expert uses presents problems to the builder.

The knowledge elicitation technique chosen for the filter-tuning project was the protocol analysis approach. Protocol analysis was first described by Newell and Simon [3] and, in recent years, by Ericsson and Simon [4]. The reason to employ a verbal on-line technique becomes more visible when one considers the nature of the decision process. The operator interprets, plans and executes tasks by visually inspecting the measurement set which displays the magnitude response of the connected filter. By adjusting the set the operator is able to inspect the full response or part of it. It was felt that the operator would have found it impossible to describe the response in a verbal off-line format.

The tuning of the chosen filter was then video-taped twice. The expert was instructed to "think-aloud" about the process and to include not only his mental skills but also his manual skills. Manual means the skills needed to operate the measuring set. Mental means the reasons behind each action taken, such as why to turn component X instead of Y. Then the process of transcribing the video-tapes and analysing the transcripts commenced. The benefits of the transcription analysis were as follows :-

Identification of order for specification checking ie. what features and in what order were checked. Possible corrective actions were also identified.

Identification of the classes of activity the operator was involved with.

Identification of objects which resulted in the production of a dictionary. By objects is meant the most primitive words that the expert uses to express domain knowledge (eg. frequency, coils etc.).

Clustering of objects and identification of their relations.

Identification of the expert's tuning process.

What failed to surface is best described by the following scenario. It was identified that the expert begins with the tuning of the stopband region of the magnitude response. Also, it was found that only the trimmer capacitors were of any use for this region. Although there are only two candidates the expert failed to provide a theory for which one to pick, which direction to tune and by how far to turn at certain situations. To overcome that problem the possibility of automatically acquiring and updating the knowledge was considered. This involved the implementation and comparison of three paradigms which learn through the use of examples.

GENERATION OF EXAMPLES

The three techniques (ID3, adaptive combiners and neural nets), function according to a similar principle. They require a set of examples, referred to as the learning set. Each example is described in terms of attributes, with each attribute in turn specified by a value, together with a class identifier. The purpose of the techniques is to determine the relationships between the attributes which then can be used for classification of other examples.

A database of examples was not readily available, therefore a set of examples was collected using the "de-tune" procedure. This procedure misses out the heuristics employed by the expert but a more complete set of examples can be collected. By complete is meant a learning set which contains most attribute values likely to arise thus eliminating the possibility of having only extreme or rare values.

Prior to the generation of the learning set using the previous transcript analysis and further discussions with the expert the following suitable attributes were identified:-

(i) Locations of sharp positive peaks of the waveform (number=4, MHz units)
(ii) Relative magnitudes of sharp negative peaks of the waveform (number=2, dBs units).

During the de-tuning the expert was asked to tune the stopband region. Then the attribute values were recorded together with the class "end-of-process". A systematic de-tuning followed. That was achieved by keeping one tunable component constant and mis-adjusting the other one. This was reapeated by misadjusting in a different direction and by using the other component. Obviously, the filter was re-tuned in between. For those examples the component, direction of turn and how far the component was turned were recorded. In this way 43 examples were collected for one filter. Six filters were de-tuned resulting in a total of 258 examples.

THE ID3 ALGORITHM

ID3 (Iterative Dichotomiser 3) was developed by Quinlan [5] in 1979. The goal of the algorithm is to induce a decision tree (which can easily be transformed into rules of the form IF x THEN y). A decision-tree is then generated from a collection of examples by recursively sub-dividing this collection into smaller subsets. A decision tree consists of a number of nodes (the "IF" part) representing attribute-based tests together with a number of terminal nodes, also known as leaves. The terminal nodes (the "THEN" part) may take the label of a class, or be labelled "empty" or "clash". Empty appears when there are no examples which can be used for that particular branch. Clash emerges when there are two (or more) examples covering that specific branch but their classes are distinct.

In order to illustrate the technique, a simple example will be used. The objective is to obtain rules to help us identify the class to which a filter belongs. By class is meant highpass, lowpass, bandpass or bandreject. Those classes are then the outcomes. For attributes the following were used:

(i) Number of stopband regions (abbreviation: nostop)
(ii) Number of passband regions (abbreviation: nopass)
(iii) Number of transition regions (abbreviation: notrans)

Table 1 shows the examples used. Using a commercial package Xi-

NOSTOP	NOPASS	NOTRANS	OUTCOME
one	one	one	highpass
one	one	one	lowpass
two	one	two	bandpass
one	two	two	bandreject

Table 1. Examples

Rule, which implements the algorithm, the decision tree in Figure 3 was generated. Certain observations can be made:-

```
notrans  = two
   |—— nostop = one: bandreject
   |        |—— = two: banpass
   |—— = one (Clash)
```

Figure 3 Decision tree

(i) When the number of transitions is less than two, a clash exists. This is due to the fact that the first two examples have the same attribute values but different classes. This

means that more attributes are needed in order to discriminate between highpass and lowpass filters.

