16

KNOWLEDGE ACQUISITION FOR EFFECTIVE AND EFFICIENT USE OF ENGINEERING SOFTWARE

by

D L HAWLA

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosphy in the Faculty of Engineering, University of Cape Town.

National Institute for Aeronautics and Systems Technology
Council for Scientific and Industrial Research

August 1987

ABSTRACT

The problem of effective and efficient use of engineering software can be thought of as a Pareto optimal problem. However, the complexity of modern engineering software precludes the possibility of acquiring complete knowledge of the software's Pareto optimal set. Instead heuristic knowledge must be acquired. The thesis proposes that heuristic knowledge be acquired via a knowledge acquisition procedure. The use of a knowledge acquisition system, which may be computerised, forms an integral part of this procedure. Two examples of knowledge acquisition illustrate the use of the knowledge acquisition procedure.

DECLARATION

I, David Lesley Hawla, hereby declare that this thesis is essentially my own work and that no part of it has been submitted for a degree at any other university.

D L HAWLA
August 1987

ACKNOWLEDGEMENTS

I wish to express my gratitude to the following people.

My supervisors, Dr Hanoch Neishlos and Prof J B Martin for their guidance, patience and encouragement throughout this work.

The National Institute for Aeronautics and Systems Technology for their financial support.

My colleagues at UCT and NIAST for their encouragement and advice.

Ms Joanne Kirsten for typing the thesis.

My friends for interrupting me and keeping me sane.

Gino! I guess I lost the bet?

CONTENTS

PREFACE

The original title of this thesis was to have been 'Numerical Modelling of Reinforced Concrete Frames under Transient Dynamic Conditions'. The aim of the research would be to develop suitable constitute models for concrete. When the research was begun, Prof Martin suggested that I write a finite element program which could be used as the primary tool for such research (At that stage, the Dept. of Civil Engineering did not have a suitable program). So I wrote NLFRAM. NLFRAM could analyse plane frames made of reinforced concrete (or any other material) under transient dynamic loading. Geometric and material nonlinearities could be taken into account.

One of the first applications of NLFRAM was to the modelling of a reinforced concrete beam, loaded statically. Even in this relatively simple case, I had considerable difficulty selecting step sizes and finding suitable equilibrium iteration algorithms, but eventually, by an iterative process of trial, error and observation, I succeeded. Many computer runs were required. During the process, it became quite clear that, depending on the selection of the numerical modelling parameters (control variables) wide variations in predicted response could be produced, with even wider variations in computational effort. At some point in the research, Prof Martin suggested that, rather than concentrating on concrete constitutive modelling, I concentrate on improving the efficiency of the algorithms used in the modelling of reinforced concrete structures. He also suggested that Dr Neishlos might be a useful guide for such research. Well, Dr Neishlos became so enthusiastic that, within six months, I was not looking to improve the efficiency of solving reinforced concrete problems, but was looking to improve the general use of engineering packages. The question was how could one systematize the acquisition of cost-reliability-control information for an engineering package. Dr Neishlos pointed out that, for any given package, the variety of problems and ways to solve them numerically, i.e. the number of controls, was enormous, so I should concentrate on a class of problems and controls. He also pointed out that the conventional way to acquire information on this class would be to use statistically designed experiments, but that this approach

might be very expensive. A pattern recognition approach, he said, might be more efficient. To test these ideas I wrote a prototype version of ELIXIR, chose the reinforced concrete beam mentioned above as a problem class and proceeded to use ELIXIR to acquire cost-reliability-control information about NLFRAM. Even with only three variables in the control class and only one problem in the problem class, the cost-reliability-control behaviour was quite difficult to understand from the initial statistically designed experimental programme. But, by using interactive graphics for pattern recognition, and sorting, filtering (according to the Pareto principle) and clustering, I learnt enough about the package to design an additional experimental programme which, on average, would produce more efficient solutions than the initial programme. Pattern recognition, filtering, etc. were then repeated. In fact, the whole cycle of experimental programme design, pattern recognition, etc. was repeated four times before I was satisfied with the cost-reliability-control knowledge I had acquired. We (Dr Neishlos and I) clearly recognised the heuristic nature of the knowledge acquisition process that I had followed and the heuristic nature of the knowledge acquired. From Lakatos (Philosophy of Science) and Pearl (Artificial Intelligence) we also realised that it was correct to use heuristic procedures to acquire heuristic knowledge.

The next two questions to be answered were: how could the knowledge acquisition process be made generally applicable, and how should the heuristic knowledge be disseminated? The first led to my adoption of Klir's general systems concepts, and the second, to the adoption of a rule form (Expert Systems, Artificial Intelligence) for knowledge representation. In order to computerise the general systems version of the knowledge acquisition procedure, I realised that the new version of ELIXIR would require flexible data and program structures, so I turned to the database and artificial intelligence literature for help. I also realised that the new ELIXIR would be quite complex and, in order to manage its development, I drew on ideas in software engineering. Once completed, the new ELIXIR was tested on the reinforced concrete beam problem again, and then also on a problem involving linear elastic curved beams. Finally, I put all these ideas together and wrote this thesis.

Dave Hawla
August 1987

NOTATION

All the below symbols and abbreviations are defined in the text but are listed here for reference purposes.

DEP	Design evaluation problem
KAP	Knowledge acquisition problem
HK	Heuristic Knowledge
KAS	Knowledge Acquisition System
KAS_i	Components of the KAS
CBKAS	Computer-Based KAS. The components of the CBKAS have names such as FELIX, PRELIX, ELIXIR, etc. and do not correspond one-to-one with the components of the KAS
KAc	Knowledge Acquirer, i.e. the person performing the knowledge acquisition
FEA	Finite Element Analysis
AI	Artificial Intelligence
EK	Evaluative knowledge
MM	Mathematical model
NM	Numerical model
DEProc	Design evaluation procedure
DEProg	Design evaluation programme
KAProc	Knowledge acquisition procedure
KAProg	Knowledge acquisition programme

The following symbols are based on the general systems concepts of [55]. Unsubscripted capital letters denote sets. A lower case letter with a subscript is used for names of attributes or variables.

KAP(P,G)	A KAP for processor P with goal G
P	Processor, i.e. an engineering software package
G	The goal of knowledge acquisition. It will always be: "Find HK characterising an approximate Pareto optimal set for P"

$KAP(P,G^i)$	A knowledge acquisition subproblem for P with G^i being the goal: "Find HK^i characterising an approximate Pareto optimal set for problem class PC^i"
$KAP(P,G^{ij})$	A knowledge acquisition subproblem for P with G^{ij} being the goal: "Find HK^{ij} characterising an approximate Pareto optimal set for problem class PC^i and control class CC^{ij}"
PC^i	Problem class i of processor P
HK^i	Heuristic knowledge about PC^i
CC^{ij}	Control class j for PC^i
HK^{ij}	Heuristic knowledge about PC^i and CC^{ij}

Notational Patterns in Systems Symbols

Before providing details of the contents of the various systems, we will first give an overview the main symbols used for primitive, data and transformed data systems. The pattern in the representation should then become evident.

The primitive systems are:

$\sum^0 = (S^0, S^0, \beta)$, a source system

where

$S^0 = (X^0, F^0, Y^0, I^0)$ is an object system

and $S^0 = (X^0, F^0, Y^0, I^0)$ is an image system

Now $X^0 = (X^N, X^R)$, $F^0 = (F^N, F^R)$, etc. so we can also

define $S^N = (X^N, F^N, Y^N, I^N)$ and

$S^R = (X^R, F^R, Y^R, I^R)$.

Thus we can also write S^0 as $S^0 = (S^N, S^R)$

Similarly $X^0 = (X^N, X^R)$, $F^0 = (F^N, F^R)$, etc.

When data is added to S^0, we get

$$S^D = (X^D, F^D, Y^D, I^D), \text{ a } \underline{\text{data system}},$$

where $X^D = (X^N, X^R, X)$, $F^D = (F^N, F^D, F)$, etc.

We can also define $S^N = (X^N, F^N, Y^N, I^N)$, $S^R = (X^R, F^R, Y^R, I^R)$ and $S = (X, F, Y, I)$ and then write

$$S^D = (S^N, S^R, S).$$

Note: Although it is difficult to distinguish visually between S and S or Y and Y, the context of their use should make it clear enough.

Transformed data systems simply have '~' or '^' above each symbol

i.e. $\tilde{S}^D = (\tilde{S}^N, \tilde{S}^R, \tilde{S}) = (\tilde{X}^D, \tilde{F}^D, \tilde{Y}^D, \tilde{I}^D)$ and

$$\hat{S}^D = (\hat{S}^N, \hat{S}^R, \hat{S}) = (\hat{X}^D, \hat{F}^D, \hat{Y}^D)$$

Note, however, that $\hat{S}^D$ does not have $\hat{I}^D$ as a component.

Primitive Systems

$\sum^0$ A source system $\sum^0 = (S^0, S^0, \beta)$, i.e. it consists of an object system, an image system and a homomorphism between them

S^0 An object system: $S^0 = (X^0, F^0, Y^0, I^0)$, i.e. attribute names and appearance sets for problem inputs, processor controls, outputs and indices

X^0 Input attributes and appearance sets, i.e. $X^0 = (X^N, X^R)$

X^N Set of input attribute names, i.e. $x^N = \{X_1^N, x_2^N, \ldots\}$

X^R Set of possible appearances or ranges of appearances of input attributes, i.e. $X^R = \{x_1^R, x_2^R, \ldots\}$

x_j^N Name of j'th input attribute

x_j^R Range of appearances of j'th input attribute

F^O Processor control attributes, i.e. $F^O = (F^N, F^R)$

F^N Set of names of processor control attributes, i.e. $F^N = \{f_1^N, f_2^N, \ldots\}$

F^R Set of ranges of appearances of conrol attributes. $F^R = \{f_1^R, f_2^R, \ldots\}$

f_j^N Name of j'th control attribute

f_j^R Appearance range of j'th control attribute

Y^O Output attributes, i.e. $Y^O = (Y^N, Y^R)$

Y^N Set of names of output attributes, i.e. $Y^N = \{y_1^N, y_2^N, \ldots\}$

Y^R Set of ranges of appearances of output attributes, i.e. $Y^R = \{y_1^R, y_2^R, \ldots\}$

y_j^N Name of j'th output attribute

y_j^R Appearance range of j'th output attribute

I^O Index attributes, i.e. $I^O = (I^N, I^R)$

I^N Set of names of index attributes, i.e. $I^N = \{i_1^N, i_2^N, \ldots\}$

I^R Set of ranges of appearance of index attributes, i.e. $I^R = \{i_1^R, i_2^R, \ldots\}$

i_j^N Name of j'th index attribute

i_j^R Apperance range of j'th index attribute

S^0 An image system: $S^0 = (X^0, F^0, Y^0, I^0)$, i.e. variable names and value sets for problem inputs, processor controls, outputs and indices

X^0 Input variable names and ranges of values, i.e. $X^0 = (X^N, X^R)$

X^N Set of names of problem input variables. $X^N = \{x_1^N, x_2^N, \ldots\}$

X^R Set of possible values or ranges of values of input variables, i.e. $X^R = (x_1^R, x_2^R, \ldots\}$

x_j^N Name of j'th input variable

x_j^R Value range of j'th input variable

F^0 Processor control variable names and value ranges, i.e. $F^0 = (F^N, F^R)$

F^N Set of names of processor control variables. $F^N = \{f_1^N, f_2^N, \ldots\}$

F^R Set of value ranges of processor control variables. $F^R = \{f_1^R, f_2^R, \ldots\}$

f_j^N Name of j'th processor control variable

f_j^R Value range of j'th processor control variable

Y^0 Output variable names and value ranges, i.e. $Y^0 = (Y^N, Y^R)$

Y^N Set of output variable names, i.e. $Y^N = \{y_1^N, y_2^N, \ldots\}$

Y^R Set of value ranges of output variables. $Y^R = \{y_1^R, y_2^R, \ldots\}$

y_j^N Name of j'th output variable

y_j^R Value range of j'th output variable

I^O Index variable names and value ranges, i.e. $I^O = (I^N, I^R)$

I^N Set of index variable names, i.e. $I^N = \{i_1^N, i_2^N, \ldots\}$

I^R Set of value ranges of index variables, i.e. $I^R = \{i_1^R, i_2^R, \ldots\}$

i_j^N Name of j'th index variable

i_j^R Value range of j'th index variable

β Homomorphic mappings between attribute names and variable names and between attribute appearances and variable values, i.e. $\beta = (\beta^N, \beta^R)$

β^N Homomorphism between (X^N, F^N, Y^N, I^N) and (X^N, F^N, Y^N, I^N)
$\beta^N = \{\beta_1^N, \beta_2^N, \ldots]$

β^R Homomorphism between (X^R, F^R, Y^R, I^R) and (X^R, F^R, Y^R, I^R).
$\beta^R = \{\beta_1^R, \beta_2^R, \ldots\}$

β_j^N j'th homomorphism between attribute and variable names. Often between x_j^N and x_j^N.

β_j^R j'th homomorphism between appearance and value sets of the attributes and variables related by β_j^N

Data Systems

Data systems are primitive systems to which arrays of data have been added [55]. Adding data collected from running experimental programmes to S^O we get S^D. Individual cases of data, i.e. data for a single experiment are denoted with Greek subscripts.

S^D	(X^D, F^D, Y^D, I^D)
X^D	(X^N, X^R, X)
F^D	(F^N, F^R, F)
Y^D	(Y^N, Y^R, Y)
I^D	(I^N, I^R, I)
X	Set of problem input data, i.e. $\{X_1, X_2, \ldots\}$
X_α	α'th set of input values of experimental data. $X_\alpha = (x_{1\alpha}, x_{2\alpha}, \ldots\}$, $X_\alpha \in X^R$
$x_{j\alpha}$	α'th value of x_j^N; $x_{j\alpha} \in x_j^R$
F	Set of processor control data, i.e. $\{F_{11}, F_{12}, F_{13}, \ldots, F_{21} \ldots\}$
$F_{\alpha\beta}$	β'th value of processor control values associated with X_α. $F_{\alpha\beta} = \{f_{1\alpha\beta}, f_{2\alpha\beta}, \ldots\}$, $F_{\alpha\beta} \in F^R$
$f_{j\alpha\beta}$	$\alpha\beta$'th value of f_j^N; $f_{j\alpha\beta} \in f_j^R$
Y	Set of output data, i.e. $\{Y_{11}, Y_{12}, Y_{13}, \ldots, Y_{21}, \ldots\}$
$Y_{\alpha\beta}$	Output produced by P from $X_\alpha, F_{\alpha\beta}$ $Y_{\alpha\beta} = \{y_{1\alpha\beta}, y_{2\alpha\beta}, \ldots\}$; $Y_{\alpha\beta} \in Y^R$
$y_{j\alpha\beta}$	$\alpha\beta$'th value of y_j^N ; $y_{j\alpha\beta} \in y_j^R$
I	Set of index values, i.e. $\{I_{11}, I_{12}, I_{13}, \ldots, I_{21}, \ldots\}$
$I_{\alpha\beta}$	Index values associated with $X_\alpha, F_{\alpha\beta}, Y_{\alpha\beta}$. $I_{\alpha\beta} = \{i_{1\alpha\beta}, i_{2\alpha\beta}, \ldots\}$
$i_{j\alpha\beta}$	$\alpha\beta$'th value of i_j^N ; $i_{j\alpha\beta} \in i_j^R$.

Transformed data systems

The transformed data system $\tilde{S}^D$ is simply denoted by putting a '~' above each element/item occurring in S^D. In a very similar manner $\hat{S}^D$ is denoted but it does not contain any index variables, i.e. $\hat{S}^D = (\hat{X}^D, \hat{F}^D, \hat{Y}^D)$ with $\hat{X}^D$, $\hat{F}^D$, $\hat{Y}^D$ having the same constituents as X^D, F^D, Y^D but with a '^' above the relevant symbols.

Subsets

Whereas Greek subscripts were used to denote individual cases of data, Roman subscripts are used to denote groups of cases or subsets of the data sets or range sets. Mostly this will only be used with $\hat{S}^D$.

$$\hat{S} = (\hat{X}, \hat{F}, \hat{Y})$$

$$\hat{S}_{k\ell} = (\hat{X}_k, \hat{F}_{k\ell}, \hat{Y}_{k\ell}), \quad \hat{X}_k \ \hat{X}, \quad \hat{F}_{k\ell} \ \hat{F}, \quad \hat{Y}_{k\ell} \ \hat{Y}$$

$$\hat{X}_k = \{\hat{X}_{k\alpha} \mid \alpha=1,2,\ldots,n_k\}$$

$$\hat{F}_{k\ell} = \{\hat{F}_{k\ell\alpha\beta} \mid \alpha=1,2,\ldots,n_k,\ \beta=1,2,\ldots,n_{k\ell}\}$$

$$\hat{Y}_{k\ell} = \{\hat{Y}_{k\ell\alpha\beta} \mid \alpha=1,2,\ldots,n_k,\ \beta=1,2,\ldots,n_{k\ell}\}$$

A K-partition of $\hat{S}$ will be denoted by $\hat{S}_K$

where $\hat{S}_K = \{\hat{S}_{k\ell} \mid k=1,2,\ldots K,\ \ell=1,2,\ldots,L(k)\}$

Using similar conventions, we denote subsets of $\hat{S}^R = (\hat{X}^R, \hat{F}^R, \hat{Y}^R)$ by

$$\hat{S}^R_{k\ell} = (\hat{X}^R_k, \hat{F}^R_{k\ell}, \hat{Y}^R_{k\ell})$$

where $\hat{X}^R_k \ \hat{X}^R$, $\hat{F}^R_{k\ell} \ \hat{F}^R$, $\hat{Y}^R_{k\ell} \ \hat{Y}^R$.

Whereas the $\hat{S}_{k\ell}$ are finite point sets, the sets $\hat{S}^R_{k\ell}$ generally consists of contiguous intervals or cells in $\hat{S}^R$.

1 INTRODUCTION

A structural engineer faced with the problem of certifying the strength of a mechanical component will plan a set of stress analyses, execute the plan, i.e. calculate the stresses, and then integrate this information/data into an evaluative statement of the strength of the component. The plan will contain some simple analyses done by hand or via handbook tables and some detailed analyses such as finite element analysis (FEA). An evaluation of the strength of a mechanical component is however just a special case of what we will call a design evaluation problem (DEP).

Complementary to design evaluation problems are the problems of providing/inventing/developing engineering analysis theories formulae, handbooks, codes of practice, computer packages, guides for using such packages and training in the use of these tools. In this thesis, we wish to address only the problem of providing guidance for the use of engineering computer packages for aiding the solution of design evaluation problems.

Numerical modelling packages exist for simulating many physical phenomena exhibited by solids, fluids and complex machines. Until recently however, complex numerical models such as those used in FEA were only built for the evaluation of expensive or otherwise important components of large engineering projects. However, the decreasing cost of computer hardware and increasing availability of sophisticated engineering software has meant a much greater use of such software tools ([78] is entitled 'FEA for the Masses'). On the other hand, the scope and complexity of such software has in general increased and continues to increase [69,70]. The combination of these factors has led to concern among such software users, vendors and academics regarding the problem of effective and efficient use of such software and the credibility of the results of such use. In the field of FEA there has recently been a considerable increase in the number of publications and conferences addressing this problem [39,40,69, 70, 76,77,26]. In fluid dynamics, [80] says that: "Computational fluid dynamics has many successes for the solution of simple standard problems. For relatively complex problems especially if nonlinear and of mixed type, the computed approximate solutions are mostly of

dubious accuracy and credibility." The contention of this thesis is that part of the problem may be alleviated by the provision of adequate appropriate knowledge of the Pareto optimal set of engineering packages. By the Pareto optimal [54] set of an engineering package, we mean the set of values of the numerical modelling variables (e.g. step sizes) which, for any values of the design variables (e.g. dimensions), simultaneously maximises the accuracy and minimises the cost of solution. Currently such knowledge is derived from some basic theory of numerical analysis, user's manuals with their demonstration examples, experience and common (engineering) sense. It is inevitably vague and incomplete and thus of a heuristic nature. Such heuristic knowledge is often held by a relatively small number of experts who have built up their expertise through years of experience with the software. Acquiring such heuristic knowledge is therefore a significant problem and in this thesis will be called the knowledge acquisition problem (KAP). That knowledge acquisition is a problem is recognised by expert systems people [5,6,39,40] as the 'bottleneck' in expert system development. For convenience, heuristic knowledge will hereafter be abbreviated to HK.

The principal aim of the thesis is to show how heuristic knowledge for numerical modelling via engineering software can be acquired. The aim of this introductory chapter, is to provide the reader with most of the concepts needed for the thesis. It thus forms a condensed version of the thesis as a whole. Later chapters will deal more thoroughly with the methods and ideas presented in this chapter.

The next section presents a simplified systems view of the use of engineering software. It introduces the notation and some of the basic concepts to be used in the thesis and explains why heuristic knowledge must be sought. Section 1.2 deals with knowledge acquisition. It shows how heuristic knowledge is usually acquired and for this it is necessary to explore the design evaluation process. It then shows briefly how heuristic knowledge can (and should) be acquired. Section 1.3 then presents a knowledge acquisition procedure incorporating a knowledge acquisition system (KAS). This procedure and the KAS consist of an integration of traditional and artificial intelligence (AI) techniques unified via the concepts of general systems [55]. Section 1.4 briefly explains the necessity of computerising the KAS. Section

1.5 presents an exercise of acquiring heuristic knowledge for numerical modelling of a reinforced concrete beam via an FEA package called NLFRAM.

The approach to knowledge acquisition presented in the thesis supports the increased effort towards adequate training and education of users of engineering software. In fact it should ease the process of education and training because the knowledge acquisition procedure is a systematic procedure by which software users may organise their experience and so acquire HK themselves.

1.1 A SIMPLIFIED SYSTEMS VIEW OF THE USE OF ENGINEERING SOFTWARE

As an introduction to the general systems concepts and notation to be used in the thesis, a simplified systems view of the use of engineering software will be given. Most of the general systems concepts used in this section are based on Klir [55].

Any engineering software package may be considered as a processor P that takes problem inputs plus some engineer (user) supplied processor controls and generates outputs. The process of using P may be represented as in Figure 1.1.1.

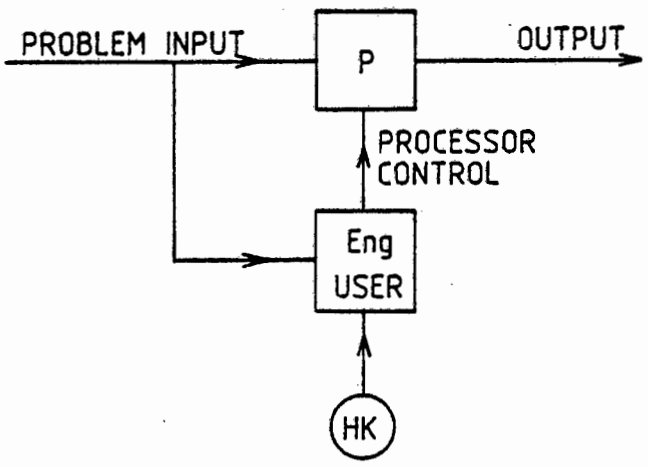


FIGURE 1.1.1 : Systems view of the process of using an engineering software package P

The presence of HK in the above serves to emphasize the fact that the engineer makes use of heuristic knowledge (experience, advice in manuals, etc.) to select appropriate processor controls.

For FEA of structures, typical problem inputs are geometric data, material properties, loads, boundary conditions and initial conditions. Typical processor controls are mesh density, element type, time step sizes, equilibrium iteration algorithms, iteration convergence tolerances and numerical integration order and typical outputs are displacements, velocities, reactions and stresses.

Usually the HK consists of vague relations between inputs, controls and outputs derived from user's manuals, especially the example problems sections, previous experience, some basic theory and common (engineering) sense. It consists of knowledge of trends such as accuracy increases with increasing mesh density, decreasing time step sizes and tighter convergence tolerances; mesh density should be higher in areas of large gradients; high accuracy usually involves high cost; etc.

Figure 1.1.1 perhaps gives the impression that for any problem, the selection of processor controls and generation of outputs is a straightforward process. In practice, however, complex mathematical modelling problems are solved by an iterative refinement process involving a succession of increasingly complex or more detailed numerical models. The value of the HK lies in its use for effective and efficient control of this process.

The problem space of P may be defined as the set of all possible problems which P can solve. Similarly, the control space of P consists of all possible ways of using P to solve problems in its problem space. P maps its problem and control spaces onto its output space. Ideally the user would like to know how to control the solution process for any problem in the problem space. However, even for quite small engineering packages, the enormous variety of problems and controls make the problem and control spaces so complex that it is usually impossible (or at least unrealistic) to acquire such complete knowledge. Instead, engineering users usually settle for knowledge

which is derived from the study of P on simplified problem and control spaces. These simplified problems and control spaces will be called problem classes (PC's) and control classes (CC's). Simplification can be achieved by Pearl's approach of constraint modification [2]. The knowledge derived from the study of such simplified spaces will be heuristic [2]. An extreme application of simplification would produce a set of examples, i.e points in the problem and control spaces.

A problem class (PC) can be defined in terms of variables and ranges of values of these variables, i.e. as $X^0 = (X^N, X^R)$, where $X^N = \{x_1^N, x_2^N, \ldots\}$ is the set of variable names for the PC and $X^R = \{x_1^R, x_2^R, \ldots\}$ is the set of ranges of values which the variables may take. Similarly, a control class (CC) can be defined by $F^0 = (F^N, F^R)$, where $F^N = \{f_1^N, f_2^N, \ldots\}$ is the set of variable names for the CC and $F^R = \{f_1^R, f_2^R, \ldots\}$ are the corresponding value ranges. Corresponding to X^0 and F^0 is the output class $Y^0 = (Y^N, Y^R)$. Taken together, the problem, control and output classes form a primitive system $S^0 = (X^0, F^0, Y^0)$.

Any given problem in the PC can be represented by its data $X_\alpha \in X^R$. Because the selection of processor controls is dependent on X_α and in addition on cost and reliability aspects, processor control data will be represented by $F_{\alpha\beta} \in F^R$. When X_α and $F_{\alpha\beta}$ are fed into P, the output is $Y_{\alpha\beta} \in Y^R$. This view is shown in Figure 1.1.2.

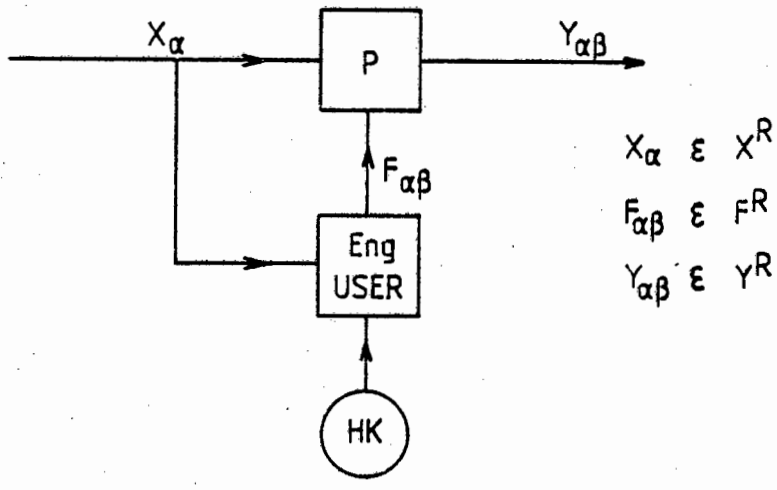


FIGURE 1.1.2 : Systems view of the process of using of an engineering package P on a particular problem class and a particular control class.

The HK in Figure 1.1.2 is now specific to the problem and control class specified by S^O. Taken together, all data for a particular use of P, i.e. $(X_\alpha, F_{\alpha\beta}, Y_{\alpha\beta})$ will be referred to as a data case $(S_{\alpha\beta})$. A set of data cases will be represented by $S = (X,F,Y)$. When S^O is supplemented by data S, the result is a data system $S^D = (X^D, F^D, Y^D)$. $X^D = (X^N, X^R, X)$, $F^D = (F^N, F^R, F)$ and $Y^D = (Y^N, Y^R, Y)$. Alternatively, $S^D = (S^N, S^R, S)$ where $S^N = (X^N, F^N, Y^N)$, $S^R = (X^R, F^R, Y^R)$ and $S = (X,F,Y)$.

It was stated in the introduction to this chapter that the goal of knowledge acquisition is to acquire knowledge concerning the behaviour of P such that, armed with this knowledge, P may be used effectively and efficiently. Effective use requires that the engineer has sufficient knowledge to select processor controls $F_{\alpha\beta}$ which will solve his problem X_α to an appropriate level of reliability (accuracy, confidence, etc.). Efficient use requires in addition that the controls selected produce the desired reliability in the cheapest manner possible.

The use of the term reliability of solution rather than the term accuracy of solution needs explanation. The term accuracy of solution is usually understood to be some measure of closeness to an exact solution. However, most engineering software packages were developed

for applications for which exact solutions are not available. The reliability of a solution will therefore be defined as some measure of closeness to what we will call a reference solution. A reference solution is one which is believed to be sufficiently accurate for all engineering purposes. The term 'believed' rather than 'proved' is used because, while proof will usually be impossible, quite strong arguments from physical and mathematical principles can be made to justify belief in the validity of the reference solution. The use of such terminology follows that of [56-60]. Invariably reference solutions will be expensive to obtain. In some ways reference solutions are like measurement standards or control experiments - they represent standards against which other things are measured. However, standards are usually chosen for their accuracy with repetition (this is no problem for most engineering software) whereas reference solutions are chosen as a sort of ideal solution or for their credibility as an approximation to an ideal solution.

The efficiency of obtaining a solution depends on the selection of $F_{\alpha\beta}$ because a similar level of reliability can be achieved with widely varying costs using different $F_{\alpha\beta}$ [80]. The appropriate level of reliability will depend on the reliability requirements and cost constraints of the particular design evaluation for which the problem X_α is to be solved. While effective use requires only knowledge of the range of possible levels of reliability and their generating $F_{\alpha\beta}$, efficient use requires in addition that these $F_{\alpha\beta}$ produce solutions with minimum cost. In the language of polyoptimisation this may be expressed as the following optimisation problem: find the $F_{\alpha\beta} \in F^R$ which are Pareto optimal [54] for each $X_\alpha \in X^R$, where the objectives are to maximise reliability and minimise cost. However, acquiring this knowledge of the Pareto optimal set of S^0 may be too expensive. The expense may be reduced by converting the optimisation problem to an approximate optimisation problem. By extending the definition of approximate optimisation given in [2] to approximate Pareto optimisation, the above stated optimisation problem becomes: find the $F_{\alpha\beta} \in F^R$ which are approximately Pareto optimal for each $X_\alpha \in X^R$. An approximately optimal solution is defined in [2] as one which has a sufficiently high probability of being within a specified factor of the optimal solution.

1.2 KNOWLEDGE ACQUISITION

In section 1.1, and Figure 1.1.2 in particular, we gave a systems view of the process of using an engineering package P to solve a problem defined by X_α. The purpose of using P to solve X_α was stated in the introduction to be that of aiding the solution of design evaluation problems. In using P the engineer (user) made use of HK. The purpose of this section is to show how HK is usually acquired - for this we need to study design evaluation more closely - and to provide an introduction to a systematic way of acquiring HK.

Given a DEP (D,G), i.e. a DEP for design D and a goal G capturing the design requirements, an engineer will use P, any HK and a prior evaluative knowledge EK^0 to find evaluative knowledge EK for the design. For example, D might be a turbine blade with G being to evaluate D with respect to strength. EK^0 would typically be derived from a simple hand stress calculation indicating that the stresses are close to the material rupture stress and so more detailed stress analysis is required. P would typically be a structural FEA package and the HK would then relate to the use of this package. The EK sought is a statement of whether the blade will fail or not. In general, G defines appropriate evaluation criteria (failure, performance, serviceability, etc.) which are in turn defined in terms of the behaviour (stresses, flows, efficiencies, etc.) of the design. EK is all knowledge/information/data needed to evaluate the design with respect to G.

The above view may be formulated in the following block diagram form.

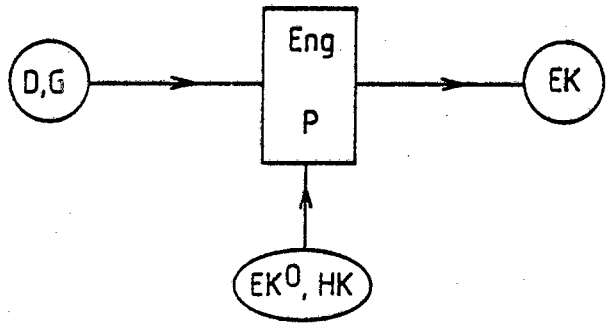


FIGURE 1.2.1 : An abstract block diagram view of the design evaluation process

In Figures 1.1.1 and 1.1.2, the engineer (user of P) was given the problem X_α and selected only the controls $F_{\alpha\beta}$. Now, however, he is given a DEP(D,G) and not only selects controls $F_{\alpha\beta}$ but also : decides how to model D via P by selecting X_α and $F_{\alpha\beta}$, interprets the output $Y_{\alpha\beta}$ and finally makes an evaluative statement. All of these are guided by EK^0 and HK. This more detailed view is depicted as follows.

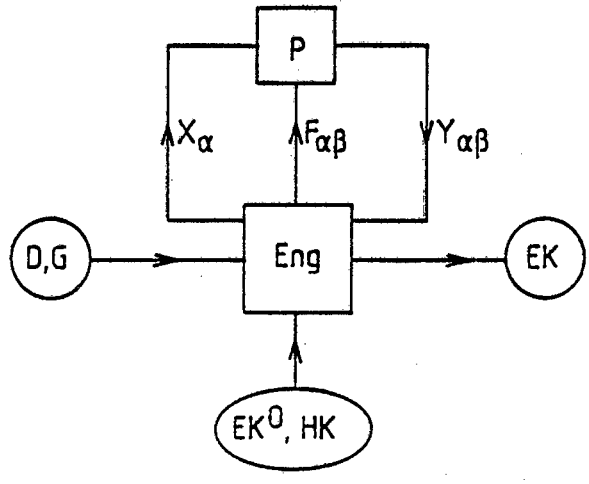


FIGURE 1.2.2 : The use of P to simulate the behaviour of D in the design evaluation process

The X_α essentially specifies a mathematical model of D while $F_{\alpha\beta}$ specifies how to solve X_α numerically. Usually a number of mathematical models each solved via a number of numerical models will be required for each design evaluated. This set of mathematical and

their associated sets of numerical models together form what will be called a design evaluation programme (DEProg). Ideally a DEProg will be planned in the first phase of evaluation. A DEProg allows the engineer to learn systematically about the physical behaviour of a design D and so enables him to make evaluative statements about D. During the planning of a DEProg the engineer will use many heuristics like making conservative simplifying assumptions, relaxing some constraints, etc. (cf simplification of problem and control spaces). Although the process is based on sound mathematical, physical and logical principles, the heuristic human decision element automatically makes the (planning) process and its output (the DEProg) heuristic too.

A by-product of using P to learn about designs is that the engineer simultaneously learns about the behaviour of P. With sufficient intelligent and varied use of P, the engineer will build for himself an experiential heuristic knowledge base. Such knowledge will usually be very difficult to state explicitly and will therefore be difficult to convey to others. Also, due to time constraints, etc. the knowledge may not be acquired systematically and so will usually be fragmented and far from Pareto optimal. The learning process is essentially passive. What is needed instead is systematic active experimentation to acquire explicit HK.

While the primary goal of design evaluation is the production of EK for a design D, in the KAP the primary goal G is the production of HK which ideally characterises the approximate Pareto optimal set of a processor P. Such a KAP will be denoted by KAP(P,G). If a knowledge acquirer (KAc) makes use of a KAS (a computer-based system for aiding knowledge acquisition) and any a priori heuristic knowledge (HK^0) available to him to solve a KAP(P,G), the knowledge acquisition procedure can be represented in the following block diagram form.

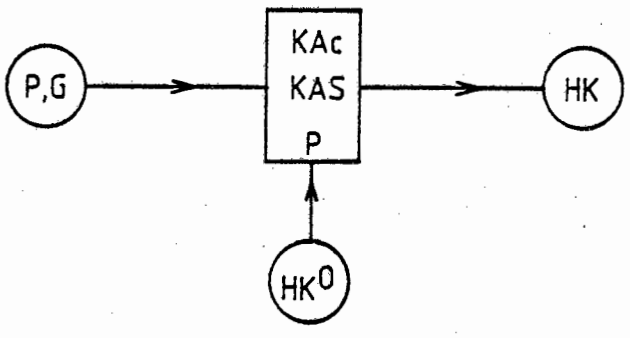


FIGURE 1.2.3 : An abstract block diagram view of the knowledge acquisition process (Cf. Figure 1.2.1).

In contrast with the passive acquisition of HK about P, the knowledge acquisition procedure uses active experimentation on P to learn about its behaviour and thus formulate explicit HK. The following figure illustrates this view.

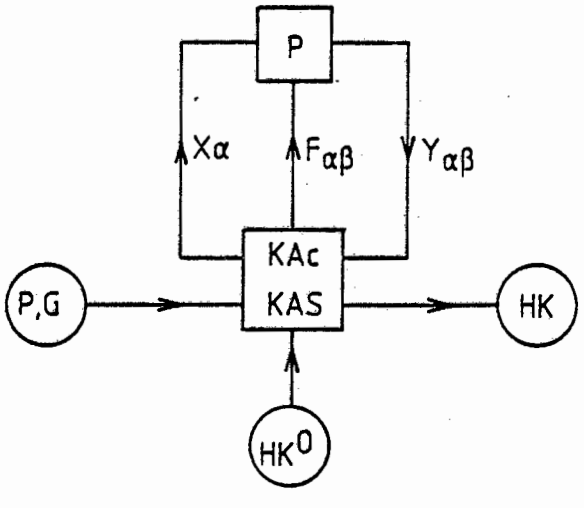


FIGURE 1.2.4 : Learning the behaviour of P by active experimentation in the knowledge acquisition procedure (Cf. Figure 1.2.2)

An important difference between the design evaluation and knowledge acquisition processes shown in Figures 1.2.2 and 1.2.4 respectively should be highlighted. In the former P is used to simulate the behaviour of D while in the latter P itself is the object of study and so appears twice in Figure 1.2.4.

The knowledge acquisition procedure to be presented in the next section will be based on the following. Given a KAP(P,G) for P with goal G, a KAc, aided by a KAS, uses any available HK^0 to design experimental programmes on the problem and control classes, collect the data, apply appropriate transformations, recognise patterns in the data, formulate HK by inference from these patterns and finally tries to explain the HK obtained. The whole process may be repeated and, as more is learnt about P, the HK becomes increasingly refined.

Although inferred from experimental data, the knowledge will generally be heuristic rather than empirical because

(a) the programmes of experiments will be designed using heuristic principles, not only statistical methods,

(b) the knowledge is inferred by pattern recognition, human and machine, and

(c) the process is one of learning (by the KAc primarily).

Heuristic knowledge is always goal-specific because all of (a) to (c) are done with the goal borne in mind. It will usually be incomplete because problem and control classes are simplifications of the whole spaces and cater only for the most important mathematical and numerical modelling problem types. The knowledge is clearly not deductive because experimentation is basic to the procedure.

1.3 A KNOWLEDGE ACQUISITION PROCEDURE

The knowledge acquisition procedure we propose consists of three main phases. First is a planning phase, where the KAP is defined, goals set and strategies or plans for solving the problem and attaining the goals are developed. It involves decomposition of the problem. Second is the execution phase, i.e. carrying out the tasks or subproblems set in the planning phase. It involves data or information gathering. Planning and execution are usually guided by current knowledge and experience. Finally comes an integration phase where the results of

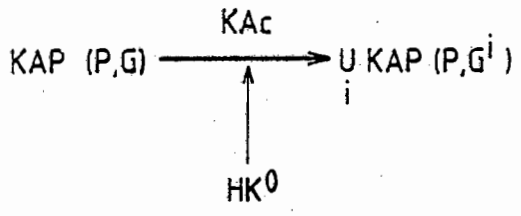
executing the individual tasks or subproblems are integrated into a statement of the result of the knowledge acquisition exercise as a whole. It also involves relating knowledge and experience acquired to previous knowledge and experience.

The Planning Phase

In the planning phase, the original KAP(P,G) is replaced by a set of knowledge acquisition subproblems, $KAP(P,G^i)$ and $KAP(P,G^{ij})$. Each goal G^i is: "Find HK^i characterising an approximate Pareto optimal set for problem class PC^i", while each goal G^{ij} is: "Find HK^{ij} characterising an approximate Pareto optimal set for problem class PC^i and control class CC^{ij}". The set of knowledge acquisition subproblems, i.e. $U_{i,j}$ $KAP(P,G^i)$, $KAP(P,G^{ij})$ will be referred to as a knowledge acquisition programme (KAProg). Note that G^i is defined in terms of PC^i and G^{ij} in terms of PC^i and CC^{ij}. The planning phase is done (almost) exclusively by the KAc. He will draw on any available a priori heuristic knowledge to help him formulate the KAProg. P is supposed to be able to analyse complex mathematical models via the analysis of numerical models and the objective of knowledge acquisition is to provide the user of P with HK to aid the design of such numerical models. Complex mathematical models are usually analysed by some approximate decomposition into a set of simpler submodels and then these simpler submodels analysed via numerical models. The set of problem classes in the KAProg should (ideally) be such that numerical modelling advice for each mathematical submodel exists in one or more of the problem classes. Of course, this is only an ideal and ultimately a decision must be made on the appropriate level of complexity of individual problem classes. Typically the KAProg should have a number of simple problem and control classes plus some more complex ones but few trivial classes (classes with only one variable). Even though the knowledge acquisition effort is focussed on a set of problem classes rather than the whole problem space, for each problem class the set of all possible ways of controlling P (i.e. the control subspace for the problem class) may still be large and complex. So just as a problem space is replaced by a set of problem classes, control subspaces of P are replaced by sets of control

classes. Each problem class PC^i will have its own set of control classes namely CC^{ij}. The planning phase is summarised symbolically by the following figure.

(i)



(ii) For each i:

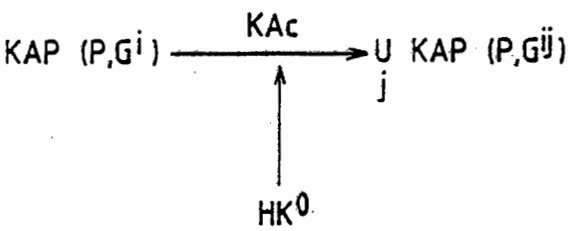


FIGURE 1.3.1 : The planning phase of knowledge acquisition

The Execution Phase

The execution phase consists of acquiring heuristic knowledge HK^{ij} for each of the subproblems $KAP(P,G^{ij})$. The KAc will utilise any a priori heuristic knowledge HK^0 and knowledge acquisition system (KAS) available to him. This process is shown in Figure 1.3.2.

For each i and for each j:

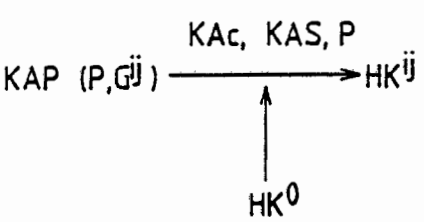


FIGURE 1.3.2 : Execution of each of the knowledge acquisition subproblems

The execution phase for each knowledge acquisition subproblem divides naturally into five basic steps, namely

(0) system definition, i.e. defining variables to be measured

(1) experimentation and data collection

(2) data transformation, e.g. calculation of reliabilities

(3) 'partitioning' of the transformed data using pattern recognition [9]

(4) formulation of HK^{ij}.

The above five steps are easily identified in Figure 1.3.3 which shows a symbolic view of the whole execution phase. Each step involves the use of an element KAS_i of the KAS. Each of these steps will be explained in terms of the KAS_i and their products S^0, S^D, $\hat{S}^D$ and $\hat{S}^D_K$. It will also be assumed that the KAc operates the KAS_i and makes use of HK^0 throughout the procedure.

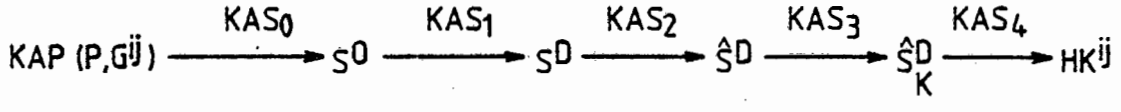


FIGURE 1.3.3 : A more detailed symbolic view of the execution phase showing the five basic steps involved

KAS_0 and S^0

When the KAProg is designed, its constituent PC^i and CC^{ij} will normally be defined in fairly descriptive terms. For example they might be defined via a set of diagrams and lists of problem and control attributes. These diagrams and attributes must be converted to a set of variables and value ranges, i.e. $S^0 = (X^0, F^0, Y^0)$ where $X^0 = (X^N, X^R)$, etc. must be defined. The objective of defining S^0 is to provide a proper framework for experimentation.

KAS_1 and S^D

KAS_1 represents the design of experimental programmes [13-17] on the sets X^R and F^R, the execution of the experiments and the collection of the data. Each experiment or run of P results in a data case

$(X_\alpha, F_{\alpha\beta}, Y_{\alpha\beta})$. Together these data cases form $S = (X,F,Y)$. When S^O is supplemented by data S the result is the data system S^D.

KAS_2 and $\hat{S}^D$

The KAc is usually not interested in the outputs Y themselves but rather in particular features $\hat{Y}$ of the output such as reliability and cost. In general $\hat{Y}$ will be some transformation of Y. The transformed variables and their ranges are represented by $\hat{Y}^N$ and $\hat{Y}^R$. Similarly the KAc may require transformed problem and control variables. The result is a transformed data system $\hat{S}^D = (\hat{X}^D, \hat{F}^D, \hat{Y}^D)$ or $\hat{S}^D = (\hat{S}^N, \hat{S}^R, \hat{S})$. It will, however, be quite common to have $\hat{X}^D = X^D$ and $\hat{F}^D = F^D$.

KAS_3 and $\hat{S}^D_K$

By $\hat{S}^D_K$ we mean $(\hat{S}^N, \hat{S}^R_K, \hat{S}_K)$. $\hat{S}^R_K$ is a set of subsets of $\hat{S}^R$ and $\hat{S}_K$ is a set of subsets of $\hat{S}$. $\hat{S}^R_K = \{\hat{S}^R_{k\ell} \mid k = 1,2,\ldots, K, \ell = 1,2,\ldots, L(k)\}$, where $\hat{S}^R_{k\ell} = (\hat{X}^R_{k\ell}, \hat{F}^R_{k\ell}, \hat{Y}^R_{k\ell}) \subset \hat{S}^R$. $\hat{S}_K$ is similarly defined. K is the number of subsets $\hat{X}^R_k$ of $\hat{X}^R$ while L(k) is the number of subsets $\hat{F}^R_{k\ell}$ of $\hat{F}^R$ associated with $\hat{X}^R_k$. The KAc's goal here is to form $\hat{S}^R_K$. This he will do by first studying the data $\hat{S}$, selecting $\hat{S}_K$ and then by induction forming $\hat{S}^R_K$. At this point the reader may be justifiably confused because we have not explained why the KAc's goal is to form $\hat{S}^R_K$ nor constrained him in his selection of the sets of subsets ($\hat{S}^R_K$ and $\hat{S}_K$). In the next subsection we will see how the above set represention matches the rule form we have chosen for representing the HK. Each set $\hat{S}^R_{k\ell}$ will correspond to a single rule. Some of the constraints on selecting $\hat{S}^R_K$ are: the union of the $\hat{X}^R_k$ should equal $\hat{X}^R$ so that the whole problem class is covered, the union of the $\hat{Y}^R_{k\ell}$ should approximately cover $\hat{Y}^R_{k\ell}$ for each ℓ so that an adequate selection of (for example) costs and reliabilities is provided, the $\hat{F}_{k\ell}$ should yield approximately Pareto optimal solutions ($\hat{F}^R$ will not normally be covered), the $\hat{S}^R_{k\ell}$ (the individual rules) and $\hat{S}^R_K$ (the set of rules) should have simple forms and should be explainable. Sets of subsets which satisfy such constraints will be called 'partitions', i.e. $\hat{S}^R_K$ is a 'partition' of $\hat{S}^R$. More detail can be found in [Chapter 3].