(ii) The attribute "number of passband regions" is redundant. The attribute can be eliminated and the rules will be the same.

From the decision tree rules can be generated by simply following a branch through the tree to one of the leaves. Table 2 contains the two rules extracted from the decision tree in

```
IF notrans IS two
AND nostop IS one
THEN outcome IS bandreject (Rule 1)

IF notrans IS two
AND nostop IS two
THEN outcome IS bandpass (Rule 2)
```

Table 2. Generated example rules

Figure 3. What has to be remembered is that induced rules do not generate new knowledge but prompt to what the rules appear to be. For example, on observing Table 2, one can see that the condition for attribute "notrans" is not required.

ADAPTIVE COMBINERS

In recent years one class of adaptive architectures, linear combiners, has been used for the design of intelligent systems (Mirzai [6]). Figure 4 illustrates a simple combiner structure.

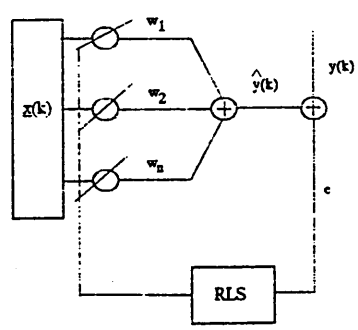


Figure 4 Adaptive combiner architecture

Given knowledge about a particular problem in the form of input attributes, and the class, it is desirable to estimate the weight vector in such a way that, when the system is presented with a new set of examples, it can predict the correct outcome.

The adaptive combiner structure used here can be thought of as a one layer connectionist network and the recursive least squares (RLS) algorithm is employed for the estimation of the weight vector.

NEURAL NETWORKS

Only recently research in neural networks was revived resulting in the development of various techniques (Pollack [7]) which attempt to eliminate the original shortcomings. Back-propagation is a such technique, which was developed independently by several people (LeCun [8], Parker [9], Rumelhart [10]). Figure 5 illustrates a three layer network

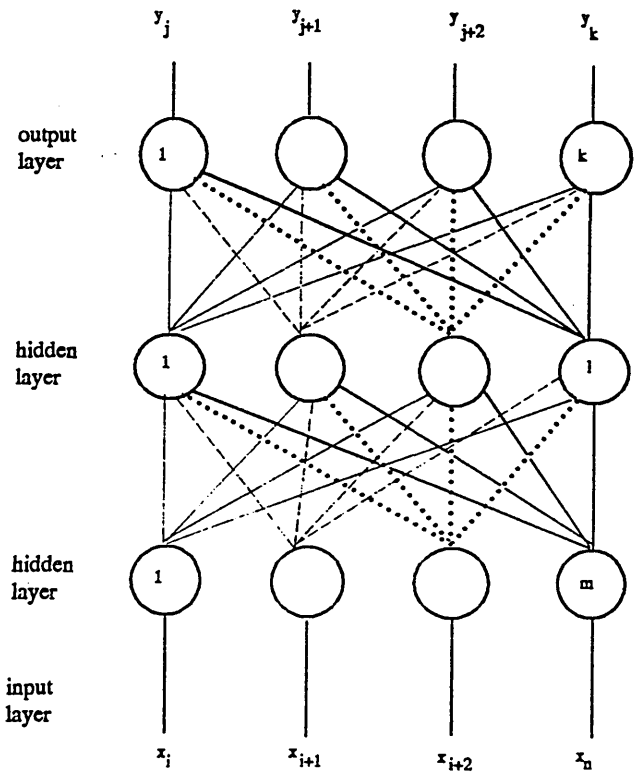


Figure 5 neural net architecture

which consists of an input layer, an output layer and two hidden layers. The input layer contains the information coming from the features (attributes) from each example. The hidden layers perform a recoding of the original feature-set which is then passed to the output layer for the generation of a pattern (class). The characters written on the arrows represent the connectionist strength of the weights. The characters in the circles represent the thresholds of each unit. The back-propagation algorithm uses the sigmoid function which results

in a continuous threshold. By initialising all weights and thresholds to small random values and using the sigmoid function the outputs are calculated. Then the algorithm recursively adjusts the weights and the thresholds. Reference (Rumelhart [10]) provides a more detailed mathematical analysis of the technique.

To illustrate the technique we adopt the example used by Lippman [11]. Eight examples representing two classes 1 and 0 (Table 3) were introduced as the learning set. Examples from

X1	X2	CLASS
1	2	0
1	1	1
-1	3	0
-1	-1	1
-1	-3	0
1	-1	1
1	-4	0
-1	1	1

Table 3. Learning set

class 1 were distributed to the edges of a circle of radius 1 centered at the origin. Examples from class 0 were distributed outside the circle. After 548 iterations using a two-layer network with two hidden nodes the weights and thresholds were calculated. Then the network was tested with eleven previously unseen examples (Table 4). The results were satisfactory for either class except those examples, of class 0, with values close to the circle.

X1	X2	Desired class	Produced class
0	0	1	0.97
3	1	0	0.85
-0.5	-0.9	1	0.96
1.1	1	0	0.92
-0.1	0	1	0.97
-1.1	-1.1	0	0.95
5	5	0	0.04
1.1	1.1	0	0.89
1	1.1	0	0.89

Table 4. Testing set

In the above example and in the investigation the three techniques are trained to function as classifiers. The goal is to learn to classify correctly during training so that in future use they will be able to classify correctly new examples.

PRESENTATION OF EXAMPLES

Previous work resulted in three search spaces for the tuning of the stopband.

(i) search space one: to carry-on or to end the process
(ii) search space two: which component to adjust and which direction
(iii) search space three: how far to turn.

The examples were introduced to the techniques in an incremental fashion. The reader should note that the same examples were presented to each technique for every search. The number of classes were different in each search. Search one has two classes, search two has four classes, search three has eleven classes. Initially 8 examples were used in the learning set. They comprised of 4 "end-of-process" and 4 "carry-on" examples of the same filter. The latter included those examples generated when the components were adjusted to their maximum positions in both directions. Then, 4 more examples were introduced, the ones generated when the components were turned halfway. Finally the 4 examples which arose when the components were adjusted to their minimal positions were presented.

At each stage of the procedure the generated set of rules or weights was tested against the learning set (S1), the remaining unseen examples of the same filter (S2) and the unseen examples of the rest of the filters (S3). Finally, the total performance was calculated (TOTAL). In this way it could be determined how well the algorithms learnt and generalized.

An obtained example took the following form,

1.39678 1.40234 1.41967 1.42003 45 53 C4a0.5

which can be translated as "turn the C4 component anti-clockwise, half a revolution when the attributes have the given values". The exact numbers were presented to the three techniques as above. The presentation of the classes was different for the combiner and neural net. For example, in search three class 0.5 was presented to the neural net as 4 nodes (eg. 0 0 1 0). In search two, class C4a was presented to the adaptive combiner as two nodes (eg. -1 0).

CRITERIA OF PERFORMANCE

The comparison was based on two criteria. The percentage of examples used in the final learning set and the predictive accuracy of the final learning set. The reason for using those two interrelated criteria is that an expert system's knowledge base is constantly refined. This is due to that the correct number and nature of examples to be used is unknown. The need

arises to identify that technique which uses the lesser number of examples in conjunction with a satisfactory performance.

SEARCH ONE COMPARISON

The following points can be concluded regarding the results obtained using ID3 (Table 5). ID3 seems to perform better when a small learning set was used and it is always capable of predicting accurately those examples presented to it in the learning set.

NUMBER OF LEARNING EXAMPLES	(%) RATE OF SUCCESS ON ... S1	S2	S3	TOTAL
8	100	82	80	81
12	100	81	80	81
16	100	100	93	94
18	100	100	96	97

Table 5. ID3 Predictive accuracy (Search I)

The satisfactory success rate achieved when 16 examples were used can be misleading because the success rate was due to the presence of a large number of "carry-on" examples. ID3 predicted succesfully the "carry-on" examples but failed to recognize the "end-of-process" ones. The only option available to improve the ID3 performance was to introduce further "end-of-process" examples. It was found that by increasing the learning set to 18 the objective was achieved (Row 4 of Table 5).