But how does the KAc 'partition' $\hat{S}^R$ into subsets satisfying these constraints? Usually the first step would be to apply a filter to the data $\hat{S}$, where the criteria for filtering are based on the Pareto optimality of the $\hat{Y}_{\alpha\beta}$. In other words, individual elements $\hat{S}_{\alpha\beta}$ whose $\hat{Y}_{\alpha\beta}$ are relatively inefficient are discarded. This should yield a typical trade-off pattern (for example of cost increasing with reliability) in the remaining data. Then the KAc would look for groups of cases in $\hat{F}$ and $\hat{X}$ which produce similar $\hat{Y}$. Clustering techniques [12,20] can be quite useful as an aid in this regard. At some point the KAc decides on a 'partition' $\hat{S}_K$ of the data $\hat{S}$. By induction he then 'partitions' $\hat{S}^R$ into $\hat{S}^R_K$, i.e. he infers a set of $\hat{S}^R_{K\ell}$ whose properties can be represented by the data $\hat{S}_{K\ell}$. The process is shown symbolically in Figure 1.3.4. Much more detail of the techniques and processes involved is given in [Chapter 3].

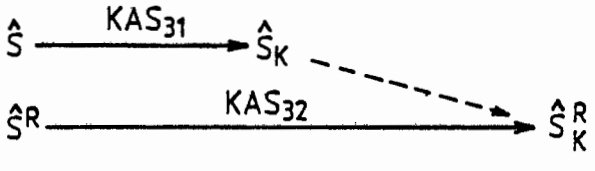


FIGURE 1.3.4 : Symbolic view of the operation of KAS_3. KAS_{31} involves pattern recognition and 'partitioning' of data while KAS_{32} involves 'partitioning' of $\hat{S}^R$ into $\hat{S}^R_K$ by induction (indicated by the broken line) from $\hat{S}_K$.

KAS_4 and HK^{ij}

KAS_4 involves formulating HK^{ij} from the 'partitioned' data set $\hat{S}^D_K$. Knowledge representation forms are important here. This thesis proposes that the HK be represented as rules. A rule will be taken to mean a statement of the following form:

IF antecedent THEN consequent

Examples of rules can be seen in Section 1.5. Such rules are to be interpreted as providing sufficient conditions (antecedents) to

provide desired consequents. In HK^{ij} the desired consequents would be a particular $\hat{Y}_{\alpha\beta}$, for example a desired level of reliability and an acceptable cost. The antecedents would then specify to which $\hat{X}_{\alpha}$ (mathematical model) such $\hat{Y}_{\alpha\beta}$ is applicable and which $\hat{F}_{\alpha\beta}$ is required to achieve it. Quite clearly the subsets ($\hat{X}^R_k$, $\hat{F}^R_{k\ell}$, $\hat{Y}^R_{k\ell}$) convey such information. In fact one may write the $k\ell$'th rule as

$$\text{IF} \quad \hat{X}_{\alpha} \in \hat{X}^R_k \quad \text{and} \quad \hat{F}_{\alpha\beta} \in \hat{F}^R_{k\ell} \quad \text{THEN} \quad \hat{Y}_{\alpha\beta} \in \hat{Y}^R_{k\ell}.$$

Feedback

Although not shown in any of the previous figures, feedback is an essential part of the knowledge acquisition procedure just as it an integral part of any adaptive, iterative or learning process. Feedback of knowledge already learned becomes a priori knowledge for the next step in the process. Typically it is used for: redesign of experimental programmes; selection of features, reliability criteria, data subsets, data views for pattern recognition and pattern recognition aids and even to reformulate the primitive system. In this way knowledge may be refined. Many feedback paths need to be added to Figure 1.3.3. More detail may be found in [Chapter 3].

The Integration Phase

The final phase of knowledge acquisition is to combine the separately acquired HK^{ij} into a structured integrated whole, i.e. into HK. Symbolically this may be represented as in Figure 1.3.5.

(i) For each i:

$$\bigcup_j HK^{ij} \xrightarrow{KAc} HK^i$$

(ii)

$$\bigcup_i HK \xrightarrow{KAc} HK$$

FIGURE 1.3.5 : Integrating the separately acquired HK^{ij} and HK^i into HK for the original problem

Strongly connected to such integration is the explanation of the HK^{ij} and the HK. Explanation is important because

(i) It helps the user of the HK to understand it and hence accept and use it.

(ii) It serves as a check on the validity/acceptability of the HK. If the KAc cannot find an explanation this should be regarded as an indication of lack of understanding and hence a motivation for further study. Learning to understand the behaviour of P is important for knowledge refinement. Usually explanations will be of an intuitive nature.

(iii) It can form the basis of generalisations of the HK^{ij} to broader problem and control classes [50].

Also important in integration is description of how to use the HK and what its limitations are.

Dissemination of the HK

The simplest form of dissemination is to provide the users of P with the HK in a document accompanying the standard documentation of P. If the HK or parts thereof are sufficiently robust and their antecedents easily recognised in the data input for P, it may be built directly into P. Some automatic time stepping schemes in FEA packages are examples of this form of dissemination. Increasingly popular is dissemination in the form of an expert system. In fact recent developments in expert systems technology were partly inspirational for the present work. The rule form we have chosen is suitable for dissemination by any of these methods.

1.4 THE NEED FOR A COMPUTER-BASED KNOWLEDGE ACQUISITION SYSTEM

The HK acquired will normally be shared by a number of users who may each have a different set of criteria for measuring reliability. For example, suppose P computes outputs y_1 and y_2. If P is an FEA package, y_1 might be displacements and y_2 might be stresses . One user may define reliability in terms of y_1 while another in terms of y_2. Although their reliabilities $\hat{y}_1$ and $\hat{y}_2$ may be positively correlated, the magnitudes may be very different and inappropriate criteria will lead to either inefficient use of P or to inadequate reliability. The reliabilities may however be negatively correlated in which case further trade-offs are necessary. Even if it is intended that the HK deal with whichever criteria yields the worst reliability, the KAc may not know beforehand which one it will be. Faced with such a situation the KAc would ideally like to acquire knowledge based on more than one criteria and hence need to record both y_1 and y_2 values. For numerical solution of PDE's, this could result in vast quantities of data being recorded and processed. Computer implementation of the KAS thus becomes essential.

Although it was realised that a computer-based knowledge acquisition system (CBKAS) applicable to quite general engineering packages would be needed, it was also realised that its development would require considerable effort. Therefore, we decided to test the feasibility of

computerisation by developing a prototype version of ELIXIR which was specific to a particular KAP (In fact it was specific to the KAP dealt with in the next section). This also helped to identify the data management and processing requirements for the more general version of ELIXIR. The feasibility study highlighted the need for very flexible data and program structures. This led to the adoption of ideas from database theory, artificial intelligence program and data structures, interactive programming, graphics and open systems. To manage the implementation of the second version of ELIXIR, some software engineering principles were adopted. Although very useful, as demonstrated in the examples, ELIXIR will need additional features not foreseen and perhaps even restructuring. Due to the variety of possible applications, it is believed that ELIXIR will continue to evolve.

Briefly the ELIXIR system consists of the following set of programs:

FELIX - Corresponds to KAS_0. It is used to define variable names, types, etc. It initialises a database called EDB (ELIXIR Database). All data specific to the KAP is stored in EDB or any database created by the ELIXIR system. Also included is a specification of how PRELIX is to interpret the data produced by the processor.

PRELIX - Corresponds to KAS_1. It is used to capture data (X, F, Y), i.e. the values of variables.

ELIXIR - This is the main module. KAS_2 and KAS_3 are done with ELIXIR. Transformations of S^D to $\hat{S}^D$ are done via ELIXIR. $\hat{S}^D$ is stored on another database called EDBn in a form more suited to the pattern recognition processes of KAS_3.

Other modules exist for reporting and graphical viewing of the database contents.

1.5 A SIMPLIFIED EXAMPLE TO ILLUSTRATE THE MAIN CONCEPTS

The example presented below is a simplified version of an example in [Chapter 5]. It demonstrates the basic ideas and steps in the knowledge acquisition procedure.

The Planning Phase - KAP(P,G) and KAProg

NLFRAM [79] is a finite element structural analysis program. Plane frames with material and/or geometric nonlinearities may be analysed under static or transient dynamic loading.

It was known how to use NLFRAM to get reliable solutions. However, efficient use of NLFRAM requires knowledge of the possible levels of reliabilities and their accompanying costs and how to achieve these. Such knowledge was not available.

Even though NLFRAM is a relatively small program, it is still not realistic to acquire complete knowledge of its behaviour. The heuristic approach replaces this complete knowledge with HK for a set of problem and control classes (i.e. a KAProg). Here the use of the KAS can only be demonstrated on a single problem and control class. By suitable generalisation, the classes could easily be extended to cover most of the applications of NLFRAM. The KAProg therefore consists of only one problem and one control class, namely $KAP(P,G^{11})$.

The Execution Phase - $KAP(P,G^{ij})$ and HK^{ij}

KAS_0 and S^0

The goal G^{11} for the chosen classes is as follows:

G^{11} = Find the heuristic knowledge characterising the approximate Pareto optimal set for the problem shown in Figure 1.5.1. (The problem class contains only one problem.)

The problem, control and output considered relevant are characterised by the following attributes.

Inputs : (beam dimensions; materials; supports; loading programme)

Processor controls : (mesh density; number of depth integration points; concrete tensile stress release rate)

Outputs : (displacement under point load; computational cost)

The measurements of these attributes are made via the variables in the primitive system:

S^O = $\{(X^N, X^R), (F^N, F^R), (Y^N, Y^R)\}$

Here

X^N = {BMLEN, BMDEP, BMWID; MATNO; BCONDS; PLOAD}

X^R = {all variables fixed to values as shown in Figure 1.5.1}

F^N = {NELTS; NDEPTH; ALPHA}

F^R = {(4, 6, 8, 10, 12, 14, 16); (4, 5, 7, 9, 11, 13); (real numbers [4.0, 12.0])}

Y^N = {DISP; COST}

Y^R = {(real numbers [0.0, 0.012]); (real numbers [0.0, 1000.0])}

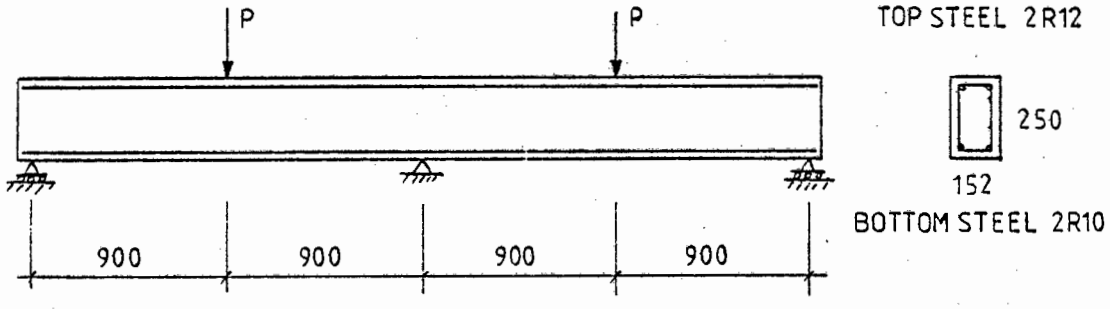


FIGURE 1.5.1(a) : Reinforced Concrete Beam

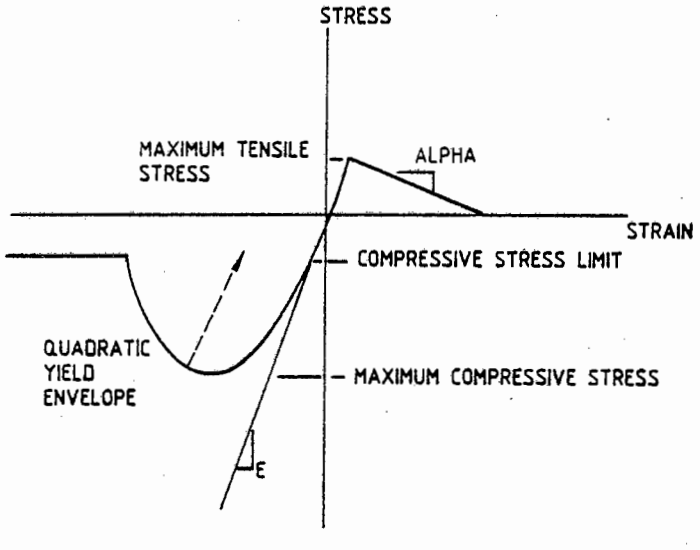


FIGURE 1.5.1(b) : Concrete Behaviour

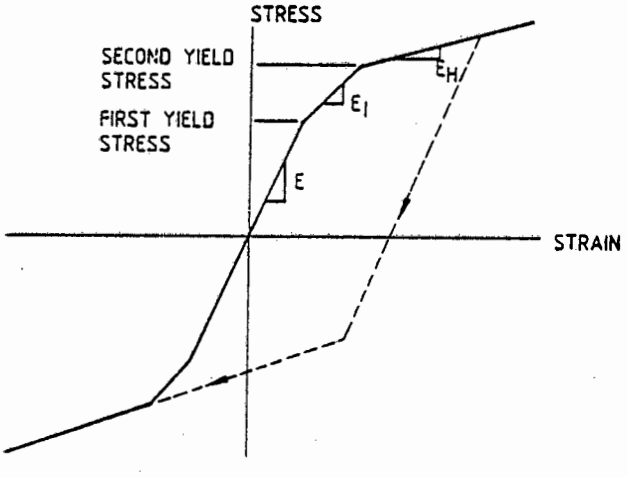


FIGURE 1.5.1(c) : Steel Behaviour

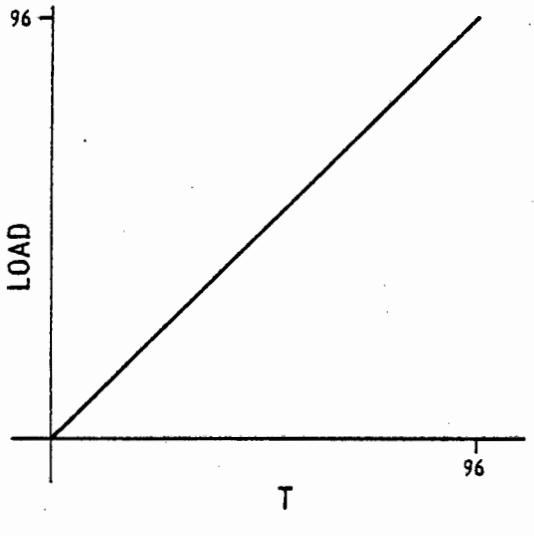


FIGURE 1.5.1(d) : Load-Time History

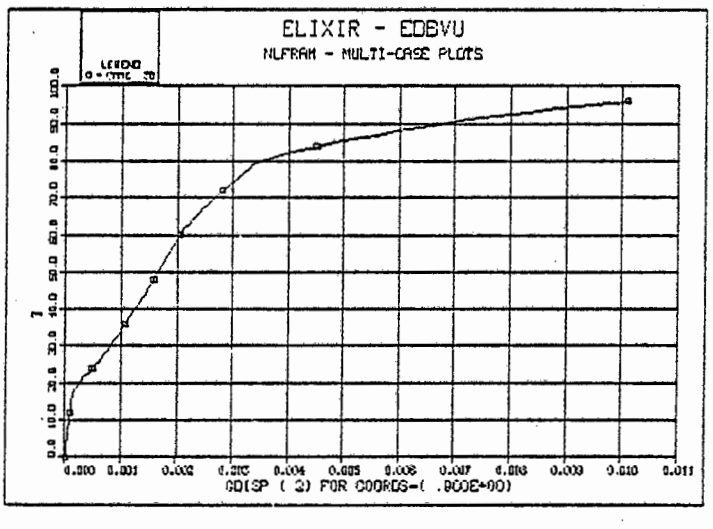


FIGURE 1.5.1(e) : Load-Deflection Response

KAS_1 and S^D

The next step was to generate S = (X,F,Y) and form S^D. X is just the data needed by NLFRAM to describe the problem. F is a set of values which was established in a series of programmes of experiments on F^R (a total of 65 experimental cases were generated). From X and F, NLFRAM generated Y. Amongst these cases was one which produced a highly reliable solution (a reference solution).

KAS_2 and $\hat{S}^D$

We were not interested in Y itself but only in its reliability and cost aspects. To calculate the reliabilities we selected DISP as the relevant output feature and compared it to that of the reference solution. Thus $\hat{Y}$ consisted of the reliability and cost of each solution. In this example problem and control variables were not transformed, i.e. $\hat{X}^D = X^D$ and $\hat{F}^D = F^D$.

KAS_3 and $\hat{S}^D_K$

Figure 1.5.2(a) shows a plot of the $\hat{Y}$ data, i.e. reliabilities and costs, for the initial experimental programme. Since we were interested in the computational efficiency of obtaining solutions, we removed by a filtering process, based on the Pareto principle, those cases that were inefficient. The result of this is shown in Figure 1.5.2(b).

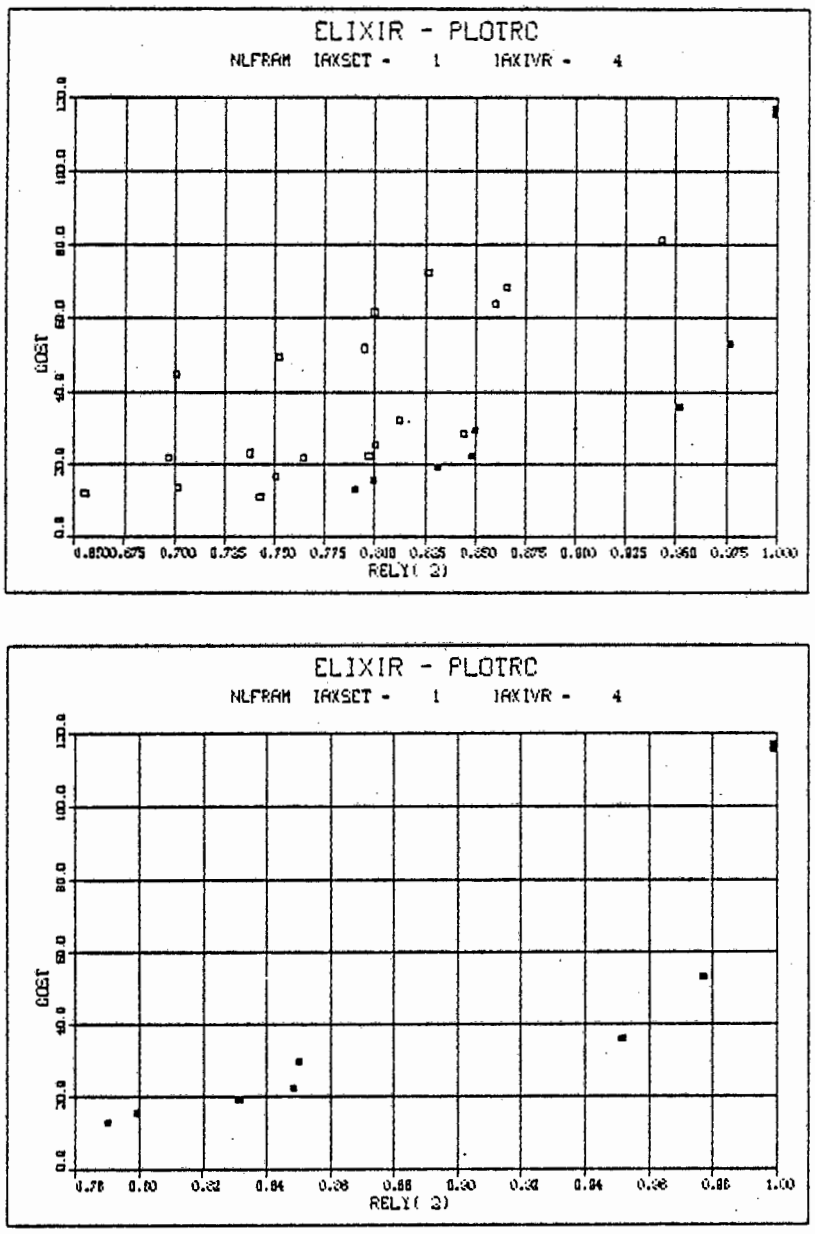


FIGURE 1.5.2 : Reliability versus Cost for the Initial Programme of Experiments

(a) Unfiltered

(b) Relatively inefficient solutions removed

No clear pattern emerges from the above : the data is just too sparse. However, from the tabulated filtered data it was possible to identify regions in $\hat{F}^R$ which would yield points in the RELY-COST plane closer to the Pareto optima. A series of further experimental programmes was run - a total of four programmes. $\hat{S}_K$ could then be identified. Then by induction from $\hat{S}_K$, $\hat{S}_K^R$ was selected and a final programme of

experiments chosen to provide a better 'empirical basis' for these subsets ($\hat{S}^R_{k\ell}$). Some more inefficient cases were filtered and some cases which would complicate the rules were removed too. The results of this are shown in Table 1.5.1 and Figure 1.5.3.

CASE NO.	NELTS	NDEPTH	ALPHA	RELY (1)	COST SRU
2	.160E+02	.130E+02	.400E+01	.999	117.
27	.160E+02	.130E+02	.400E+01	.999	116.
60	.800E+01	.900E+01	.450E+01	.983	41.4
61	.800E+01	.110E+02	.450E+01	.980	45.9
52	.800E+01	.110E+02	.400E+01	.980	47.6
50	.800E+01	.900E+01	.400E+01	.979	45.3
51	.800E+01	.900E+01	.500E+01	.975	43.9
53	.800E+01	.110E+02	.500E+01	.974	49.5
57	.600E+01	.110E+02	.500E+01	.954	37.1
56	.600E+01	.900E+01	.500E+01	.947	36.2
63	.600E+01	.110E+02	.550E+01	.944	32.9
62	.600E+01	.900E+01	.550E+01	.944	29.0
45	.600E+01	.900E+01	.700E+01	.857	21.2
46	.600E+01	.110E+02	.700E+01	.855	21.5
59	.600E+01	.110E+02	.750E+01	.850	23.1
58	.600E+01	.900E+01	.750E+01	.850	23.3
38	.600E+01	.110E+02	.800E+01	.845	24.2
36	.600E+01	.900E+01	.800E+01	.844	21.9
31	.400E+01	.900E+01	.600E+01	.815	13.9
43	.400E+01	.700E+01	.600E+01	.804	15.9
32	.400E+01	.900E+01	.800E+01	.799	14.7
7	.400E+01	.700E+01	.800E+01	.790	13.2

TABLE 1.5.1 : Filtered and clustered $\hat{S}$ data after the final programme of experiments

Table 1.5.1 contains $\hat{S}_K$. The induction to $\hat{S}^R_K$ was simple here and will be clear from the rules given in the next subsection. Rather than write out the $\hat{S}^R_{k\ell}$, we simply wrote down the rules based on them.

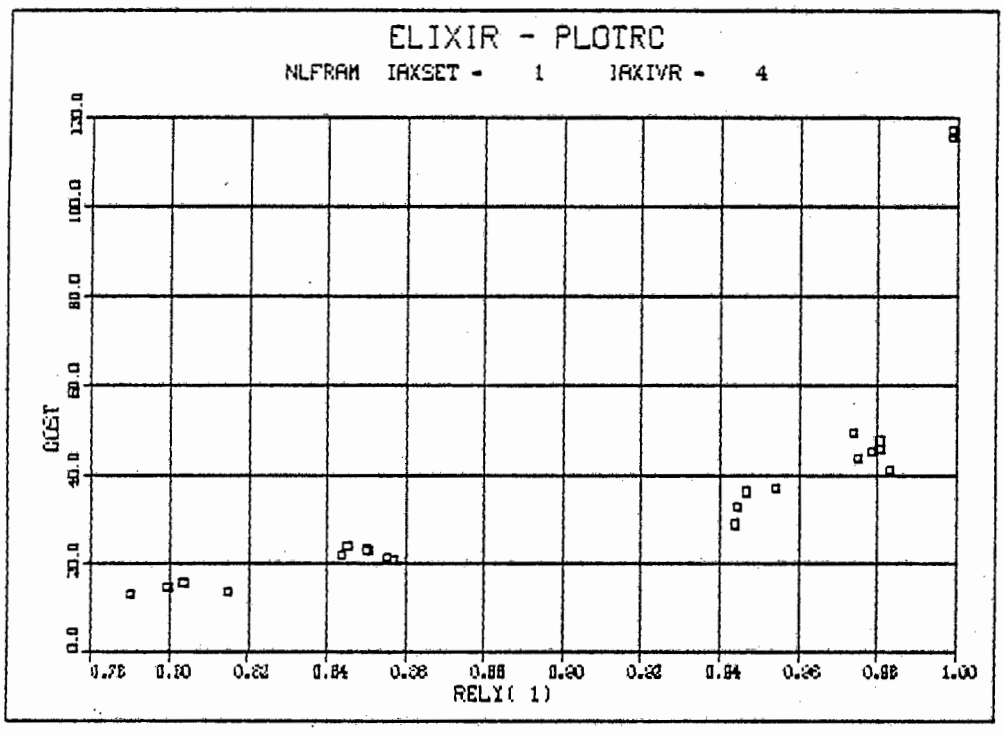


FIGURE 1.5.3 : Plot of $\hat{Y}$ (COST-RELY) for data of Table 1.5.1

KAS_4 and HK^{ij}

The final knowledge acquisition step was to formulate the information in Table 1.5.1 into heuristic knowledge. Each cluster will contribute a rule. Thus the heuristic knowledge is as follows:

Heuristic Knowledge:

Rule 1 : If NELTS = 16 and NDEPTH = 13 and ALPHA = 4
then RELY ≈ 0.999 and COST ≈ 117 SRU.

Rule 2 : If NELTS = 8 and NDEPTH ε (9,11) and ALPHA ε [4.0, 5.0]
then RELY ≈ 0.97 and COST ≈ 47 SRU.

Rule 3 : If NELTS = 6 and NDEPTH ε (9,11) and ALPHA ε [5.0, 5.5]
then RELY ≈ 0.94 and COST ≈ 36 SRU.

Rule 4 : If NELTS = 6 and NDEPTH ε (9,11) and ALPHA ε [7.0, 8.0]
then RELY ≈ 0.84 and COST ≈ 23 SRU.

Rule 5 : If NELTS = 4 and NDEPTH ϵ (7,9) and ALPHA ϵ [6.0, 8.0]
then RELY $\approx$ 0.80 and COST $\approx$ 15 SRU.

In addition all other variables must be set at the reference values or some suitably specified values found from a previous knowledge acquisition exercise.

The Integration Phase

In a more extensive knowledge acquisition exercise, the first step in the integration phase would be to combine HK^{ij} for the various CC^{ij} for PC^{i} into HK^{i}. Next it would be necessary to combine the HK^{i} into HK. In this example these tasks cannot be demonstrated. However one of the keys to successfully performing these tasks is the explanation of the HK in terms of basic principles and its generalisation to broader problem and control classes. This we will illustrate briefly.

The pattern of rules can be related to the physics of the problem as follows. At high loads the steel has yielded and hence, for higher reliability, more elements are needed to describe the region of steel plasticity. The concrete behaviour is highly nonlinear, hence the high number of depth integration points. Also, at higher loads, the steel dominates the beam behaviour so that values of ALPHA higher than the reference value, could be used without much loss of reliability. This physical support of the HK is important because it increases confidence in the validity of the knowledge. Also, it makes generalisation easier. For example, the HK should be applicable to beams with any span to depth ratio where self-weight is insignificant.

Also important are the limitations of the HK, e.g. although reliability estimates may be excellent for general frames, for frames with more than two elements/members connected at a node, cost estimates will usually be too low. It should also be noted that the rules are usually used by selecting a desired reliability and acceptable cost and then finding which values of the processor

variables may be used to obtain such cost and reliability. The word 'may' is used because HK provides only guidance on use. It should be sufficient but not necessary.

Feedback

The presentation only hints at the use of feedback but this is merely a simplification for illustration purposes only. In fact the 65 experiments were performed as four experimental programmes, successive programmes using knowledge of the $\hat{Y}$ and $\hat{F}$ patterns (which represents the approximate Pareto set) to focus the search onto the areas expected to contain the Pareto set.

1.6 SUMMARY AND CONCLUSION OF CHAPTER 1

In this chapter, we saw that, for effective and efficient use of engineering software, the user requires knowledge of its Pareto optimal set. This knowledge essentially concerns numerical modelling. But why do we consider numerical modelling knowledge so important? Next we explained that complete knowledge of the Pareto optimal set could not be obtained and instead we stated briefly that heuristic knowledge should be sought. This statement needs elaboration. In order to acquire heuristic knowledge, we proposed a systematic heuristic procedure for knowledge acquisition, but did not explain adequately why we should use a heuristic procedure. What are the alternatives? Dissemination of heuristic knowledge was mentioned quite briefly but requires further discussion. All these questions and elaborations are dealt with in the next Chapter.

The main result of this thesis is the development of a knowledge acquisition procedure and chapter 3 has therefore been devoted to its elaboration.

Quite clearly the knowledge acquisition system used in the procedure requires computerisation. Chapter 4 expands upon the computer-based knowledge acquisition system, called ELIXIR, presented in Section 1.4. In addition, it discusses software development issues.

Whereas in section 1.3 the aim was mainly to demonstrate the knowledge acquisition procedure, the presentation in Chapter 5 also aims to demonstrate the use of ELIXIR. This it does via two examples, the first being an expanded account of the example in section 1.5 and the second being an example of acquiring knowledge for modelling curved structural members efficiently. Unfortunately, the two examples cannot demonstrate a full scale knowledge acquisition study, but we believe that they do provide adequate indication of how such a study might be started.

2 KNOWLEDGE ACQUISITION

This chapter of the thesis deals with some of its more fundamental aspects. Specifically, it tries to answer the questions of why numerical modelling knowledge is important, of why heuristic knowledge should be sought and of why heuristic procedures are appropriate. Finally it deals with some aspects of dissemination of heuristic knowledge and explains why the rule form of knowledge representation is appropriate for dissemination purposes.

2.1 WHY IS NUMERICAL MODELLING KNOWLEDGE IMPORTANT?

The engineering design process can roughly be divided into four steps, namely design specification, generation of designs, evaluation of designs and the selection of a design. Evaluation of a design requires prediction of its behaviour or response and comparison with its specified requirements. This view of the design process is applicable to both conceptual and detail design. However, the emphasis will be different. In conceptual design most effort will be spent on design specification and design generation with fairly rough evaluation and comparisons being made of the fairly rough designs generated. On the other hand, in detail design, most effort will be spent on evaluating designs reliably and on modifying the components of the detail designs generated. The different emphasis of the effort is dictated essentially by the reliability of the required evaluation. Reliable evaluation may require complex mathematical and numerical modelling and physical experimentation in order to predict reliably the behaviour of the design.

In both conceptual and detail design, a significant proportion of the effort is spent on numerical modelling. The increasing pressure to make designs lighter, stronger, more efficient and higher-tech [38] in order to compete in world markets will require increasingly reliable evaluation and so an increasing proportion of the design effort will be spent on numerical modelling. The different emphases of conceptual and detail design will require knowledge of a fairly wide range of

expected reliability and cost levels of a large variety of numerical models in order to improve the effectiveness and efficiency of numerical modelling and, thereby, allow the design engineer to spent more effort on the generation and mathematical modelling of designs.

2.2 WHY SEEK HEURISTIC KNOWLEDGE?

In Chapter 1, it was said that due to the complexity of the problem and control spaces, the acquisition of complete knowledge would be too expensive or even impossible. However, even if it were possible for a user to have any desired degree of completeness of the knowledge, there would still be a problem. As the degree of completeness increased, so would the complexity of the knowledge. As the complexity of the knowledge increased, so would the difficulty and expense of applying or using it. At some point the effort to use the knowledge would equal the benefit derived from its use. Beyond this point the effort to use the knowledge would exceed its benefit. The following figure illustrates this situation.

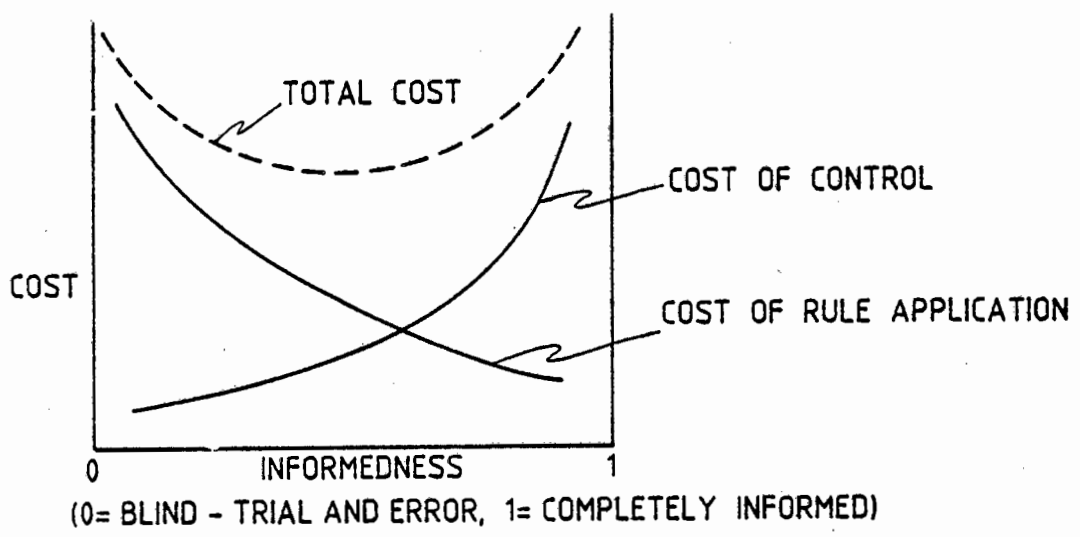


FIGURE 2.2.1 : The cost of control knowledge. From [1].

While this figure refers to the cost of rule application in AI production systems and the cost of using the control knowledge, it is strongly related to knowledge acquisition for numerical modelling since the knowledge sought is for controlling P: for 'informedness'

read 'degree of completeness of knowledge', for 'cost of control' read 'effort to use knowledge' and for 'cost of rule application' read 'computational effort involved in solving numerical model'. Thus it will not usually be worthwhile to have complete (and hence complex) knowledge. HK, because it is derived from simplifications of the problem and control spaces, will necessarily be incomplete but will be much simpler and often worthwhile having.

The fact that HK is incomplete means also that it will be subjective because someone (the KAc) has to decide what the HK will deal with and what it will not deal with. It is this subjective element of HK which allows only relevant and sufficiently simple knowledge to be acquired.

2.3 WHY USE A HEURISTIC PROCEDURE TO ACQUIRE KNOWLEDGE?

Formalists or purists may find the use of heuristics unacceptable but Lakatos in [35] shows clearly the importance of heuristic to pure mathematical discovery. In fact he argues strongly against the possibility of perfect formalisation of any nontrivial field of endeavour. Similarly AI research shows that without heuristics many complex problems cannot be solved within a realistic time limit. Engineering problems are easily so complex and engineers have always used heuristics to solve their problems. Knowledge acquisition problems for engineering software systems are also of such complexity that the only realistic approach is to use heuristic methods to find heuristic knowledge.

The main heuristic elements of the KAProc (knowledge acquisition procedure) presented in Chapter 1 are in the design of the KAProg in the planning phase, in the design of experimental programmes and the pattern recognition in the execution phase and in the organisation of the HK in the integration phase.

In the planning phase, the design of a KAProg requires the selection of a set of PC's and CC's. Because this set does not represent the problem and control spaces completely, its selection will necessarily be subjective. It will, however, be based on quite sound heuristic

principles such as 'decomposition' and successive approximation. By 'decomposition' we mean, for example, that the problem space is 'decomposed' into a set of PC's. In the sense that some PC's will represent more of the problem space than others, the set of PC's will reflect the notion of successive approximation. If the KAc widens his perspective to include the users of the software and of the HK, then some objectivity can be introduced into the selection of the set of PC's and CC's by basing their selection on a survey of user's requirements.

An alternative heuristic approach to acquiring heuristic knowledge is that followed by developers of expert systems [5,6,8,39,40,44,45, 46,48,49]. An expert system to advise users on how to use an engineering package effectively and efficiently would at least need to be able to: establish the nature of the mathematical model required, find a set of suitable numerical models, display the reliability and cost levels that may be achieved, prompt the user to select a reliability level and, finally, output the relevant control variable values that would produce the selected reliability. Explanation facilities may also be necessary. Most knowledge bases for expert systems have been built by eliciting a set of rules (objects, entities, relations, etc., - whichever representation scheme is used [1,2,3,4]) from one or more experts. In the context of this thesis an expert would be some experienced user of the relevant package. The process is almost completely heuristic relying heavily on the judgement of the KAc. At present no agreement exists on techniques for knowledge elicitation. This is partly due to the psychological factors of (i) the way experts (and humans in general) hold their knowledge and (ii) the difficulty of communicating this knowledge. However, for eliciting the rules or relations comprising the knowledge, better or improved techniques based on psychological and psychometric theories are becoming available [41-43]. Another way of deriving a set of rules is to use a set of examples elicited from the expert. If the set of examples is fairly large, induction, hopefully automatic (machine learning [34]), may be used to derive the rules. If the set of examples is fairly small, generalisations based on (deductive) explanations of the examples can be made [50] and these generalisations then formed into a set of rules. These example-based elicitation techniques show considerable promise because of the central role that examples play in the knowledge of experts [47] and in the discovery of knowledge in general [9,35]. The process of

considerable promise because of the central role that examples play in the knowledge of experts [47] and in the discovery of knowledge in general [9,35]. The process of defining PC's and CC's has much in common with these example-based techniques. PC's and CC's are very much like generalisations of example prcblems, but, rather than trying to make the expert's knowledge explicit, his knowledge can be used as a priori knowledge for designing a set of PC's and CC's.

The execution phase of the KAProc deals with individual PC^i and CC^{ij}. Approaches to acquiring HK^{ij} may be divided into deductive, empirical, optimisation and heuristic approaches.

The deductive (rationalist's) approach requires: knowledge of each component (subroutine) which would be active when running P on a problem in the PC, knowledge of their input-output relations and knowledge of their interrelationships. From such knowledge the complete behaviour of P on the PC and CC can be deduced by rigorous mathematical analysis. However, some components may have nondeterministic input-output relations with unknown statistical properties, for example iterative algorithms, in which case, the deduction is not even possible in principle. Even if it were possible in principle, the effort require would make it unprofitable for all but very simple P.

The empirical approach, on the other hand, requires no knowledge of the internal structure of P. It is simply treated as a black box. Statistically designed experiments are conducted on the whole of X^R and F^R and the HK^{ij} inferred from an analysis of the experimental data. While this approach is quite feasible for simple PC^i and CC^{ij} it is wasteful because many of the data cases (experiments) would be inefficient and would not appear in the final HK^{ij}. Tne use of a set of experimental designs beginning with a pilot experiment can of course reduce the wastage significantly. In [54] a similar, quite efficient, procedure is used to find approximations to a Pareto optimal set. The approach in our KAProc clearly resembles such a procedure.

An optimisation approach combines elements of empiricism and rationalism into a search procedure, i.e. experiments are made, gradients, etc., are calculated to indicate search directions, a step in the search direction is taken and the whole process repeated until an optimum is found. Such procedures are quite efficient for finding a single optimal point, say where reliability is fixed and the control for minimum cost is to be found. When a Pareto optimal set is to be found it would be inefficient because no knowledge or data used to find a single optimum would (normally) be used to find any other optimal point. Of course, by using a previously found optimum as a starting point for the search for a subsequent optimum, the procedure could be made more efficient. The use of response surface methodology [13] can also improve the efficiency of optimisation approaches.

There are two main problems with the above approaches. Firstly, they may produce complex characterisations of the Pareto set which would then still have to be simplified to produce a relatively simple set of rules. Secondly, they are wasteful because they are objectivist approaches and make very little or no use of any a priori knowledge available to the KAc, except as a check on the validity of the HK^{ij}. The KAProc (execution phase) presented in this thesis, however, tries to search directly for simple rules and tries to make full use of any a priori knowledge of the expected behaviour of P to design experimental programmes on X^R and F^R and to recognise patterns in the data. Of course, this makes it subjectivist and so requires intelligence. For the near future, the intelligence needed to design heuristic experimental programmes and recognise patterns by using a priori knowledge will have to be supplied by the KAc. The KAProc must therefore remain semi-automatic in the near future. As research into artificial intelligence, pattern recognition and machine learning uncovers useful appropriate ideas, the KAProc may be increasingly automated.

The heuristic element in the integration phase mainly concerns the way HK is formed from $U_{i,j}$ HK^i, HK^{ij}. Many of the HK^i and HK^{ij} may have similar forms and explanations. It may, therefore, be possible to simplify the set of rules by clustering PC's or CC's and hence their

HK^i or HK^{ij}. However, recognising these similarities and performing the clustering will very likely require intelligence and judgement, i.e. a heuristic approach.

2.4 DISSEMINATION OF THE HEURISTIC KNOWLEDGE

At least three forms of disseminating the HK may be identified, namely

(i) as accompanying documentation with the standard user documentation for P,

(ii) built into P, or

(iii) in an expert system.

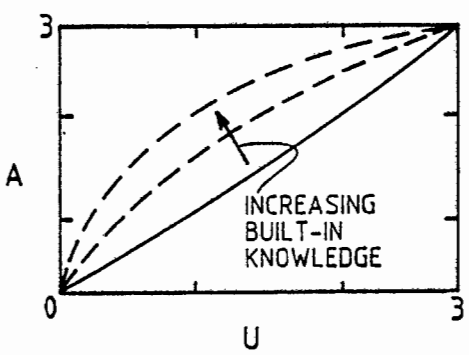
Option (i) is clearly the most obvious form. However, the example problems contained in the standard user documentation already constitute a large part of the documentation. If these examples are upgraded into problem and control classes, the documentation may become unwieldy. Computational costs will usually vary between installations and this aspect of the HK would therefore require recalibration for each installation. While the calibration is expected to be relatively simple, the dissemination of installation dependent HK in paper form would probably be impractical. Of course, even without calibration, costs quoted in terms of some industry 'standard' like a VAX 11/780 would still be of enormous benefit. The modern approach, however, is towards computer-based dissemination, i.e. options (ii) and (iii), even for standard documentation.

Building the HK directly into the software (option (ii)) may well be the best form of dissemination because it automatically reduces the user's effort. It is, however, restricted to rules whose antecedents can easily be identified from the data input or from special performance indicators built into the numerical algorithms and to control regimes which produce very highly reliable solutions. Only relative cost improvements are of importance for this form of

dissemination. Many such rules exist in engineering packages, for example, preselected numerical integration rules, local iteration convergence tolerances and hourglass stabilisation schemes.

Finally, the HK can be disseminated in an expert system. Expert systems are becoming an increasingly popular form of disseminating (and storing) knowledge. To date most expert systems have been rule-based [5,6,8]. Rules have the advantage that they are easy to understand, easy to build into conventional software (IF...THEN...ELSE...) and many rule-based expert system development aids (shells) are available [5,9]. Thus by choosing to represent the HK as rules we have allowed for dissemination by any of the three forms listed above. The KAProc has additional relevance to the development of expert systems because by providing a methodology for knowledge acquisition it should alleviate the effect of the knowledge acquisition 'bottleneck' on system development. It does not address the problems of ascertaining the nature of the mathematical model nor the problem of matching the mathematical model to an appropriate problem class but it does provide a sound framework for acquiring the basic rules as well as for structuring the rule-base (Inadequately structured rule-bases can result in excessive computational effort for rule processing).

An interesting way to view the effect that the different dissemination forms will have is via Figure 2.4.1. (This figure is based on ideas and Figures in [76]).



A = application difficulty
(0 - trivial; 1 - easy:
2 - rather difficult;
3 - limit of software's
scope)
U = user proficiency
(0 - untrained; 1 - novice,
2 - journeyman; 3 -expert)

FIGURE 2.4.1 : Reliability profile for an engineering package

The solid line in Figure 2.4.1 represents the fact that as application difficulty increases, required user proficiency increases proportionally. This would be the case for a package with normal reliability [76]. User proficiency here includes both mathematical and numerical modelling skill. Increasing the quality and quantity of numerical modelling knowledge specific to the software will have different effects depending on how the knowledge is disseminated. If built into the software it will have the effect of raising the curves as indicated by the broken curves. If supplied directly to the user it will, like education and training, shift the user along the U-axis. Figure 2.4.1 also illustrates the fact that there will always be problems an untrained person cannot solve and there will always be problems that require experts to solve. Therefore, it will be pointless pitching the HK at too low a level but also it will be pointless to pitch it at too high a level (the effort involved will exceed the cost of consulting an expert).

CHAPTER 3 : THE KNOWLEDGE ACQUISITION PROCEDURE AND THE KAS

This chapter provides a more detailed account of the knowledge acquisition procedure presented in Chapter 1. It is assumed that the reader is familiar with the concepts and terminology already introduced. The following section summarises the knowledge acquisition procedure and points out the main modifications that will be made to it in this chapter.

3.1 THE KNOWLEDGE ACQUISITION PROCEDURE

Recall that the knowledge acquisition process may be represented as in Figure 3.1.1.

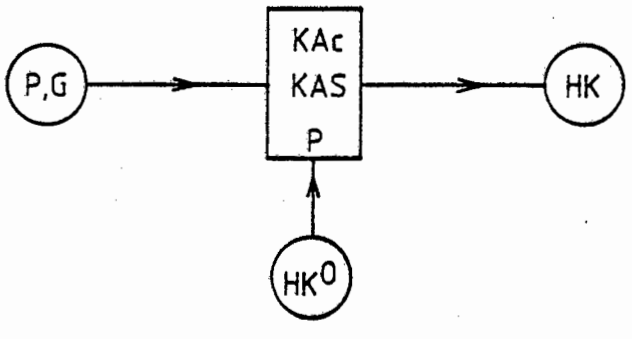
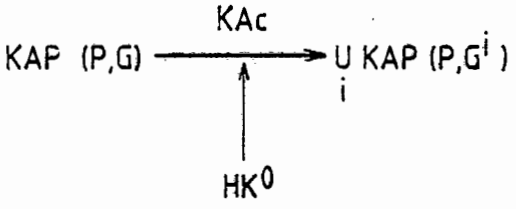


FIGURE 3.1.1 : An abstract view of the knowledge acquisition process

The procedure involved three stages, namely planning, execution and integration phases as shown in Figure 3.1.2.

(i)



(ii) For each i:

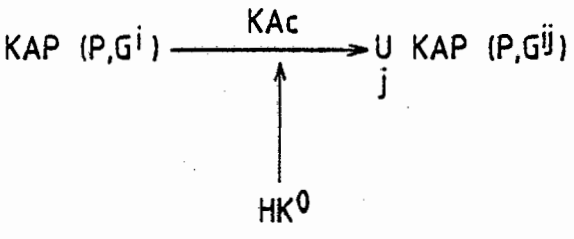


FIGURE 3.1.2.(a) : The planning phase of knowledge acquisition

For each i and for each j:

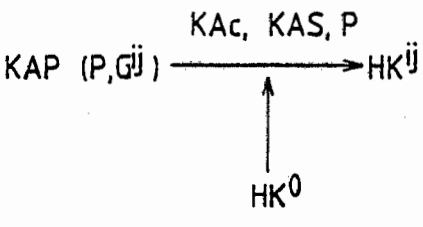
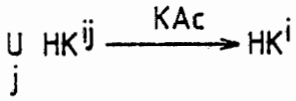


FIGURE 3.1.2(b) : Execution of each of the knowledge acquisition subproblems

(i) For each i:



(ii)

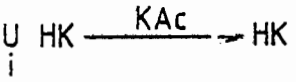


FIGURE 3.1.2(c) : Integrating the separately acquired HK^{ij} and HK^{i} into HK for the original problem

However, we need to modify the detailed view of the execution phase given in Figure 1.3.3. This figure gives the impression that knowledge acquisition is a straight forward process whereas we have already indicated that much feedback for knowledge refinement occurs. So we need to add feedback paths. The other modification is to replace the system $S^0 = (X^0, F^0, Y^0)$ by the more complex system $\sum^0$. Detailed discussion of the contents of $\sum^0$ will be deferred until section 3.3. For the moment, it will suffice to say that conceptually the contents of $\sum^0$ is similar to that of S^0. With these modifications the symbolic view of the execution phase is now as shown below.

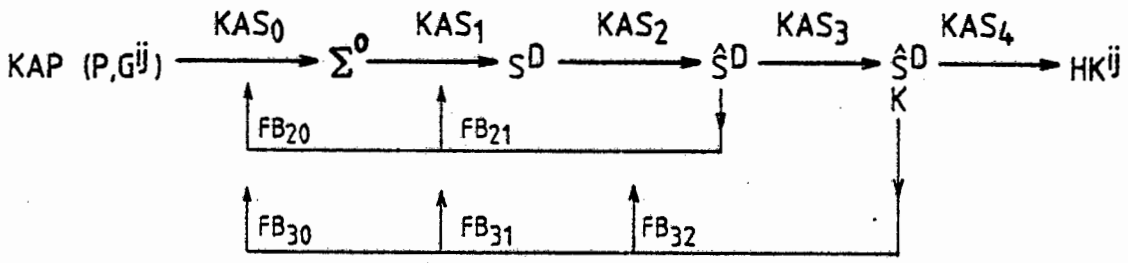


FIGURE 3.1.3 : Detailed symbolic view of the execution phase

In the above figure FB_{ij} means feedback from KAS_i (or its products) to KAS_j.