Obtained results employing the adaptive combiner architecture are displayed in Table 6. Despite unstable behaviour of the learning set, the success rate of the total

NUMBER OF LEARNING EXAMPLES	(%) RATE OF SUCCESS ON S1	S2	S3	TOTAL
8	87	57	67	67
12	100	77	82	82
16	75	100	91	91

Table 6. Combiner predictive accuracy (Search I)

test was improved. Unfortunately, like ID3, a large number of "end-of-process" examples were mis-classified. Experiments were carried out to improve the performance of the combiner. That took the form of manipulating parameters and the presentation of the attribute values. Table 7 shows the predictive accuracy of the combiner when the forgetting factor equals 0.9 and the

learning set was presented to the combiner 9 times. Similarly, Table 8 shows the results when the attribute values were re-scaled between 0 and 1. In both cases the misclassification problem was resolved.

FORGETTING FACTOR : 0.9	RE-LEARNING LOOPS : 9			
NUMBER OF LEARNING EXAMPLES	(%) RATE OF SUCCESS ON			
	S1	S2	S3	TOTAL
16	94	100	91	92

Table 7. Combiner predictive accuracy (Search I)
(Adjusted Parameters)

NUMBER OF LEARNING EXAMPLES	(%) RATE OF SUCCESS ON			
	S1	S2	S3	TOTAL
8	100	88	90	90
12	100	91	84	85
16	94	98	85	87

Table 8. Combiner predictive accuracy (Search I)
Scaled Values

There are some obstacles in using neural networks. One does not know how many hidden units are required, with what values to initialise the weights etc. Using Table 9, where some results are displayed, various points can be made. When the

(%) RATE OF SUCCESS ON LEARNING SET (S1)	ARCHITECTURE
72	6-4-4-1
71	6-3-5-1
74	6-4-5-1

Table 9. Neural net predictive accuracy
(Search I - Eight examples)

examples were eight the prediction performance averaged 72 per cent. At the same time 10 out of 24 "end-of-process" examples were mis-classified. An increase of 4 examples produced an average performance of 78 per cent. Again the mis-classification rate of "end-of-process" examples was 50 per cent except when the number of hidden nodes in the first layer was four (Table 10). That was irrespective of the number of nodes in the second layer. The best true classification was

(%) RATE OF SUCCESS ON LEARNING SET (S1)	ARCHITECTURE
77	6-2-2-1
78	6-3-2-1
78	6-4-2-1
77	6-2-3-1
78	6-3-3-1
77	6-4-3-1
77	6-2-5-1
77	6-3-5-1
78	6-4-5-1
78	6-4-4-1

Table 10. Neural net predictive accuracy
(Search I - Twelve examples)

achieved when the 6-4-4-1 architecture was employed. Increasing the examples in the learning set to 16 produced an average performance of 93 per cent with a mis-classification rate of 4 examples out of 24 (Table 11).

(%) RATE OF SUCCESS ON LEARNING SET (S1)	ARCHITECTURE
93	6-4-2-1
91	6-4-3-1
93	6-5-2-1
93	6-5-3-1
93	6-6-2-1
93	6-6-3-1
93	6-4-4-1
93	6-5-4-1
93	6-3-5-1
93	6-4-5-1
94	6-5-5-1

Table 11. Neural net predictive accuracy
(Search I - Sixteen examples)

SEARCH TWO COMPARISON

The three learning sets were introduced to the ID3 algorithm. This time the "end-of-process" examples were replaced with those examples generated with the minimum mis-adjustment. Table 12 shows the results obtained. Note that even when eight

NUMBER OF LEARNING EXAMPLES	(%) RATE OF SUCCESS ON ... S1	S2	S3	TOTAL
8	100	100	88	91
12	100	100	88	91
16	100	100	88	91

Table 12. ID3 Predictive accuracy (Search II)

examples were used the prediction rate was acceptable and that the performance did not improve with the introduction of further examples. This is probably an indication that further attributes are required if better performance is to be achieved.