The planning phase results in a set of knowledge acquisition subproblems with goals G^i and G^{ij} specified with respect to a problem class PC^i and control classes CC^{ij}. In section 3.3, it will be seen that the G^{ij} actually consist of eight goal conditions which must be satisfied simultaneously. Following on from this, the next five sections (3.4 to 3.8) deal with the five main steps in the execution phase and the corresponding components of the KAS. Although the feedback loops in the execution phase are discussed together with the five main steps, a separate section (section 3.9) brings together these feedback aspects and simultaneously summarises the execution phase. The last section deals with the integration of the individually acquired HK^{ij} into HK.

3.2 THE PLANNING PHASE AND THE GOAL G

The Goal G

Knowledge acquisition begins with the problem KAP(P,G) where G is:

"Find HK characterising an approximate Pareto optimal set for P".

To understand this statement of G we need to explain the terms "Pareto optimal set", "approximate Pareto optimal set" and "characterising" in some detail.

"Pareto optimal set"

A multi-objective optimisation problem, such as: maximise both u and v simultaneously, will not usually have a unique solution. Instead a set of (u,v) involving the best trade-offs may be found and, depending on factors not involved in the optimisation problem, a preference is selected. This set of best trade-offs is the Pareto optimal set [54]. Figure 3.2.1 illustrates some Pareto optimal (non-inferior, non-dominated, efficient) and inefficient (inferior, dominated) points.

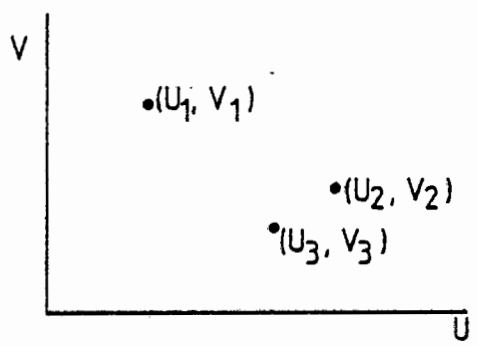


FIGURE 3.2.1 : (u_1,v_1) and $u_2,v_2)$ are Pareto optimal if the objective is to maximise (u,v). (u_3,v_3) is dominated by (u_2,v_2) but not by (u_1,v_1).

Mathematically, a Pareto optimal point (u*,v*) is one for which no other (u,v) exists for which either $u > u^*$ and $v \geq v^*$ or $u \geq u^*$ and $v > v^*$. Obviously this is easily extended to higher dimensions and to situations where u is to be maximised while v minimised. In knowledge acquisition the optimality criteria will usually be given in terms of some well-defined measures of costs and reliabilities. More than one

cost and/or reliability measure may be involved. The main thing to note is that the criteria for optimality, i.e. which features of the output are to be maximised and which are to be minimised, form part of the definition of the goals, G, G^i and G^{ij}. A major source of difficulty is that no single set of criteria can cover all efficient uses of P.

"Approximate Pareto optimal set"

In Chapter 1 we mentioned that we would not seek Pareto optimal sets but, instead, will seek approximate Pareto optimal sets. An element of an approximate Pareto optimal set has a high probability of lying sufficiently close to an element of the Pareto optimal set, for most practical purposes. It is possible to define objectively the terms "high probability", "sufficiently close" and "most practical purposes", but such objective definitions will only result in another (slightly relaxed?) formal approach with all its associated difficulties. Instead we will rely on subjective definitions. The question is whose subjectivity? Clearly the developer of P, the users' club of P, expert users of P and the users of the desired HK are most appropriate since they will know what "most practical purposes", "sufficiently close" and "high probability" are. It is the subjective elements in the definition of approximate Pareto optimal set which are primarily responsible for the fact that the knowledge sought will be heuristic. There are, however, further reasons why the knowledge will be heuristic. These reasons are implicit in the term "characterising".

"Characterising"

The HK that is sought must, in addition to describing an approximate Pareto optimal set, also be adequate, simple, robust and credible. Again these are subjective properties required of the HK. The word "characterising" is used to capture these additional properties.

Goals G^i and G^{ij} are defined very similarly to G but instead of applying to the whole problem space, G^i applies only to PC^i while G^{ij} applies only to PC^i and CC^{ij}. No further clarity is obtained by

expanding on G^i. Expanding on G^{ij}, however, is essential and will be dealt with in section 3.3. Before this, we need to explain in more detail how a KAProg is defined.

Designing a KAProg

Recall that the KAProg was defined as

$$KAProg = \bigcup_{i,j} KAP(P,G^i),\ KAP(P,G^{ij})$$

where G^i relates to PC^i and G^{ij} to PC^i and CC^{ij}. Thus it requires the selection of a set of PC's and CC's. In order to make this selection, the KAc can draw heavily on analogous procedures and heuristics used in design evaluation.

Recall that the DEProc could be represented (in section 1.3) as shown below.

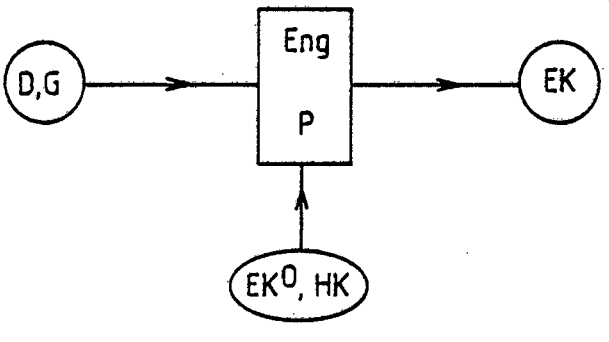
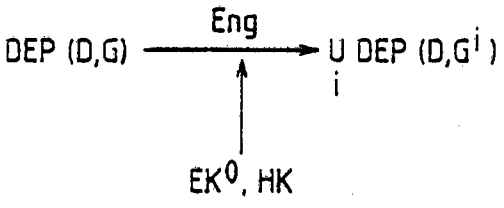


FIGURE 3.2.2 : An abstract block diagram view of the design evaluation process

In almost identical fashion to the KAProc, the DEProc is composed of a planning phase, an execution phase and an integration phase. These three phases are summarised in Figure 3.2.3. Of particular importance is the planning phase of the DEProc and we will later return to it.

(i)



(ii) For each i:

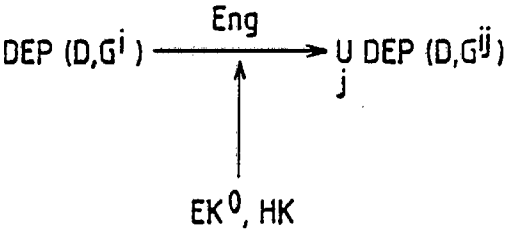


FIGURE 3.2.3(a) : The planning phase of the DEProc (Cf. Figure 1.3.1)

For each i and for each j:

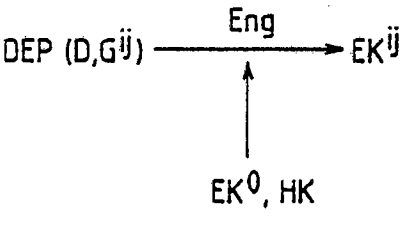
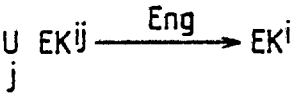


FIGURE 3.2.3(b) : The execution phase of the DEProc (Cf. Figure 1.3.2)

(i) For each i:



(ii)

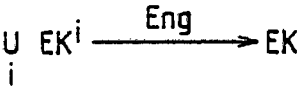


FIGURE 3.2.3(c) : The integration phase of the DEProc (Cf. Figure 1.3.5)

Just as the planning phase of the knowledge acquisition procedure (KAProc) replaced KAP(P,G) by a set of KAP(P,G^i) and KAP(P,G^{ij}), the planning phase of the DEProc, shown symbolically in Figure 3.2.3(a), replaces DEP(D,G) by a set of DEP(D,G^i) and DEP(D,G^{ij}).

Each goal G^i is to find EK^i. EK^i is an evaluation of some behavioural aspect of some component of D. The relevant behavioural aspects of the component are predicted via a mathematical model MM^i. Each of the MM^i may need to be solved by a set of numerical models NM^{ij} for each of which the goal G^{ij} is to find EK^{ij}. G^{ij} is really just the transference of G^i from MM^i to NM^{ij}. Thus, while in knowledge acquisition, the G^i relate to PC^i (classes of mathematical models) and the G^{ij} relate to CC^{ij} (classes of numerical models), in design evaluation, the G^i relate to particular MM^i and the G^{ij} to particular NM^{ij}.

The objective in the execution phase, shown symbolically in Figure 3.2.3(b) is the acquisition or generation of EK^{ij}, i.e. evaluative knowledge relating to G^{ij}. Generally, it will involve running an engineering package to solve the NM^{ij}, collecting the output and interpretting this output so that it may be condensed to some simpler statement EK^{ij} about the design. Note however the use of a priori evaluative knowledge EK^0. By EK^0 in the above Figure we do not mean only EK derived from manual calculation. All EK acquired during the execution phase up to the execution for a particular i and j is included in the EK^0 for these i and j.

The objective in the integration phase, shown symbolically in Figure 3.2.3(c), is to combine the individually acquired EK^{ij} into EK^i, i.e. an evaluation of the D based on MM^i, and to combine the EK^i into an evaluation of D as a whole.

Returning to the planning phase of the design evaluation procedure, the combination of all design evaluation subproblems forms the design evaluation programme (DEProg), i.e.

$$\text{DEProg} = \bigcup_{i,j} \text{DEP}(D,G^i),\ \text{DEP}(D,G^{ij})$$

Of course, in actual design evaluation, the programme would also contain hand calculations, physical experiments, prototypes and anything else considered helpful. In this thesis however we will only consider the above narrow definition of a DEProg.

The design of a DEProg relies on a number of heuristics the most important of which are:

H1 Start with a set of simple mathematical models.

H2 For each mathematical model which requires numerical solution, start with a simple numerical model.

H3 Add detail or complexity to the model in such a way that successive approximations become closer to the true behaviour of the design.

H4 Break complex models up into interacting simpler submodels.

In short, the DEProg should reflect the good old-fashioned notions of successive approximation and decomposition. By decomposition we include partial modelling as well as H4 above. Partial modelling [17] is understood as modelling only a part of the expected behaviour, e.g. approximating 3-D objects by a set of 2-D models. It amounts to modification (relaxation, removal, addition) of constraints on the design requirements. Such constraint modification [2] is a very useful heuristic for simplifying complexity, be it in design evaluation or knowledge acquisition. In fact not only should the DEProg reflect these notions but the process of designing the DEProg should also reflect them, i.e. the engineer will design only a simple DEProg, execute it, integrate the acquired EK, modify the DEProg by adding more models, and so on. Clearly like any such successive approximation the process will be one of learning - in this case learning the behaviour of D and how it measures up to its specification.

We expect that normally the engineer designing the DEProg would start by selecting a set of mathematical models MM^i. Some might be partial models of D and others might be models of components of D. If a diagram representing the relation of the MM^i to each other could be drawn, we would typically expect to see the following.

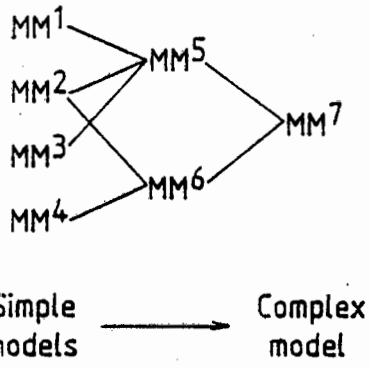


FIGURE 3.2.4 : A typical hierarchy/network of MM^i in the DEProg.

The 'hierarchy' will usually be quite deep because components models would usually be assembled into increasing complex models and perhaps ultimately into a model for D as a whole. To solve MM^i numerically a number of NM^{ij} would typically be used. It is in this selection of the NM^{ij} that the HK plays its part. If the MM^i falls within one of the PC^i for which HK^i has been acquired then the engineer can select, effectively and efficiently, a set of NM^{ij} to solve the MM^i. Usually this will require at least two NM's - an initial crude NM with low cost and presumably low reliability and a refined NM which can produce the required level of reliability at some acceptable cost. A typical set of NM^{ij} might be depicted as follows.

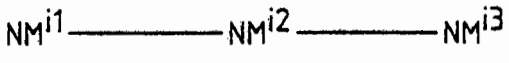


FIGURE 3.2.5 : A typical set of NM^{ij} reflecting the principle of successive approximation

A very important aspect of designing a DEProg is the use of a priori evaluative knowledge EK^0, normally derived from manual calculations. It will act as a guide on the level of reliability required and a check on the solutions.

In designing the KAProg, the KAc must be aware of the typical composition of DEProgs. Ideally every MM^i in a DEProg should fall into at least one PC. However, this would require very complex PC's to cater for the complex MM's. Instead the PC's must be limited to

catering only for most of the simpler MM^i. If the relation between the PC^i could be depicted graphically, we would typically expect the following.

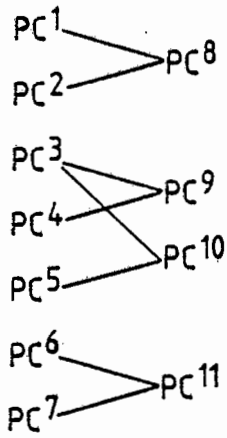


FIGURE 3.2.6 : A typical set of PC^i in a KAProg

This shows a fairly flat broad 'hierarchy' - flat due to the necessity of keeping the PC^i relatively simple but broad to cater for most of the expected simpler MM^i.

The control subspace for each PC^i may still be quite complex and to simplify knowledge acquisition it is replaced by a set of CC^{ij}. A typical set of CC^{ij} for PC^i might be related as depicted in Figure 3.2.7.

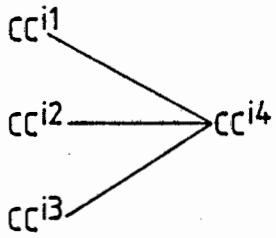


FIGURE 3.2.7 : A typical set of CC^{ij} in a KAProg

The relation depicted in figure 3.2.7 might have been derived as follows. The KAc decided that CC^{i4} is the closest approximation to the control subspace for PC^i for which knowledge acquisition is manageable. However, CC^{i4} is itself too complex to study directly so the KAc decided to study three simpler CC's first. These would then provide him with improved HK^0 for the study of CC^{i4}.

Some additional heuristics to H1-4 that may be useful in designing KAProgs are the following.

H5 The total number of variables in X^N and F^N together should be less than about ten. Beyond this the problem/control classes may become too complex. For cluster analysis [20] says that realistic numbers of variables are between 2 and 10 and realistic numbers of data cases are 50 to 1000.

H6 Strong variable interactions should be contained within a problem and/or control class and not straddle class boundaries.

H7 Discontinuities in the response of P may form natural boundaries for problem or control classes.

3.3 THE GOALS G^{ij}

Similar to the goal G, the subgoals G^{ij} can be defined as:

"Find HK^{ij} characterising an approximate Pareto optimal set for PC^i in terms of CC^{ij}".

However, a more precise formulation of G^{ij} is possible. We already hinted in Chapter 1 that it would involve finding a set of subsets of $\hat{S}^R = (\hat{X}^R, \hat{F}^R, \hat{Y}^R)$. G^{ij} must be defined on the transformed system $\hat{S}^D$ and not S^D since only in $\hat{S}^D$ do reliabilities and other solution features appear. Although X^D and F^D are also transformed to $\hat{X}^D$ and $\hat{F}^D$ this will usually be a very simple, for example, $\hat{X}^D = X^D$ and $\hat{F}^D = F^D$.

The $k\ell$'th subset of $\hat{S}^R$ is $\hat{S}^R_{k\ell} = (\hat{X}^R_k, \hat{F}^R_{k\ell}, \hat{Y}^R_{k\ell})$. If $k=1,2,\ldots,K$ and $\ell=1,2,\ldots,L(k)$ then the total number of subsets N(K) will be given by

$$N(K) = \sum_{k}^{K} L(k)$$

The set of these subsets is denoted by

$$\hat{S}_K^R = \{\hat{S}_{k\ell}^R \mid k=1,2,\ldots,K,\ \ell=1,2,\ldots,L(k)\}$$

For each $\hat{S}_{k\ell}^R$ there will be one rule in the HK^{ij}.

G^{ij} then becomes:

Choose $\hat{S}_K^R$ such that the following conditions are satisfied:

C1 : The HK^{ij} must apply to the whole of $\hat{X}^R$.

C2 : The HK^{ij} must give an adequate range and resolution level of solution features (cost and reliability) for decision making.

C3 : Only relatively efficient solutions should be considered.

C4 : The worst error made in using the HK should be a minimum.

C5 : All $\hat{F}_{k\ell}^R$ should be as large as possible.

C6 : The total number of rules in the HK^{ij} should be a minimum.

C7 : The subsets and hence each rule should be simple.

C8 : The HK^{ij} should be explainable to a user of P.

These eight conditions may be made more precise as follows:

C1 : $\bigcup_k^K \hat{X}_k^R = \hat{X}^R$.

C2 : $\bigcup_\ell^{L(k)} \hat{Y}_{k\ell}^R \approx \hat{Y}^R$, for each $k = 1,2,\ldots,K$.

C3 : The elements of the $\hat{S}_{k\ell}^R$ should be approximately Pareto optimal

C4 : $\min^{\hat{R}}[\max_{k,\ell} \Delta Y_{k\ell}]$.

C5 : $\max [\min_{k,\ell} \Delta \hat{F}^{R}_{k\ell}]$.

C6 : min [N(K)].

C7 : The subsets $\hat{X}^{R}_{k}$, $\hat{F}^{R}_{k\ell}$, $\hat{Y}^{R}_{k\ell}$ should each be simply connected and preferably convex.

C8 : The structure of $\hat{S}^{R}_{K}$ should be explainable.

$\Delta \hat{Y}^{R}_{k\ell}$ represents a measure of the worst error made by using rule $k\ell$.

$\Delta \hat{F}^{R}_{k\ell}$ represents a measure of the size of $\hat{F}_{k\ell}$.

C1 and C2 may be considered as adequacy conditions while C3 is an efficiency condition. C4 is a type of accuracy or reliability condition. If robustness is defined as the ability to tolerate error, then C5 may be thought of as a type of robustness condition. Also, by making the $\hat{F}^{R}_{k\ell}$ as large as possible, one may increase the number of data cases in $\hat{F}_{k\ell}$ and thereby improve the experimental representation of $\hat{F}^{R}_{k\ell}$. Thus C5 can also be thought of as a condition which improves the empirical basis of the HK^{ij}. C6 and C7 are clearly simplicity conditions. C8 may be considered to be a credibility condition because lack of explanation may mean that an error has been made or some aspect overlooked. This is less likely if an explanation of the HK^{ij} in terms of physical and computational theory. C8 may also be thought of as a validity check condition.

The set $\hat{S}^{R}_{K}$ will be called a 'partition' of $\hat{S}^{R}$. However, a 'partition' here does not mean the usual definition of a hard partition, i.e. that subsets do not overlap and that, taken together, they cover the whole of the original set. Instead we allow overlap (as with fuzzy subsets) if it will simplify the resulting rule set and while

$U_k^K \hat{X}_k^R = \hat{X}^R$ is necessary, we require only that $U_\ell^{L(k)} \hat{Y}_{k\ell}^R \approx \hat{Y}^R$ for each k and that the $\hat{F}_{k\ell}^R$ be as large as possible, i.e. we allow incomplete coverage of $\hat{Y}^R$ and $\hat{F}^R$. Enforcing complete coverage of $\hat{F}^R$ and $\hat{Y}^R$ by the $\hat{F}_{k\ell}^R$ and $\hat{Y}_{k\ell}^R$ is incompatible with Pareto optimality because inefficient solutions would have to be included.

In graphical terms, the $\hat{S}_K^R$ sought can be illustrated as follows. Suppose that each of $\hat{X}^D$, $\hat{F}^D$ and $\hat{Y}^D$ consists of a single variable with ranges $\hat{X}^R$, $\hat{F}^R$ and $\hat{Y}^R$. Then in the $\hat{X}^R$ X $\hat{Y}^R$ space one would typically want the following.

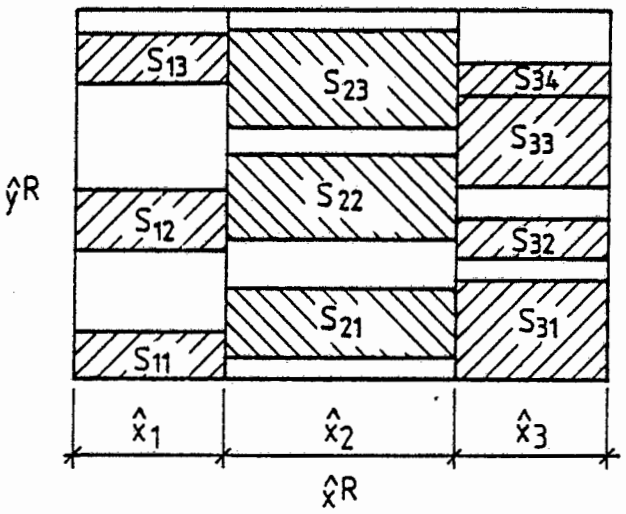


FIGURE 3.3.1 : $\hat{S}_K$ projected onto $\hat{X}^R$ X $\hat{Y}^R$ plane

Note that, in the above figure, $U_k \hat{X}_k^R = \hat{X}^R$. For each $\hat{X}_k^R$, the $\hat{Y}_{k\ell}^R$, 'span' (approximately) the whole of $\hat{Y}^R$, i.e. $U_\ell \hat{Y}_{k\ell}^R \approx \hat{Y}^R$. Also indicated in this figure is the fact that one is unlikely to achieve or desire the highest value in $\hat{Y}^R$ - values close to the highest are sufficient. One will often achieve the lowest value because one may specify some cut-off below which solutions are not good enough. Also the achievable $\hat{Y}_{k\ell}^R$ are not continuous - the empty spaces imply inefficient use.

Corresponding to the above one might have the following picture in the $\hat{X}^R$ X $\hat{F}^R$ plane.

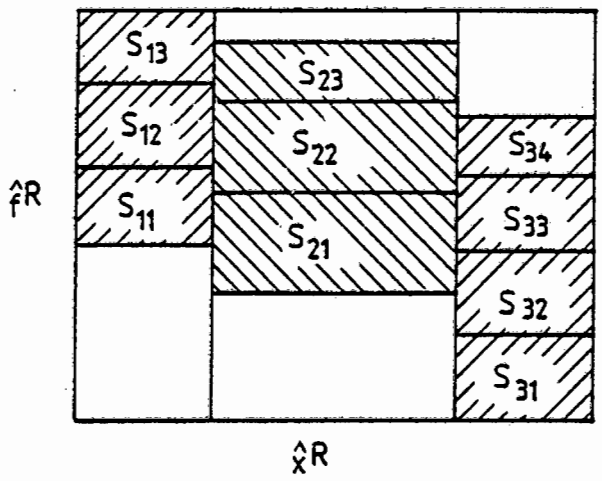


FIGURE 3.3.2 : $\hat{S}_K^R$ projected onto the $\hat{X}^R$ X $\hat{F}^R$ plane

In the $\hat{F}^R$ X $\hat{Y}^R$ plane, one would see the following for $\hat{X}^R$.

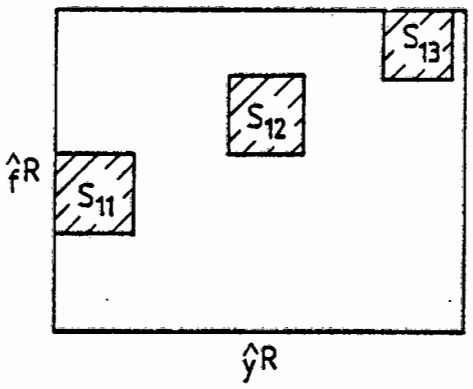


FIGURE 3.3.3 : A $\hat{Y}^R$ X $\hat{F}^R$ section of $\hat{S}_K^R$ for $\hat{X}_1^R$

Note that in the above the 'partitionings' have been hard. With fuzzy 'partitioning', the boundaries of the $\hat{S}_{k\ell}^R$ may overlap. In some situations this may simplify the acquisition of knowledge and may even simplify the knowledge itself.

With so many conflicting conditions on the HK^{ij} (or $\hat{S}_K^R$), the acquisition of HK^{ij} is, itself, clearly a Pareto optimal problem.

As mentioned in the introduction, 'partitioning' of $\hat{S}^R$ into $\hat{S}_K^R$ is to be done by induction from patterns in experimental data, $\hat{S}$. These patterns are regarded as the manifestation of some underlying 'natural' 'partitioning' of the data into subsets

3.4 KAS_0, $\sum^0$ AND FB_{i0}

The objective of KAS_0 is to define $\sum^0$. It is really a way of defining the problem and control class in a more precise fashion. Most of the rest of this section will centre on the contents of $\sum^0$ and at the end will deal with FB_{i0}.

Three changes need to be made to the old $S^0 = (X^0,F^0,Y^0)$ to arrive at $\sum^0$.

Firstly, many of the engineering packages, which this thesis considers as potential targets for knowledge acquisition, solve field and time dependent problems. The distributions of variables in such problems are usually related to space and time. Space and time variables do not really belong to any of X^0,F^0 or Y^0. So we add another variable set called I^0 to make $S^0 = (X^0,F^0,Y^0,I^0)$, where $I^0 = (I^N,I^R)$. I^N is the set of index variable names and I^R their ranges. Each index variable may be referred to as space-like or time-like. The more neutral terms index, space-like and time-like rather than space coordinates and time coordinate are used because, in other non-field problem packages, similar variables, over which outputs are distributed, may be found. Generally all outputs will be distributed over (linked to) a time-like index while only a few will be linked to space-like indices.

The second change is to distinguish between problem class attributes and variables and between appearances of attributes and values of variables. Attributes should be viewed as interpretations of variables and, simultaneously, variables should be viewed as the means of measuring attributes. Corresponding to the value range of a variable is an appearance range of an attribute, i.e a set of all possible appearances of the attribute. The correspondences between variables and attributes and between their value and appearance ranges is called a homomorphism. It forms the third change. Attributes will be denoted by X^0,F^0,Y^0 and I^0 and the homomorphism by β. As an example suppose an attribute is 'structural aspect ratio' and its appearance set is {'low', 'medium', 'high',} while the variable which will be used to measure it is called 'SAR'. SAR has a value set of [1.0, 100.0] say. β might then be defined by the following table.

Structural Aspect Ratio	SAR
Low	1.0 to 5.0
Medium	5.0 to 30.0
High	30.0 to 100.0

Attributes and appearances are much more like adjectives or adverbs of a natural language than are variables and values. They relate more closely to concepts while variables relate more to technicalities. Explanation, an important aspect of the integration phase and of the HK itself, is usually more easily understood in conceptual terms like attributes and appearances. Their inherent lack of hardness is in fact their strength because, while measuring techniques may be altered, the concepts involved may be retained. For example, the concept of reliability is largely fixed but for almost every problem class its measurement will differ. This lack of hardness also means that explanations are intuitive (heuristic) rather than deductive. Usually the homomorphism will be obvious, as in the example above. In some circumstance it may however be useful to define fuzzy correspondences [55,10].

Putting the above together, we can now define

$$\sum^0 = (S^0, S^0, \beta)$$

where $S^0 = (X^0, F^0, Y^0, I^0)$

$S^0 = (X^0, F^0, Y^0, I^0)$

$X^0 = (X^N, X^R)$, $F^0 = (F^N, F^R)$, etc

$X^0 = (X^N, X^R)$, $F^0 = (F^N, F^R)$, etc

and $\beta = (\beta^N, \beta^R)$

is the homomorphism defining correspondences β^N between the S^N and S^N and correspondences β^R between S^R and S^R.

In the above, S^0 is called the <u>object system</u>, S^D is called the <u>image system</u> and $\sum^0$ is called the <u>source system</u>. All three are <u>primitive systems</u> - hence the superscript ^{0}. The above terminology and concepts are inspired by KLIR [55] which deals with 'General Systems Concepts'.

Now on to feedback. Figure 3.1.3 shows two feedback paths FB_{20} and FB_{30} entering KAS_0. FB_{20} stems from knowledge derived during KAS_2. Mostly its presence serves to convey the idea that variables for measuring selected features of the output need to be defined in S^0, e.g. if a feature of the output to be measured is reliability, then a variable called RELY (say) must be defined in S^0.

FB_{30} serves to convey the fact that redefinition of $\sum^0$ may be required before suitable patterns in $\hat{S}$ may be found or the HK explained satisfactorily. The latter may require redefinition of either S^0, S^0 or β. In this sense knowledge fed back along FB_{30} has exactly the same function as a priori knowledge except that the latter involves foresight while the former involves hindsight. FB_{30} also deals with the definition of the β mappings which connect appearance sets and value sets. Many of these may only be defined during the integration phase.

3.5 KAS_1, S^D AND FB_{i1}

The objective here is to collect data to form the data system

$$S^D = \{X^D, F^D, Y^D, I^D\}$$

where $X^D = (X^N, X^R, X)$ and so on for F^D, Y^D and I^D.

S^D consists of S^0 plus the set of data cases - one case for each run of P. The $\alpha\beta$'th case of data consists of $S_{\alpha\beta} = (X_\alpha, F_{\alpha\beta}, Y_{\alpha\beta}, I_{\alpha\beta})$. Each of these data values must lie within the relevant value range X^R, F^R, etc. The set S of data cases is then $S = (X,F,Y,I)$ or $S = \{S_{\alpha\beta} | \alpha = 1,\ldots,n_X,\ \beta = 1,\ldots,n_f(\alpha)\}$, where n_X is the number of cases with distinct x_α and $n_f(\alpha)$ is the number of cases run for x_α with controls $F_{\alpha\beta}$.

A major problem is that the amount of data in a single case $S_{\alpha\beta}$ may be large especially for P dealing with field problems. Of course, by intelligent selection of variables the amount of data may be significantly reduced. However, this is seldom really possible nor desirable and is in fact the reason for computerisation of the KAS. With the CBKAS, such selection may be deferred until a better picture

of the behaviour of P on the problem and control class is obtained. Clearly, even for small problem and control classes, the volume of data in S^D may be very large. It is therefore very important that experimental programmes on the problem class be carefully designed.

Recall that the HK^{ij} involves subsets of $\hat{S}^R$ which characterise an approximate Pareto set for the problem class. Design of an experimental programme therefore consists of selecting suitable samples of $\hat{X}^R \times \hat{F}^R$. However, to run P, samples of $X^R \times F^R$ are needed, not the transform $\hat{X}^R \times \hat{F}^R$, and so the programme must in fact be selected with the relevant transformations borne in mind. Fortunately, it is often the case that $\hat{X}^D = X^D$ and $F^D = F^D$ and only $\hat{Y}^D$ contains transformations of Y^D (such as reliabilities). KAS_1 then consists of designing an initial programme of experiments on $X^R \times F^R$ plus subsequent programmes via feedback (FB_{21}, FB_{31}) of knowledge already acquired.

The initial programme should ideally span the whole of $X^R \times F^R$. Here the statistical designs used in Response Surface Methodology (RSM)[13] and simulation [15,16] are applicable. However, one major difference is that no experimental measurement error is involved in collecting the data from P. This simplifies the design. In particular, it removes the necessity of randomizing the order in which the runs are executed. However, the fact that many packages use iterative processes means that quasi-random elements may appear in responses. Although RSM techniques may not be used, it is very useful, when designing experimental programmes, to conceptualise a priori (expected) responses in terms of response surfaces. Smooth, continuous and nonlinear trends in response are very important considerations. For example, if there n <u>independent</u> variables in $\hat{X}^D$ and m in $\hat{F}^D$ and the response of $\hat{y}_i^N \in \hat{Y}^N$ is expected to be of first order, i.e.

$$\hat{y}_i^N = a + b_1\hat{x}_1^N + \ldots + b_n\hat{x}_n^N + c_1\hat{f}_1^N + \ldots + c_m\hat{f}_m^N + \text{error},$$

then a factorial design [13] with just two levels will have 2^{n+m} runs, i.e. all permutations of the extreme values of $\hat{X}^R$ and $\hat{F}^R$. This would only constitute the <u>initial</u> design. Any nonlinearity or discontinuity increases the number dramatically and more complex designs are needed

[13]. If during the selection of features via KAS_2, knowledge of such nonlinearities, etc. becomes available, it is fed back via FB_{21} to KAS_1.

To reduce the number of experiments one may consider splitting the variables into groups. For example, if F^0 is split into F_1^0 and F_2^0 containing m1 and m2 variables respectively, the number of experiments reduces to $2^n(2^{m1}+2^{m2})$. Of course, this is only truly valid if the variables in F_1^0 and F_2^0 do not interact, but it may be adequate. By studying only those parts of $\hat{X}^R$ which produce the worst response (high cost and low reliability), experimentation may be further reduced. The HK^{ij} would then be generalised (conservatively) to the rest of $\hat{X}^R$. Of course, knowledge of such areas of $\hat{X}^R$ may not be available. The replacement of the problem and control spaces of P by simpler PC's and CC's is based fundamentally on the same argument. Conceptually, the splitting of F^0 into F_1^0 and F_2^0 is identical to the redesign of the KAProg replacing the CC defined by F^0 with two new CC's derived from F_1^0 and F_2^0.

Subsequent experimental programmes are required for two possible reasons. Firstly, to focus further experiments into regions of $\hat{F}^R$ which appear to contain the Pareto set. Secondly, if F^0 had previously been split and HK^{ij} acquired independently for the two subclasses, the KAc may subsequently consider interaction of the two in some limited further experimentation on the whole of F^0. The knowledge required for such focussing is derived from the $\hat{S}$ patterns and fed back via FB_{31}. In particular it is derived from comparing Pareto filtered $\hat{S}$ data with unfiltered data (i.e. use of positive and negative examples to learn about the region of Pareto optimality).

Another important heuristic splitting technique is to design an experimental programme on X^R, then select some typical X_α in the programme and acquire HK for this X_α alone (this requires another programme on F^R and all the rest of the steps in the knowledge acquisition procedure). Then using this HK as a guide, knowledge acquisition on the other X_α in the programme will hopefully be less costly. If the interaction of X^N and F^N is not too strong such an approach may be very efficient. For many packages based on sound

numerical analysis principles this situation is expected to be quite common. Again response trends and interaction will be important considerations.

Incidently, some of the considerations mentioned above in connection with splitting techniques are identical to some of those considered when decomposing the problem and control spaces of P into sets of problem and control classes.

A very important point to note in experimental programme design is that reference solutions should be included in the programme. For each X_α, there will be a number cases with varying $F_{\alpha\beta}$. This set of cases will be denoted by $S_\alpha = S_{\alpha\beta} | \beta = 1,\ldots,n_f(\alpha)\}$. All data cases with the same X_α form an equivalence class with respect to the mathematical model X_α is supposed to represent and so will occasionally be referred to as an X-class. Within each S_α there should be at least one case which may be designated as a reference case (See later for further detail).

3.6 KAS_2, $\tilde{S}^D$, $\hat{S}^D$ AND FB_{i2}

The objective of this step is to transform the system S^O, which may deal with a large number of variables, to a more manageable system $\hat{S}^D$, dealing with a smaller number of relevant features (transformed variables). The problem of selecting relevant features is similar to the problem of selecting relevant variables in modelling [17]. In modelling, and perhaps in knowledge acquisition, the relevant variables or features are often dimensionless and the techniques of dimensional analysis [17] are often useful for their selection. In knowledge acquisition, the most relevant features $\hat{Y}$ of the outputs are the reliability of solutions/outputs and the cost of computing solutions. Often the most relevant features of the inputs are the inputs themselves, i.e. $\hat{X} = X$. Similarly it will often be the case that $\hat{F} = F$. If features are based on variables which are distributions (i.e. linked to an index), then some form of summarising of their distributions (e.g. sampling or averaging) will be necessary. For example, if reliability is based on a displacement distribution, then the KAc might select the displacement at a particular point in space

as the relevant displacement on which the reliability is to be calculated. Thus while $S^D = (X^0, F^0, Y^0, I^0)$ for $\hat{S}^D$ we have $\hat{S}^D = (\hat{X}^D, \hat{F}^D, \hat{Y}^D)$, i.e. no $\hat{I}^D$.

In the CBKAS to be discussed in the next chapter, the process of selecting features is split into two stages, namely KAS_{21} and KAS_{22}. KAS_{21} deals with the transformation of S^D to $\tilde{S}^D = (\tilde{X}^D, \tilde{F}^D, \tilde{Y}^D, \tilde{I}^D)$. Often $\tilde{X}^D = X^D$, $\tilde{F}^D = F^D$, $\tilde{I}^D = I^D$ and the new output features are simply added to Y^D to form $\tilde{Y}^D$. Then, via KAS_{22}, the relevant features are extracted from $\tilde{S}^D$ to form $\hat{F}^D$. The reason for the two stage process is simply for data structuring and processing reasons. Symbolically the process can be depicted as below.

$$S^D \xrightarrow{KAS_{21}} \tilde{S}^D \xrightarrow{KAS_{22}} \hat{S}^D$$

FIGURE 3.6.1 : Detailed symbolic view of KAS_2. Note that FB_{32} (See Figure 3.1.3) has been omitted from, this view.

Feature selection is significantly aided by a priori knowledge of which aspects of the outputs are likely to be relevant and which are not. However, no matter how knowledgeable the KAc is, he can never know beforehand exactly which features will be relevant. After some study of the data (by using interactive graphics, etc.) and after some attempt at trying to find 'natural' 'partitions' (by pattern recognition, etc.), he will usually have a better idea of the relevancy of various possible features. The path FB_{32} in Figure 3.1.3 represents this process of knowledge feedback.

In addition to the selection of features, KAS_{21} also deals with the calculation of the features from the basic data S. For this, the KAc must also designate which data case will be used as a reference solution and select a formula for the calculation of the relevant feature. These two aspects will be dealt with in the following two sections, viz. sections 3.6.1 and 3.6.2. Section 3.6.2, it will be noticed, is called calculation of reliabilities. This is because solution reliability will be the most commonly calculated feature. Costs will usually be simple while the variety of other possible relevant output features cannot be foreseen.

3.6.1 Designation of Reference Cases

Solution features such as reliability require comparisons between solutions and a reference solution. If possible, this reference solution should be exact. For most problems which P was designed to solve, such an exact solution will not be available. In these situations one must use another numerical solution as the reference. It should be one which is <u>believed</u> to be accurate. Often the KAc may not know how to obtain such a solution at the beginning of the exercise. He may know how to set most processor variables F but some will need more consideration. In order to cater for such situations (which are expected to be quite common), an important design principle for the software tools to be discussed later, is one which allows the designation of reference cases to be changed and thus reliabilities to be recalculated. When used in a design evaluation, the search for such a solution may be the sole objective of the knowledge acquisition exercise.

Grounds for belief will usually follow from some sequence of trials where processor variables are changed according to trends in their effects, known to lead to greater reliability. For example, in FEA, higher mesh density usually yields greater accuracy so the KAc would try a sequence of experiments with increasing mesh density, compare the solutions and then decide which to accept as reliable. Such trends may however only apply to the 'normal' range of variable values and may change beyond such ranges. For example, as mesh density increases roundoff error will also increase and will eventually swamp the solution. There may be complex interactions with other parameters such as in the analysis of strain softening materials where softening parameters and mesh density are linked by fracture mechanics principles.

It may in some cases be undesirable to designate reference solutions by fixed values of F^N. Instead reference solutions may be parametrised by certain F^N variables. This typically will be the situation where problem variables, which theoretically should be in X^N, are put into F^N. Some material properties such as artificial viscosities or ALPHA in the example of section 1.5 are of this type.

3.6.2 Calculation of Reliabilities

This section presents a number of formulae used to calculate reliabilities or deviations of Y. It will be assumed that

a is the value (or vector of values) of the selected output variable (or variables) of the case for which the reliability is to be calculated,

b is the corresponding value or vector for the reference case (which is assumed to have been designated),

and c is a 'normalising' constant, calculated or supplied by the KAc.

A reliability or deviation measure will be represented by r(a,b) and d(a,b) will be a distance measure between a and b.

The distance measures are typically one of the following five:

$$d(a,b) = \sum_i^n w_i \, |a_i - b_i| \qquad (= d_1') \qquad \text{(D1)}$$

$$d(a,b) = \sum_i^n w_i \, |a_i - b_i| \qquad (= d_1) \qquad \text{(D1A)}$$

$$d(a,b) = \left[\sum_i^n w_i (a_i - b_i)^2\right]^{\frac{1}{2}} \qquad (= d_2) \qquad \text{(D2)}$$

$$d(a,b) = \max_i |a_i - b_i| \qquad (= d_\infty) \qquad \text{(DM)}$$

$$d(a,b) = \sum w_i \left|\frac{a_i - b_i}{b_i}\right| \qquad (= d_R) \qquad \text{(DR)}$$

where it is assumed that the weight w_i satisfy

$$\sum_{i}^{n} w_i = 1 \ , \ w_i \geq 0 \text{ for } i = 1,2, \ldots\ldots, n$$

a and b are assumed to be n-vectors.

A useful relation between the above distance formulae is, that for any a and b,

$$d_1' < |d_1'| \leq d_1 \leq d_2 \leq d_\infty$$

The most common choice of weights is $w_i = \frac{1}{n}$

For the 'normalising' constants typically

$$c = KAc \text{ supplied} \qquad (CO)$$

$$c = d(b,0) \qquad (CD)$$

Typical formulae for r(a,b) are

$$r(a,b) = \frac{d(a,b)}{c} \qquad (R1)$$

$$r(a,b) = \left|\frac{d(a,b)}{c}\right| \qquad (R2)$$

$$r(a,b) = \max\left(1 - \left|\frac{d(a,b)}{c}\right|, 0\right) \qquad (R3)$$

$$r(a,b) = \text{sign}\,(d_1')\,\frac{d(a,b)}{c} \qquad (R4)$$

The (R1, R2, R4) are deviation-type measures while (R3) is a reliability-type measure. However, for all forms, r(a,b) will be referred to as reliability. r(a,b) yields relative measures whenever C≠1 or (DR) is used as the distance measure.

Reliabilities calculated according to any combination of the above are point (PT) reliabilities. By this we mean that if a and b depend on a time-like index, the reliability will be calculated for every 'time' point of a. Values for b will be interpolated, if necessary. If quasi-random responses (see 3.7.1) are expected, such pointwise

reliabilities may require smoothing. Typically one might calculate an average (AVE) or root mean square (RMS) of the reliabilities with respect to 'time'.

When relative measures are used, normalising constants are usually chosen so that $r \in [-1, 1]$ or $r \in [0, 1]$ whichever is appropriate.

The reliability forms may be referred to by concatenation of the individual form identifiers, e.g. R1/D1/CO/PT. Some combinations are equivalent, e.g. R2/D2/CD/AVE = R1/D2/CD/AVE. If c is calculated via (CD), the d form used must be the same form as used to calculate d(a,b) in r(a,b).

Often it will be useful to calculate a number of reliabilities, usually at least one with the sign preserved and one without. Use of (D2) will usually give the smoothest response of r(a,b) with respect to X and F. Such considerations are often useful for finding the pattern in $\hat{S}$.

3.7 KAS_3 AND $\hat{S}^D_K$

This part of the knowledge acquisition procedure forms the core of the procedure. It concerns the 'partitioning' of $\hat{S}^R$

$$\hat{S}^R_K = \{\hat{S}^R_{k\ell} \mid k = 1,2,\ldots,K,\ \ell=1,2,\ldots,L(k)\} \quad \text{where}$$

$$\hat{S}_{k\ell} = (\hat{X}^R_k, \hat{F}^R_{k\ell}, \hat{Y}^R_{k\ell}).$$

This 'partition' $\hat{S}^R_K$ must be chosen to satisfy the goal conditions C1-8 (See section 3.3). As already mentioned the 'partitioning' is based partly on induction from a 'natural' 'partition' of the experimental data $\hat{S}$ and partly on a decision by the KAc, that is to say that the KAc looks for a 'natural' 'partition' $\hat{S}_K$ of $\hat{S}$ and, using $\hat{S}_K$ as a guide, decides on a 'partition' $\hat{S}^R_K$ of $\hat{S}^R$. Before his decision, experimentation is directed primarily towards enhancing the 'natural' 'partition' (bearing C1-8 in mind - hence 'natural' not natural) but after his decision, experimentation is directed primarily towards improving the 'empirical basis' of each rule. Goal conditions C2 and C4 involve measurements of the subsets $\hat{Y}^R_{k\ell}$ such as maximum, minimum,

mean and/or range of $\hat{Y}^R_{k\ell}$. These measurements are based solely on the experimental data $\hat{Y}_{K\ell}$. While the use of statistically designed experiments can certainly improve the expected accuracy of such measurements, heuristic methods will usually suffice. For example, if $\hat{Y}^N$ contains some quasi-random or nonlinear element, then the maximum of $\hat{Y}^R_{k\ell}$ may not be caused by any of the extreme values in $\hat{S}^R_k$ and $\hat{F}^R_{k\ell}$. However, heuristically it should provide an adequate measurement. Obviously the more data cases in $\hat{S}_{k\ell}$ and the better they are spread around $\hat{S}^R_{k\ell}$, the better the statistical evidence for $\hat{S}^R_{k\ell}$.

The whole process of this knowledge acquisition step is as shown in Figure 3.7.1.

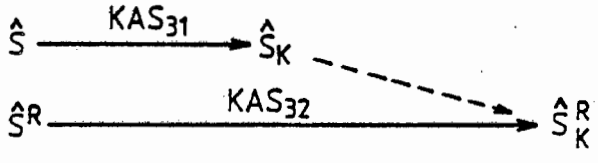


FIGURE 3.7.1 : 'Partitioning' $\hat{S}^D$ into $\hat{S}^D_K$ (copied from Figure 1.3.4)

The most difficult part of the above process is finding a 'natural' 'partition' of the data and designing additional experiments to enhance such a 'partition'. A 'natural' 'partition' manifests itself by patterns in $\hat{S}$, so the process is really one of pattern recognition and pattern enhancement. The rest of this section will deal with aids to such pattern recognition and enhancements.

Firstly, pattern recognition is highly dependent on what types of patterns are expected, i.e. it is contextual [9,32,33]. The expected patterns are recognised with the help of a priori knowledge such as expected relations between $\hat{F}$ and $\hat{Y}$ for fixed $\hat{X}_\alpha$ and expected patterns in $\hat{Y}$. Computer graphics is clearly important for any human directed pattern recognition. To increase the amount of data without further experimentation, interpolation may be important. Again this depends on the relations between the variables. Data sorting and clustering can be very useful aids. Filtering based on the Pareto principle is a key

element of the process but filtering (removal) may also be useful for simplifying the patterns. These topics are the subjects of the next seven subsections 3.7.1 to 3.7.7.

3.7.1 Pairwise Relations $\hat{y}_i^N - \hat{f}_j^N$ and $\hat{y}_i^N - \hat{x}_j^N$

While it is unrealistic to expect the KAc to have quantitative knowledge of the pairwise relations $\hat{y}_j^N - \hat{f}_j^N$ and $\hat{y}_j^N - \hat{x}_j^N$, knowledge of their qualitative aspects is a very important part of the a priori knowledge used to design experiments, recognise patterns and interpolate data. An expert user of P would be expected to know at least these qualitative aspects. If however they are not known, then their discovery becomes an important part of the knowledge acquisition problem - perhaps the most important part.

By a pairwise relation $\hat{y}_i^N - \hat{f}_j^N$, we mean the relation between $\hat{y}_i^N$ and $\hat{f}_j^N$ with the values of all other variables in $\hat{X}^N$ and $\hat{F}^N$ held constant. Pairwise relations $\hat{y}_i^N - \hat{x}_j^N$ are to be similarly understood. The purpose of this section is to describe and explain some terms which are useful for describing the pairwise relations $\hat{y}_i^N - \hat{f}_j^N$. These terms will be equally applicable to $\hat{y}_i^N - \hat{x}_j^N$ relations.

Firstly, the variables themselves may be continuous or discrete. For example, ALPHA, which is real, is continuous while NELTS, which is integral, is discrete. Relations between continuous variables are commonly described as <u>continuous</u>, <u>smooth</u>, <u>monotonic</u> or <u>stochastic</u> (<u>random</u>). However, many variables are expected to be discrete so the terms continuous and smooth are then inapplicable even though discrete relations (pairwise relations where at least one variable is discrete) are expected to exhibit behaviour similarly describable. Such discrete relations we will be described as having <u>continuous trends</u> or <u>smooth trends</u>. Monotonicity and randomness are applicable to both continuous and discrete relations. Continuity, smoothness, continuous trend, smooth trend and monotonicity are expected to be very common properties of relations. Randomness is highly unlikely because most engineering packages are based on deterministic principles (Chaotic relations [21] will not be considered in this thesis). However, many engineering packages contain iterative algorithms with tolerance-type

termination criteria. With such software the relations between the measured values of the variables may look partially 'random'. This type of behaviour will be called quasi-random.