When the three learning sets used for ID3 were presented to the combiner the results were very poor. The reason being the need to have examples which can act as reference points. That role was played by the "end-of-process" examples. The combiners were trained to indicate the "end-of-process", as well as which screw to adjust and in what direction. The first learning set contained 5 examples, ie. one "end-of-process" plus four examples when the screws were mis-adjusted to their maximum positions. Then the examples corresponding to the minimum positions of the screws were added to the learning set (ie. 9 examples all together) and finally the examples corresponding to the half way mis-adjustments of the screws were added resulting in 13 examples. The performance of the combiners was again poor. For that reason the attribute values were re-scaled as before, and an extra "end-of-process" example was introduced. This greatly improved the performance. The performances of the combiners for the three learning sets are summarised in Table 13. Again, the combiners successfully recognised all the "end-of-process" examples.

NUMBER OF LEARNING EXAMPLES	(%) RATE OF SUCCESS ON S1	S2	S3	TOTAL
5	100	93	75	78
9	100	93	81	83
14	93	95	58	64

Table 13. Combiner predictive accuracy
(Search II - Scaled Values)

The three layer networks produced an average performance of 76 per cent with 6-3-3-2 architecture gaining the highest (80% with 8 examples). Note that adding an extra node at either layer did not produce a better performance. By increasing the examples the performance improved with architecture 6-5-3-2

reaching the highest (92%) using 16 examples. Irrespective of the number of examples and number of nodes used the nets produced a better performance for the direction to turn than the component to be used. Table 14 displays a sample of results.

NUMBER OF EXAMPLES	(%) RATE OF SUCCESS ON S1	Component	Direction	ARCHITECTURE
8	72	72	98	6-4-3-2
8	80	80	97	6-3-3-2
8	77	77	98	6-3-4-2
12	77	77	98	6-3-2-2
12	81	81	97	6-3-3-2
12	74	74	97	6-5-5-2
16	93	93	98	6-5-3-2
16	92	92	97	6-8-4-2
16	87	87	97	6-3-3-2

Table 14. Neural net predictive accuracy (Search II)

SEARCH THREE

Problems arose when ID3 was implemented for search three due to the presence of a large number of classes (11). The necessity of a learning set consisting of all examples of one filter produced the problem of bushy, unstructured decision trees arose which resulted in a very poor performance. The inability of ID3 to perform succesfully when a large number of classes are present was the main factor in deciding to split the search into three separate searches, as reported previously.

The main advantage of the combiner architecture over that of ID3 is due to its capability of producing continuous output. For this search space experiments were carried out with unscaled values as well. When re-scaling took place the combiner performance improved. Again "end-of-process" examples were required. Figure 6a and 6b show the correct mis-adjustment levels for screws C4 and C7 respectively. Figure 6c and 6d illustrate the output of the combiners when 5 learning examples were used (ie. one "end-of-process" and four for the maximum mis-adjustments of the screws). Figure 6e and 6f show the same outputs when 9 learning examples were used and finally 6g and 6h show the outputs with 14 learning examples. With this limited number of examples the combiners have managed to track the desired outcomes (Figure 6g and 6h) although not to 100 per cent accuracy.

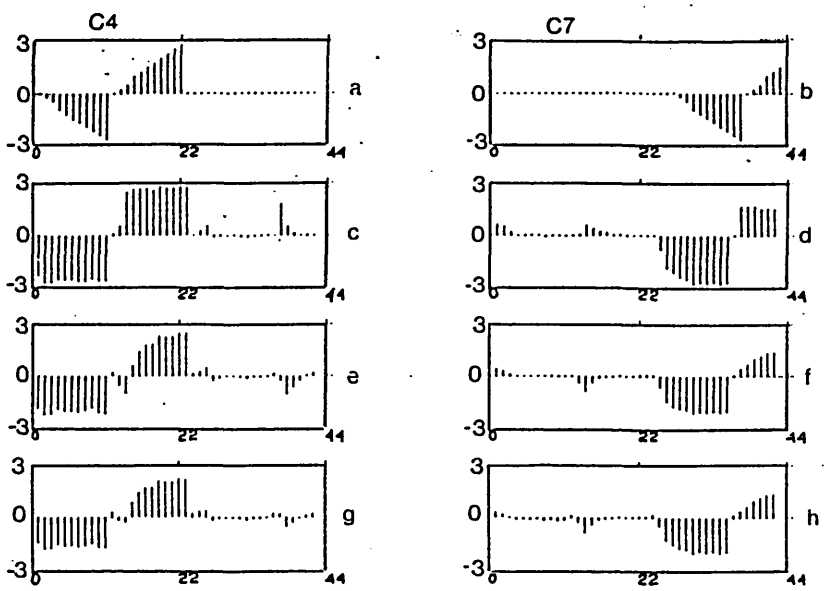


Figure 6 Adaptive combiner results

Unfortunately, the three layer network did not produce very good results. At this stage our aim was not to identify the most appropriate architecture but to determine how easy or feasible that job is. For that reason when over one hundred nets were run before further investigation was suspended. An interesting, and somewhat expected, fact arose with the use of the nets. Increasing the hidden nodes drastically improved (Table 15) the performance for each individual node. It is probable that additional re-learning of the learning set will produce better results in the future.

NUMBER OF EXAMPLES:39	NUMBER OF CORRECT PREDICTIONS ON NODE 1	2	3	4	ARCHITECTURE
	20	0	0	0	6-1-3-4
	25	3	6	0	6-2-3-4
	21	22	5	0	6-4-5-4
	29	28	16	8	6-20-10-4
	23	16	9	3	6-10-20-4

Table 15. Neural net predictive accuracy
(Search III)

DISCUSSION

Although the three techniques are different, a comparison was possible. ID3 performed slightly better than the other two for the first two searches. For the third search ID3 failed

significantly. This was due to the inability of the algorithm to handle a large number of classes. The use of ID3 for the first two searches was elected. The decision was based on the following advantages of ID3, as seen by the authors.

- An expert system cannot ever be completed. Such systems should append their knowledge through time. The incremental presentation of examples demonstrated that running ID3 was faster. Neural networks took a long time to train. Some architectures took up to 17 hours to train.

- ID3 always gives correct predictions for the examples used in the learning set. This is not quaranteed with the other two techniques.

- ID3 generated decision trees which can be transformed in the form of rules. These rules can be used directly to explain the relationships between the attributes and the decisions made. With weights a direct explanation is not feasible.

- ID3 obtained slightly better results with less manipulation of parameters and without the need to worry about the order of introduction of the examples. With adaptive combiners a lot of time was spent in experimenting with parameters. The problem with neural nets was the absence of any theory in determining the architecture.

The adaptive combiner performed well for the third search. The advantage of the combiner over ID3 is as follows. The only available option of improving ID3's performance is by introducing further examples. This results to the re-generation of a decision tree which may result in a different set of rules. One has further options using the combiners. Employing the same learning set one can experiment with the forgetting factor and/or re-introduce the same learning set. This was done for the first search and resulted in a better performance. New examples will not effect the structure of the combiners and only the weights will be updated.

In future, neural nets will be further investigated as a solution to the third search space problem and a connective expert system will be tested live on the filter and on other similar filter designs.

CONCLUSIONS

ID3 is recommended for the first two searches having the better performance and requiring little time to up-date. Also, rules may be derived from the decision trees. Adaptive combiners are prefered for search 3, for which ID3 is inapplicable and for which neural networks require excessive training time.

REFERENCES

1. Bramer, M.A. A survey and critical review of expert systems research. Introductory readings in expert systems, (Ed. Michie, D.), Gordon and Breach Science publishers, 1982.

2. Computer, Vol.19, No. 7, 1986.

3. Newell, A. and Simon, H.A. Human problem solving, Prentice-Hall, 1972.

4. Ericsson, K.A. and Simon, H.A. Verbal reports as data, Psychological review, Vol. 87, No.3, pp. 215-251, 1980

5. Quinlan, R. Discovering rules from a large collection of examples : A case study. Expert systems in the micro-electronic age, (Ed. Michie, D.), Edinburgh University press, 1979.

6. Mirzai, A.R.(Ed.). Artificial Intelligence : Concepts and applications in engineering, Kogan Page, to be published.

7. Pollack, J.B. Connectionism: past, present and future, Artificial Intelligence Review, Vol.3, No.1, pp.3-22, 1989.

8. Le Cun, Y. Learning process in an asymmetric thershold network. Disordered systems and biological organization, (Ed. Bienenstock, E., Fogelman, S.F. and Weisbuch, G.), Springer Verlag, 1986.

9. Parker, D.B. Learning Logic, Techical report-47, MIT Center for computational research in economics and management science, MA, USA, 1985.

10. Rumelhart, W.E., Hinton, G.E. and Williams, R.J (Ed.). Parallel distributed processing : Explorations in the microstructure of cognition, Vol.1, MIT Press, 1986.

11. Lipmann, R.P. An introduction to computing with neural nets, IEEE ASSP Magazine, April, pp.4-22, 1987.