Figure 3.7.1.1 illustrates some of the terms. It will usually be helpful to imagine discrete points connected by lines.

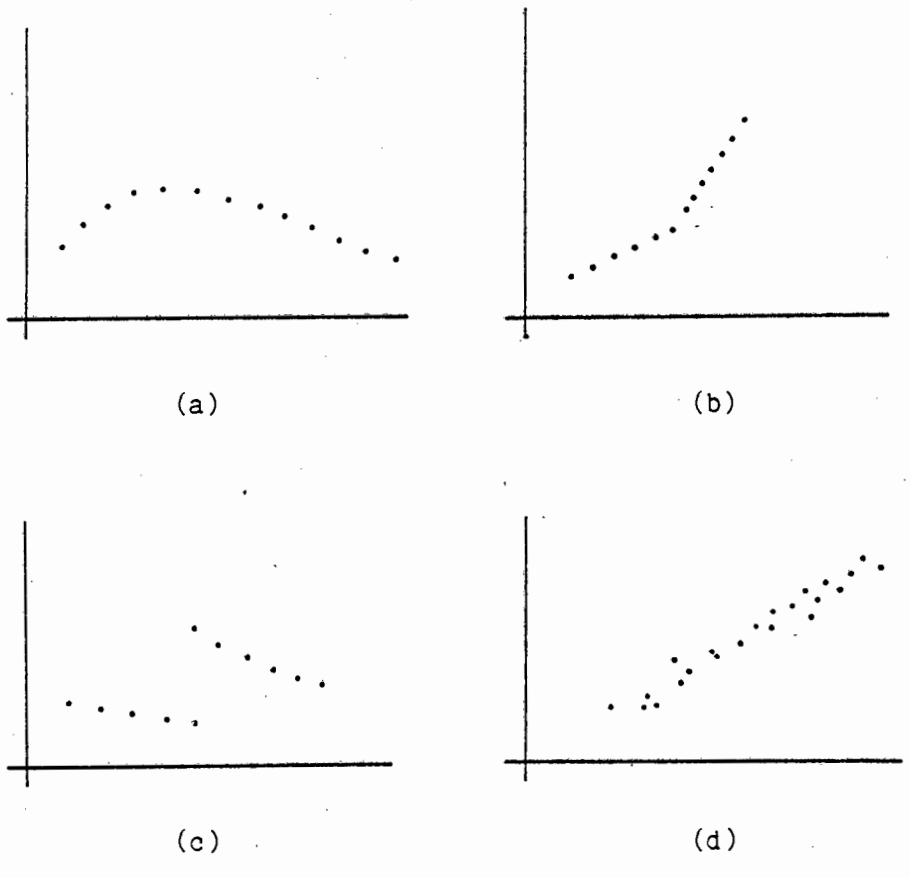


FIGURE 3.7.1.1 : (a) Continuous, smooth, non-monotonic trend
(b) Continuous, non-smooth, monotonic trend
(c) Discontinuous, non-monotonic trend
(d) Quasi-random with monotonic trend

It is important to note that different forms of reliability calculation may yield different relations. For example with an absolute value (unsigned) reliability measure, the relation may appear non-monotonic while if the sign is kept it will be seen to be monotonic. Usually monotonic, smooth relations will yield simpler patterns in the global (system level) relations.

Ideally the KAc should check that the behaviour exhibited by the data conforms to all the expected pairwise relations. If expected behaviours could be captured onto EDBn (the database containing $\hat{S}^D$) and they could be 'understood' by the CBKAS, then such checking should ideally be done by the CBKAS automatically (like a sophisticated form of database integrity constraint checking). Such an ideal is beyond the scope of the present work.

3.7.2 Patterns in $\hat{Y}$

Whereas the previous section dealt with pairwise relations, this section deals with relations between the variables in $\hat{Y}^N$ and relations between the whole (or subsets) of $\hat{F}^N$ and subsets of $\hat{Y}^N$. To include all possible such relations the word pattern will be used instead of relation.

Mostly patterns in $\hat{Y}$ concern costs and reliabilities. For a typical problem and control class, patterns like the one in Figure 3.7.2.1 result.

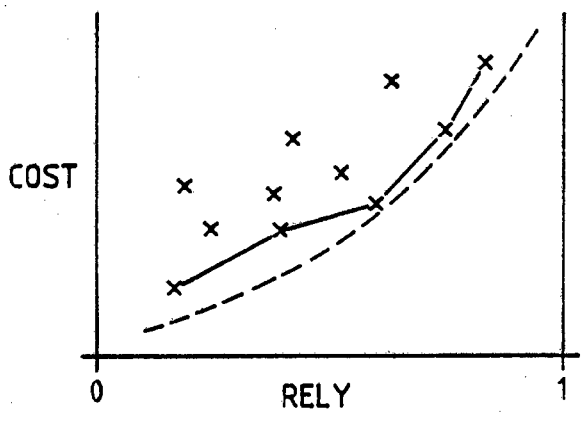


FIGURE 3.7.2.1 : A typical cost-reliability pattern

An interpretation of this figure is that

(a) on average, to increase reliability, one should expect to increase cost, and

(b) many solutions are inefficiently computed. One can hypothesize the existence of a curve of efficient solutions (Indicated by the broken line).

Points on this curve are Pareto optimal (non-inferior or non-dominated). The subset of $\hat{F}^R$ which causes $\hat{Y}$ to lie on this curve is the Pareto optimal set for the problem class and the goal is to find an approximation to this set. A filter which removes all inferior points would leave only those points connected by the solid line. Discussion of such filtering will be deferred until the next section.

Some typical $\hat{Y}$ patterns will now be illustrated and explained.

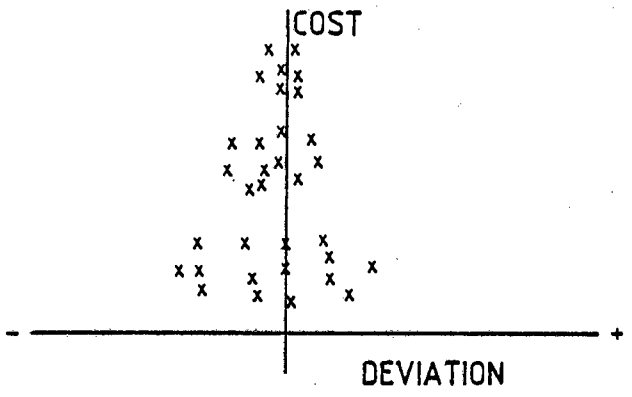


FIGURE 3.7.2.2 : A Reliability-cost pattern showing three clusters

In Figure 3.7.2.2, one sees three clusters. This pattern would probably be caused by a combination of discrete and continuous variables - the discrete variable causing the clustering while the continuous causing the variation within each cluster. The discrete variable dominates the cost dependence but perhaps not the reliability aspect. Note that the dependence is not monotone, a fact which is easily obscured if an unsigned reliability measure is used.

In Figure 3.7.2.3 (a) the broken line represents the most efficient solution for many given reliability level. However one can see that (r_2,c_2) dominates (r_1,c_1). Filtering out dominated points would result in the pattern shown in (b).

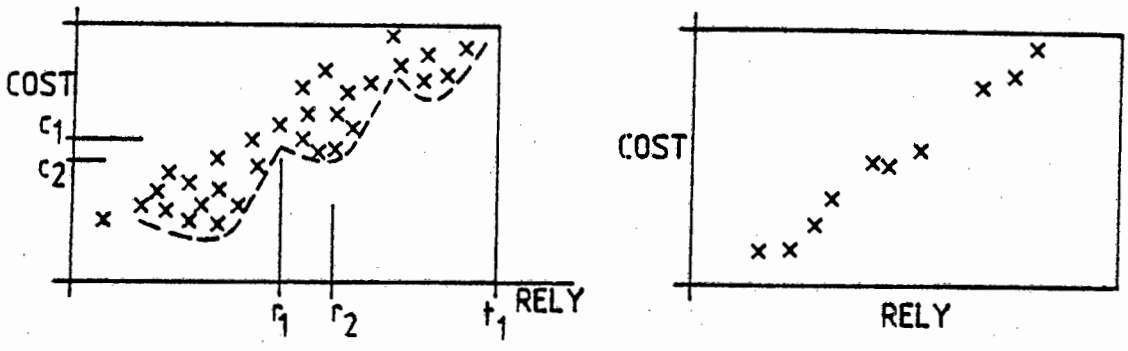


FIGURE 3.7.2.3 : Reliability-cost patterns without and with Pareto filtering

While both (a) and (b) indicate clustering, it is (perhaps) clearer in (b). Again, the above pattern would be caused by a dominant discrete variable with a superimposed scattering effect due to other variables. After filtering, the pattern shows the monotonic trend typical of a Pareto optimal set.

In some problem classes, 'tuning' variables typically produce the following pattern.

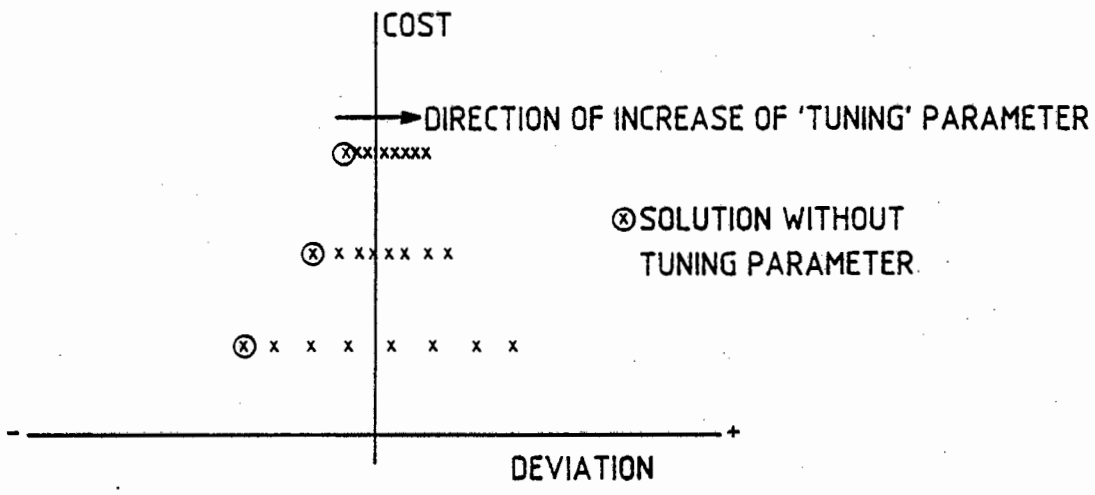


FIGURE 3.7.2.4 : Pattern caused by discrete variable plus continuous tuning variable

The discrete variable dominates the cost while its effect on reliability is monotonic convergent. However, a tuning parameter can improve reliability significantly without affecting the cost. Due to the monotonic convergence property of the dominant variable, the influence is reduced (scatter becomes smaller) as cost increases. Also

very important in this situation is the use of a signed reliability measure. An unsigned one would simply confound the pattern. Instead of Pareto filtering in the above situation, one would probably simply interpolate the appropriate value for the tuning variable for each cost level. However while tuning variables improve one measure of reliability, they often worsen another measure and a trade-off is needed so Pareto filtering may again be needed.

3.7.3 The Importance of Graphics

Very many of the judgements required in knowledge acquisition depend on pattern recognition on numerical data. Often the point of view from which the data is observed critically affects the possibility of recognizing patterns. At present, machine pattern recognition is reliable only for special purposes, not for such general purposes as in this thesis. Often they require special hardware. Humans, however, are excellent at most forms of pattern recognition and so, to capitalise on this ability, it is essential that any CBKAS have good interactive graphics facilities and facilities for easily changing points of view.

3.7.4 Interpolation of Data

Almost invariably the initial programme of experiments will yield too little data for patterns to be recognised. The most obvious way to overcome this is by further experimentation. However, if each experiment is expensive, the cost of knowledge acquisition may become excessive. An alternative is to interpolate the current data. It will usually be very much cheaper to interpolate than to experiment. This may at first sight appear invalid since, at best, interpolated data is of the same quality as the original while, at worst, it is meaningless when quasi-random relations are involved. However, if the relations are relatively smooth and/or any quasi-randomness is relatively insignificant, it can be very useful to interpolate. This is because pattern recognition is often better on more dense data. Of course, too much data may obscure patterns - but then one can use filtering techniques. It is often the case that pattern recognition on a large data sample, filtered and summarised down to the same amount as some

small sample will be easier than in the former case. It is like looking at a detailed picture from a distance rather than a crude picture from close up. A more technical analogy is the use of the following heuristic in FEA of structures : static condensation from m>n degrees of freedom down to n yields better dynamic characterisation than a model with only n degrees of freedom. Once more real data has been generated, the process may be repeated with interpolation always from real data. Even if a formula for the relation exists, visualisation of its behaviour via graphical display of points generated by the formula usually improves understanding. If the formula is complicated such visualisation may be extremely useful for finding simpler approximations. This is exactly the situation in knowledge acquisition : P is like a complex formula which the KAc and users of P wish to understand via simple approximations (HK) to its behaviour.

Response surface methodology [13] is a set of techniques for approximate modelling of complex input-output processes. Typically, a response y_i might be assumed to be linearly dependent on a set of inputs x_j, $j = 1, m$. From statistical analysis appropriate experiments are designed which would allow such response surface fitting to be done. Once response surfaces $y_i = y_i(x_j)$, $j = 1, m$ have been fitted by a least square error method they may be used for approximate optimisation purposes. As mentioned already, experimental programmes are designed with possible response surfaces borne in mind. In general, the actual fitting of response surfaces is not recommended. Firstly, there will usually be no intention of using the surfaces' equations to formulate into rules. Secondly, to reduce biasing error, surfaces of sufficiently high order need to be fitted. This added complexity may not be warranted for the generation of heuristic knowledge and local interpolations and subsequent experimentation will usually be more appropriate.

3.7.5 Sorting

After studying the patterns in $\hat{Y}$ alone and finding some approximation to the curve of efficient solutions, the next step is to find the cause of the patterns, i.e. to find a pattern in $\hat{F}$ X $\hat{Y}$. It is here that sorting, clustering and filtering have their impact. Clustering

and sorting arrange the data into groups so that first intragroup and then intergroup study can be done. Filtering reduces the search to subsets of $\hat{F}$ and $\hat{Y}$.

Sorting is obvious if only one input or control variable is involved. However, if more than one are involved, there is the problem of selecting the order of the variables on which sorting is to be done. For example, if F^N contains (f_1^N, f_2^N), does one sort on f_1^N then f_2^N or on f_2^N then f_1^N? As the number of variables increases, the number of possible sorting orders increases combinatorially. In fact finding an appropriate sorting order may be part of the goal of knowledge acquisition. The order should go from greatest to least dominance (like the ranking of goals in goal programming [18,19] for polyoptimisation). In this way, the dependence of the $\hat{Y}$ patterns on corresponding $\hat{F}$ patterns should be clearer. Again, however, dominance may be dependent on the particular features involved in the $\hat{Y}$ patterns - changing a reliability measure may change the dominance order. For situations like Figure 3.7.2.4, sorting may well be the most important tool required when the data has not been captured in a fixed order.

3.7.6 Clustering

Sorting produces a hierarchy of groups in a data set. Clustering (Appendix A) is also a grouping technique but does not necessarily produce a hierarchy of groups. The idea is to group together data cases which, according to some criteria (clustering criteria), are similar. Typically clustering criteria may be based on cost, reliability, both cost and reliability or, more generally, combinations of $\hat{Y}$ variables. Similarity measures need to be defined. Very common measures are ones based on Euclidean-type norms and are therefore most suited to clusters having the shape of balls in R^n (See Figure 3.7.2.2). Whereas in sorting a complete order of variables needs to be specified, in clustering only a subset of variables (typically only a subset of $\hat{Y}$) is involved in the clustering criteria. After clustering, the members of each cluster are commonly sorted into some order for presentation. Selection of clustering criteria and of the number of clusters are frequently encountered problems with clustering. Clustering can be used to find patterns (a set of clusters), in the $\hat{Y}$ data, it can be used to 'partition' $\hat{Y}$ even if no

obvious pattern exists or, when the pattern is clearly visible to the KAc, it can be used to manage the $\hat{Y}$ data for presentation and subsequent $\hat{F}$ - $\hat{Y}$ pattern recognition.

3.7.7 Filtering

By filtering here, is meant an algorithm which removes most of the inefficiently generated solutions from consideration. Note that most not all will usually be removed. By removing all inefficient points, one might end up with a very sparse data set in which no pattern can be found. Usually it will be better to relax the filtering so that only most are removed. A type of approximate filter results which includes a parameter specifying the level of approximation. Such a filter will be called an approximate Pareto filter.

Recalling the description of the Pareto principle given in section 3.2, we may define the Pareto optimal set U of $\hat{Y}$ as follows. If $u \in \hat{Y}$ is an n-vector, then if there is no $v \in \hat{Y}$ such that

$v_i \geq u_i$ for all i=1,2,...,n with the inequality holding for at least one i,

then $u \in U$. Then an approximate Pareto optimal set can be defined as the set U_ε with the following properties:

(1) $U \subseteq U_\varepsilon$

and (2) for all $u' \in U_\varepsilon$, $\|u' - u\| < \varepsilon$ for some $u \in U$.

$\|.\|$ is some appropriate norm and ε is the filter approximation parameter. A typical norm would be

$$\|u' - u\| = \sqrt{(u' - u)^T W (u' - u)}$$

where W is a diagonal scaling matrix. The subscript ε on U_ε is meant to denote the fact that the contents of U_ε depends on a filter parameter ε. In addition to the above conditions one may have threshold levels for $\hat{Y}$ so that solutions of very low reliability or very high cost are excluded from U_ε. As stated above, the Pareto

principle is only relevant to situations where the goal is to maximise u. In general one wants to maximise some components, minimise others and maximise or minimise the absolute values of still others. Modifications for such purposes are relatively straightforward.

The set U_ε forms the set of positive examples while the rest $V_\varepsilon = \hat{Y} - U_\varepsilon$ form the set of negative examples. The advantage of the filter parameter ε is that the boundary between the positive and negative examples can be moved. A very effective way to find patterns in $\hat{Y}$ and $\hat{F}$ X $\hat{Y}$ is to form an ε-sequence of sets U_ε, V_ε, perform clustering, sorting, etc. on those sets and then compare their contents. In this way the KAc can learn how to recognise subsets of $\hat{F}^R$ which contain the Pareto set and so can design further experimental programmes (FB_{31}) to focus the search in these subsets. Recall that goal condition C7 concerns the simplicity of the desired sets (so that the resulting HK^{ij} will be quite simple). However, after approximate Pareto filtering the set U_ε and associated $\hat{F}$ and $\hat{X}$ subsets will usually still be quite complex (i.e. not convex, not simply connected). The process of removing members for simplification will be called simplication filtering (or independent filtering in ELIXIR).

3.8 KAS_4 AND HK^{ij}

In the statement of the goal in section 3.3 an implicit form for the HK^{ij} was assumed. By seeking 'partitions' in $\hat{S}^R$, we are implicitly saying that the HK should be represented in the following <u>rule</u> form:

$$\text{IF} \quad \hat{X}_\alpha \in \hat{X}^R_k \quad \text{AND} \quad \hat{F}_{\alpha\beta} \in \hat{F}^R_{k\ell} \quad \text{THEN} \quad \hat{Y}_{\alpha\beta} \in \hat{Y}^R_{k\ell}$$

All this does is present $\hat{S}^R_K$ in more readable form. It does not however place the HK^{ij} in its original context. To do this we have to add the derivation of $\hat{S}^N$ from S^N and the interpretation of the S^N via $\mathcal{S}^N$. Along with this latter interpretation is the definition of correspondences β^R between S^R and $\mathcal{S}^R$. Thus the final form of the HK^{ij} as produced by the execution phase will be a combination of

- rules using S^0, $\mathcal{S}^0$ and $\hat{S}^D_K$
- statements and formulae describing transformations, e.g. how reliabilities were calculated,

- statements, diagrams, graphs, etc. characterising the reference solutions precisely,
- statements, diagrams, graphs, etc. which are useful for defining the problem and control class and the set of rules,
- an explanation of the structure of $\hat{S}^D_K$. Explanation will be discussed in the integration phase (Section 3.10).

3.9 FEEDBACK

Knowledge acquisition has been shown to be a iterative process. As such, feedback paths are an integral feature. Most of these feedback paths the $FB_{k\ell}$ shown in Figure 3.1.3, have already been dealt with. This section will merely summarise and collect together the main feedback ideas. Since there are feedback paths to all parts of the procedure, it will serve simultaneously to summarise the chapter. The order of presentation will be roughly the expected order that the KAc will follow.

FB_{2i} - Feedback from KAS_2 to KAS_1 and KAS_0

$\hat{S}^D$ is supposed to be an abstracted form of the data in S^D. The abstraction consists of data transformations and subset selection. In order to find the appropriate transformations and selection, the KAc will study S^D with the aid of KAS_2. During this study it may become evident that:

(i) the experimental programme is too small,
(ii) more variables need to be defined and their values recorded or
(iii) some variables may be disregarded and their values no longer collected in future experiments.

The objective of including FB_{21} is to allow for correction (i), i.e. to ensure sufficiency of data. The objective of including FB_{20} is to allow for corrections (ii) and (iii).

FB_{3i} - Feedback from KAS_3 to KAS_2, KAS_1, KAS_0

FB_{32} deals with

(i) Redefinition of $\hat{S}^D$ from $\tilde{S}^D, S^D$:

(a) Recalculation of transformed quantities, e.g. changing reference cases or changing the components of the variable used in the transformation.

(b) Redefining the transformation, e.g. new formulae, different variables or any of those in (a) above.

(c) Adding new transformed variables.

(d) Changing the subset selection of $\hat{S}^D$ from $\tilde{S}^D$.

(e) Changing the arrangement of data.

(ii) Modifying $\hat{S}^D$, e.g. by summarising $\hat{Y}$ as in operations b and c of section 3.7.8.

FB_{31}, like FB_{21}, is concerned with sufficiency of data.

FB_{30}, like FB_{20}, is concerned with expanding, contracting or modifying the spaces of variables defined in S^0. However, it is also concerned with redefinition or modification of S^0 and β, i.e. with the whole of $\sum^0$. Attributes may be dropped and new ones included. β would need adjustment accordingly. It is via FB_{30} that β^R is completed. For example the grouping of SAR in section 3.3 is derived during knowledge acquisition and only after this can the groups be connected with low, medium and high structural aspect ratio. If the KAc had found four 'natural' groupings of SAR, β^R would be a little different.

3.10 THE INTEGRATION PHASE

Recall that the integration phase was depicted symbolically as shown below.

(i) For each i:

$$\bigcup_j HK^{ij} \xrightarrow{KAc} HK^i$$

(ii)

$$\bigcup_i HK \xrightarrow{KAc} HK$$

FIGURE 3.10.1 : Symbolic view of the integration phase

The purpose of knowledge integration is to form $U_j\ HK^{ij}$ and $U_i\ HK^i$ into a compressed body of heuristic knowledge for P as a whole. While the acquisition of HK is done under the assumption that the HK will be used, this fact should not be taken for granted. However, if the HK is comprehensible, i.e. consistent with physical and computational theory, self-consistent, organized and simple, it should inspire confidence in the user of P that the use of the HK will simplify his numerical modelling tasks. In order to make the HK comprehensible to users, he should first make it comprehensible to himself.

Before describing when and how integration is done, we first need to elaborate on the features of comprehensibility mentioned above, namely consistency with basic theory, self-consistency, organization and simplicity.

Firstly, consistency with basic theory means that for each HK^{ij} the KAc can find an explanation of the structure of the HK^{ij} in terms of the physics of the problem class and the computational principles upon which P was designed relevant to the control class. We believe that the most common reason for a lack of adequate explanation will be due to errors in knowledge acquisition such as too little data (real data), improper data processing, invalid S^0, data collection error and so on. Thus consistency with basic theory is a check on the validity

of the individual HK^{ij}. If, however, no explanation can be found after carefully checking the acquisition process, the KAc may simply treat the HK^{ij} as empirical, in which case only further experimentation and knowledge acquisition by independent KAc's can guarantee the validity of the HK (as far as any empirical knowledge may be guaranteed) or he may treat it as a genuine discovery requiring detailed research and/or modification/development of P. By a genuine discovery we mean an observed phenomena that has so far not been publicized. Such 'inexplicable' phenomena may be exhibited by nonlinear mathematical and/or numerical models which have not been studied in great detail. Where no explanation is found for any HK^{ij}, the user should be warned accordingly. It should be remembered that the purpose of knowledge acquisition is not usually detailed research and explanation but to provide HK that is (intuitively) comprehensible to the user - not necessarily proved to the user.

Secondly, self-consistency concerns consistency between say HK^{i} and $HK^{k\ell}$. If PC^{i},CC^{ij} and $PC^{k},CC^{k\ell}$ involve similar mathematical and computational theory (but are, presumably, different specialisations), the structure and explanation of HK^{i} and HK^{k} should be similar (perhaps identical). If they are not similar it again indicates error in knowledge acquisition or perhaps PC^{i},CC^{ij} and $PC^{k},CC^{k\ell}$ are not as similar mathematically and computationally as expected. An explanation for such a difference is then required.

Thirdly, organization of the HK relates to the structure of the whole HK (not just the individual HK^{ij}) and its presentation. The structure of the HK should follow approximately the structure of the KAProg but whereas in designing the KAProg foresight was required, organizing the HK is done with the advantage of hindsight. Only very general statements about such organizing can be made here since each situation will probably require special organization. Obviously, the various $PC^{i}/HK^{i}/CC^{ij}/HK^{ij}$ will be similar in some respects but differ in others. The trick is to find a structure which reflects both similarities and differences in a comprehensible way. Hierarchical structures are most easily inderstood but unfortunately almost all engineering packages will result in network structures. This is because the various options (physical, geometrical, mathematical, etc.) may be combined in some problem classes. For example, a set of problem classes for a structural analysis package might deal with

beams alone, plates alone and beam/plate combinations. Further complexity arises if each of these may also have any combination of static or dynamic, linear or nonlinear analysis options. Does the KAc group the classes according to their geometrical features or analysis types? (cf sorting and clustering criteria). One reasonable way would be to have three groups of classes, the first dealing with beams alone, the second with plates alone and the third with beam/plate combinations. Then each group begins with a class dealing with the simplest analysis options, e.g. linear static analysis. Successive classes should increase in difficulty of analysis or behavioural complexity. Such an organization is based on first grouping according to geometrical features and then according analysis types. To aid users interested in problems concerning say nonlinear materials, an index pointing to all such probable classes can be provided. In general, a number of indices may be required. The problem becomes very similar to one of listing books according to certain keywords and perhaps the data structures used in their computerisation may be appropriate to a computerised presentation of the HK (for example, in an expert system).

Finally, the simplicity of the HK greatly affects its comprehension. It was partly for reasons of simplicity that a rule form of representation was chosen. Here we wish only to emphasize its importance with regard to the HK as a whole. We expect that many of the structures of the individual HK^{ij} and their explanations will be similar. Where such similarities exist it may be possible to simplify the HK by amalgamating problem classes into fewer but slightly more complex problem classes. For example, the KAc might have thought that for a certain beam problem, material and geometric nonlinear effects would cause different reliability and cost behaviour and so separate problem classes for each effect were chosen. However, if it turns out that the HK^{ij} is very similar, he may subsequently amalgamate the two. He may of course need to repeat the knowledge acquisition. Simplicity also affects the nature of explanations : if simple intuitive explanations are possible, they are to be preferred to complex proofs (which can simply be referenced). Most explanations of the HK we expect will therefore be intuitive.

We now return to when and how integration is to be done. Figure 3.10.1 gives the impression that integration is done only after all HK^{ij} has been acquired. This approach does have certain advantages in that the execution phase can easily be divided amongst a number of KAc's and then the integration done all at once. By looking at U_j HK^{ij} and U_i HK^i all at once, the KAc (and his team) may find many simplifying patterns in the behaviour of P and so produce a relatively simple body of organised heuristic knowledge about P. Alternatively, the KAc might attempt to integrate the HK^i and HK^{ij} incrementally, i.e. as each HK^{ij} is acquired it is integrated into the current body of HK. This incremental approach also has advantages. Firstly validity checking is done just after execution (perhaps even during execution) so that errors found are more easily corrected because the overheads (psychological as well as computational) involved in restarting an execution are avoided. Secondly, and most importantly, the KAc is continually increasing his knowledge and understanding of P's behaviour and so can bring it to bear on knowledge acquisition of subsequently studied problem classes. In other words, the a priori knowledge that he brings into the execution phase is continually increasing. This should improve both the efficiency with which the execution phase is done and the quality of the HK^{ij} produced. Incremental integration may itself be easier once a few HK^{ij} have been integrated. We expect that the most suitable approach would combine the above, i.e. would involve studying a group of similar problem and control classes, integrating the HK^{ij} for the whole group at once and repeating this process until all groups of problem classes have been executed and their HK^{ij} integrated. Thirdly the incremental approach allows redesign and/or modification of the KAProg during the knowledge acquisition process. Because it is impossible to predict accurately the behaviour of P during the initial planning phase, it may be essential to repeat the planning phase as more a priori knowledge becomes available. This is once more a form of knowledge refinement via feedback.

4 DESIGN AND IMPLEMENTATION OF A COMPUTER BASED-KNOWLEDGE ACQUISITION SYSTEM

The preceding chapters may be viewed as an analysis of requirements for conceptual design of a knowledge acquisition system. Except for fairly simple studies, the acquisition process will evidently require considerable data handling and processing. In order to make the system viable it must be computerised. By this it is not intended to automate the whole process, but, realising that human involvement will in most cases be essential, the aim is to develop a computer based system which will aid the knowledge acquirer.

The chapter consists of three parts. The first part deals with the user's view of a CBKAS. The user in this case is the KAc. Instead of presenting and explaining the concepts involved in a CBKAS in a general form, we will illustrate these concepts by presenting them as implemented in ELIXIR. The second part looks behind the user's view to the developer's view of a CBKAS. Again ELIXIR is used. Some issues concerning data, program and control structures will be discussed. The final section deals with software engineering issues. In developing ELIXIR it became very clear that a CBKAS is itself a complex and sophisticated piece of software. Some software engineering principles that proved useful plus some that we believe would have been (and will be) useful are discussed.

4.1 USER'S VIEW OF A CBKAS

A computerised database (DB) is a file or set of files which holds all data related to a particular application. Setting up and manipulating an application database is done by the user from a user device (UD) via a database management system (DBMS) [22-25]. Application software (AS) for special data processing also makes use of the DBMS. Schematically this situation can be represented as follows.

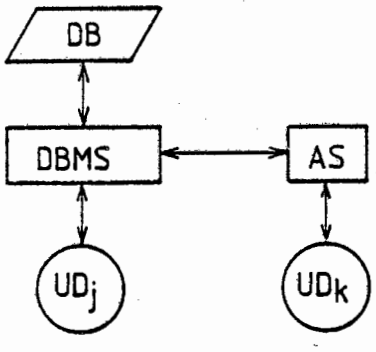


FIGURE 4.1.1 : An Abstract User's View of an Application System making use of a Database Management System

A CBKAS can be thought of as an application system and a DBMS rolled into one. But, whereas a general DBMS would allow the user to tailor his database structure (schema) appropriately, the appropriate structures have already been built into the CBKAS. The user of the CBKAS is of course the KAc. In the CBKAS to be discussed in this thesis, namely ELIXIR, more than one database is usually involved in a particular knowledge acquisition exercise. These are EDB and EDBn. The ELIXIR system can be represented as follows:

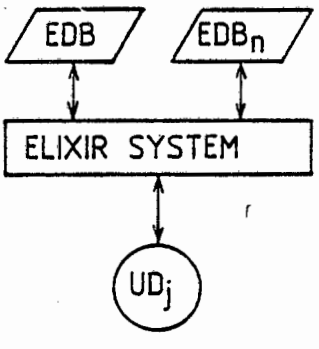
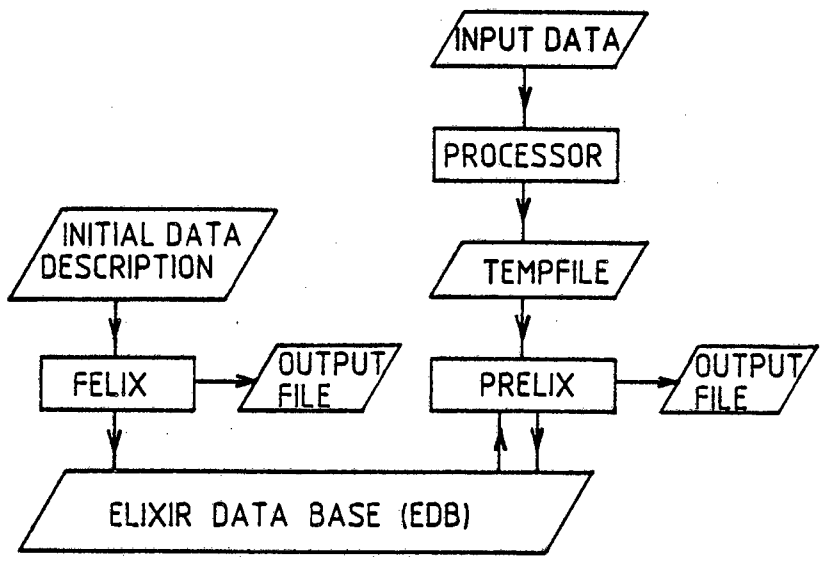


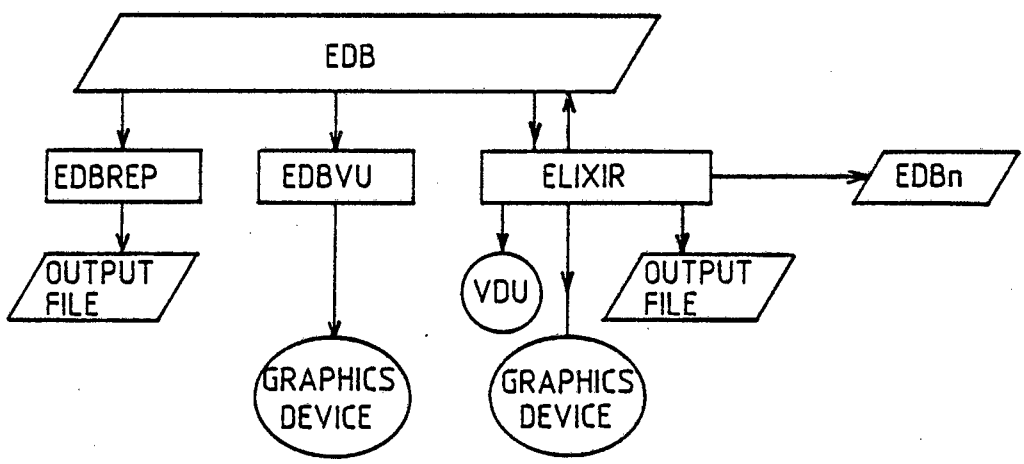
FIGURE 4.1.2 : Abstract User's View of the ELIXIR CBKAS

EDB, the ELIXIR database will contain all information and data concerning $\sum^{O}$, S^{D} and $\bar{S}^{D}$. The database subset EDBn will contain $\hat{S}^{D}$ and perhaps $\hat{S}^{D}_{K}$.

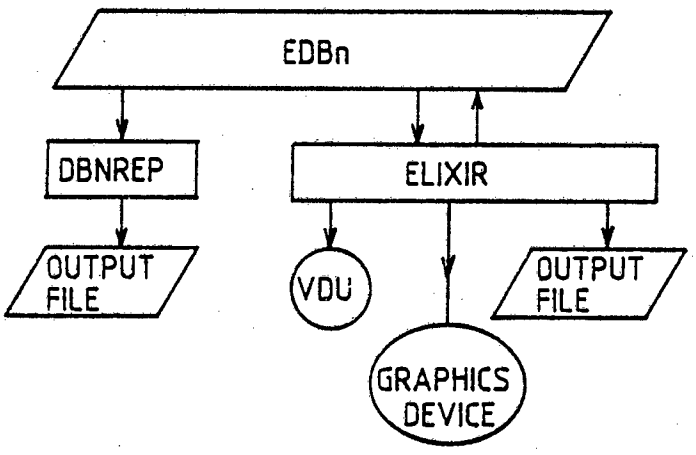
The part of a DBMS which is used for setting up an application database is usually called the database definition language (DDL) while the part used for manipulating the database is usually called the database manipulation language (DML). In the ELIXIR system, the program FELIX is used for setting up the database EDB while all the other programs (PRELIX, EDBREP, EDBVU, ELIXIR, DBNREP) mainly involve data manipulation. However, additional variables like reliabilities are defined during knowledge acquisition so FELIX is used only for the initial data description and the databases EDBn are set up from within the program module ELIXIR and not via FELIX. The whole ELIXIR system can be viewed as shown in Figure 4.1.3.



(a) Initial data description and automatic data capturing



(b) Data base reporting, graphical viewing, data transformation and formation of transformed databases



(c) Reporting of transformed databases and pattern recognition

FIGURE 4.1.3 : Detailed User's View of the ELIXIR KAS

The explanation of this figure is as follows. The KAc sets up a file containing an initial data description. An example of such a file is shown on page 105 in Chapter 5. FELIX reads this file and sets up EDB. EDB thus contains $\sum^0$. The process corresponds to KAS_0. EDB will also contain some details concerning automatic data capturing needed by PRELIX. Perhaps the single most important design criterion of ELIXIR was that it should be applicable to the study of fairly general processors P. This hinges on the initial data description and the file structure of TEMPFILE. The target field of application of ELIXIR is engineering software. A study of the data types and relations and possible output file structures was done on finite element packages, finite difference packages and other semi-analytical engineering software. The form of the initial data description and TEMPFILE structures was then designed to cater for all these packages. It is the adoption of the DBMS approach to the ELIXIR design which is largely responsible for the applicability of ELIXIR to general P.

The next step is to apply KAS_1. This is done by preparing data for the each case in the experimental programme, running this through the processor P and collecting the output plus relevant input data on file TEMPFILE. TEMPFILE is a sequential file, the contents and structure of which depend on the data description of $\sum^0$. PRELIX uses this data description on EDB to interpret TEMPFILE and then stores the data in structured form on EDB. Thus S^D is formed. At this stage EDB contains

Σ^O and S^D. The output files produced by FELIX and PRELIX contain information which helps the KAc to verify that the data description and data captured are correct.

Further detail concerning FELIX, PRELIX and their associated files will not be given here, but some detail should emerge from the descriptions of the use of ELIXIR given in Chapter 5.

EDBREP produces a print file of the contents of EDB while EDBVU is used for selective interactive graphical viewing of its contents. Future versions of EDBREP will allow selections of the contents to be printed while future versions of EDBVU will allow printing of the graphically presented data.

The primary menu in the ELIXIR program is the following:

1. PROCESS/DISPLAY PRIMARY DATABASE EDB
2. PROCESS/DISPLAY DATABASE SUBSET EDBN
3. HELP
4. END ?

The first option puts ELIXIR into KAS_2 mode while the second puts it into KAS_3 mode. In KAS_2 mode, the KAc may select output variables to be used in reliability calculations, designate reference solutions, select an appropriate formula for reliability calculation, select a subset of cases for which reliabilities are to be calculated (typically a single X-class, i.e. all cases with the same X_α) and then calculate the reliabilities for this subset (X-class). All these operations are part of KAS_{21}. Use of EDBREP and EDBVU also constitute operating the CBKAS in KAS_{21} mode. The product of this effort is $\tilde{S}^D$ which is also stored in EDB. The next step is to select a subset of $\tilde{S}^D$ and rearrange it into $\hat{S}^D$. This implies operating ELIXIR in KAS_{22} mode. $\tilde{S}^D$ is stored in a new database EDBn created by ELIXIR. By taking different subsets, the KAc may create EDB1, EDB2,..., EDBn. Each of these is a data system $\hat{S}^D$. There is no unique $\hat{S}^D$ for any G^{ij}. It is the KAc's task to find a $\hat{S}^D$ appropriate to satisfying G^{ij}.

In KAS_3 mode the KAc uses the graphical, filtering, sorting, clustering, reporting and set operation features of ELIXIR to help him find 'natural' patterns in $\hat{S}$ (on EDBn) so that it may be 'partitioned' into $\hat{S}_K$. From $\hat{S}_K$ the KAc may then induce $\hat{S}_K^R$. Using DBNREP to print the contents of EDBn will usually be helpful.

The examples in Chapter 5 demonstrate the usefulness of ELIXIR for knowledge acquisition.

At present the contents of the databases are accessible to the KAc only via the ELIXIR system and he has very little control over the format of presentation. In future versions the system will be made more open by providing more database utilities for directly accessing their contents.

4.2 THE DEVELOPER'S CONCEPTUAL VIEW OF A CBKAS

In order to design a software system, the software engineer needs to know what types, volumes, structures and variability of data will be required and which operations, sequences of operations and variations of these will be performed on the data. In addition he needs to know whether additional data types, data structures and operations are likely to be required in the future so that he may plan for the evolution of the system. For designing a CBKAS, the above required information is essentially provided by the first three chapters of the thesis.

From the first three chapters we find that character, integer and real data are all required. Each variable might be scalar, vector or matrix. Output variables may be linked to index variables. The dimensions of these vectors or matrices may be large and vary from (data) case to case. Even the presence of data for a variable may vary from case to case if the KAc decides not to continue collecting data for a specific variable or realises that he should be collecting data for another. Data structures for dealing with sets (of cases), clusters and 'partitions' are required. These data structures may be dependent on the types of patterns found in the data but we cannot say

beforehand what all the pattern types are likely to be. Thus new data structures will be needed. In short, what is required is a system which can deal with flexible data structures and types.

Chapter three lists some important operations which may be performed on the data as well as likely sequences of such operations. It is clear that the sequences of operation may vary from study to study and that many more operations (e.g. multi-variate statistical techniques) may be required. This implies that the 'library' of operations in the CBKAS will always be incomplete. One way to increase the library without adding extra computer code would be to allow the KAc to extract data from the databases, format it as desired, process it on some other package (e.g. SAS) and insert the results into the database without destroying its integrity. The program or system structure and its control should therefore be flexible.

In response to very similar problems AI researchers [1,3-8] invented production rules, semantic nets, frames, etc. to provide flexible data structures. They also separated the data, the operations and the control aspects so that flexible program structures could be achieved. Special languages like PROLOG, LISP, SMALLTALK, etc. were invented to ease the coding of such flexible structures. Data management and control aspects are supposed to be handled by the language compilers, interpreters and meta-interpreters. Database researchers [22-24,3] responded to similar problems by building up complex flexible data structures from sets of simple relations (such as in relational databases) and providing for flexible operation and control by separating the data, data processing, data management and control. These were achieved by developing database management systems (DBMSs). Data is held separate from programs in a database, data processing is done by application and utility programs, data management is handled by the DBMS and control over the processing is done by the user or his application programs. Unfortunately neither AI languages nor DBMSs currently support vector and matrix data types (FEA software developers who used database techniques had to write their own DBMSs). The concepts invented by the AI and DBMS researchers are, however, excellent for CBKAS design.

Two more ideas should also be mentioned before describing an 'ideal' developer's view of a CBKAS. Firstly data management can be separated into database management and memory management. Modern DBMSs already incorporate this idea, at least to some degree. Secondly, pattern recognition requires sophisticated user interfaces and the current trend is to introduce a user interface management system (UIMS) [52]. Traditional languages like FORTRAN were designed with printers and punch cards in mind whereas interactive computing requires windowing, graphics and multitasking facilities.

Figure 4.2.1 shows how we envisage a CBKAS based on the above ideas.

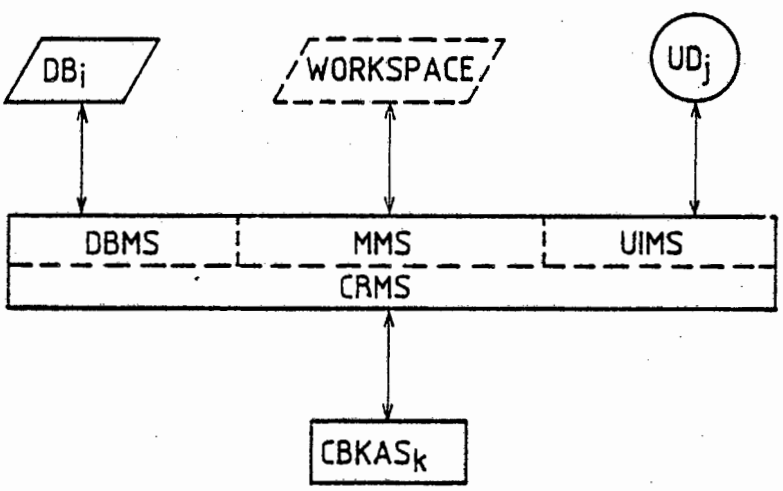


FIGURE 4.2.1 : A Developer's Conceptual View of an Ideal CBKAS

The CRMS shown in this figure is what we will call a computer resource management system. It consists of a DBMS for managing the databases (DB_i), a memory management system (MMS) for managing the workspace set aside in memory for the application and a UIMS for managing the user devices (UD_j). Ideally the CRMS is like a very high level language tailored to CBKAS requirements. Each module of the CBKAS ($CBKAS_k$) could then be written in this high level language. The main differences between the CRMS language and commercial DBMS languages are the relative separation of the DBMS, MMS and UIMS functions in the CRMS and the fact that the CRMS would be tailored to knowledge acquisition applications. Ideally the CRMS language should be able to

call subprograms written in other languages and/or CRMS procedures should be callable from these other languages. By the workspace we mean that part of memory set aside for holding data. However, the program code may itself use a large amount of memory so perhaps memory management should also deal with program use of memory. Even with the larger and larger memories of modern systems, memory management will be required because the problems tackled get larger and more complex. These ideas are discussed in more detail in [61-68].

The ELIXIR system achieves these ideals to a limited extent. The main modules FELIX, PRELIX, EDBREP, ELIXIR, EDBVU and DBNREP are not written in a very high level language (FORTRAN was used) but they do consist primarily of calling subroutines from a CRMS library of subroutines written for knowledge acquisition purposes. To some extent separate DBMS, MMS and UIMS routines could be written but many CRMS routines have elements of more than one of these functions. We believe however that with the experience gained from developing ELIXIR and perhaps using a more sophisticated language than FORTRAN, the degree of separation of function can be increased.

Most of the ideas presented in this section can be summarised under the label of modularisation. By this we mean the development of many functionally separate modules which can be assembled into complex systems like a CBKAS. It allows such a CBKAS to deal with quite general P by separating data definition from the CBKAS programs and eases maintenance and development by localising errors (bugs) and allowing modules to be replaced.

4.3 SOFTWARE ENGINEERING FOR A CBKAS

Reference [53] defines software engineering as: "The establishment and use of sound engineering principles (methods) to obtain economically software that is reliable and works on real machines". Like any other engineering discipline software engineering involves design, quality assurance, cost estimation and project management. The software design process, like engineering design, consists of design specification, generation of designs, evaluation of designs, comparison of designs,

selection of a design and implementation. Design specification is the formalisation of the design requirements according to an analysis of the user's requirements. Detailed data and process analysis are necessary for proper specification [51,53]. The rest of the thesis provides most of the basic material needed for such analyses for a CBKAS.

By starting with design specification of the software as a whole, decomposing into components each with their own specification and repeating the process until a completely specified system is obtained, the software engineer follows what is called top-down design. Of course, when evaluations of components are made, design and even specification modifications may be required so the process is one of iteration, i.e. not strictly top-down. Just as engineering structures are built from the ground up, so software implementation is usually done from the bottom up, i.e. the small low level routines are first coded and tested, these are then assembled into larger routines and the process continued until the whole system has been coded. However, a relatively new implementation approach is top-down, i.e. coding first the highest level routines and then successively connecting the lower modules into the system and testing them. This approach effectively eliminates integration or assembly of the coded modules. Possibly its main advantage is that, at all stages of development, a partially usable system exists.

The ELIXIR systems design and implementation were primarily top-down. The structure of EDB was designed first and two simple test processors (P) were chosen. Then the structures of FELIX and EDBREP were coded. To complete these three the necessary CRMS routines were coded and inserted into FELIX and EDBREP. These two could then be tested. However, because EDB was stored in binary form it was difficult to tell from the errors in EDBREP output whether the program errors were in FELIX or EDBREP. So to aid this error detection a very simple program (DBDUMP), which simply dumped the contents of EDB, was written. Once EDB initialisation and its reporting were correct PRELIX was coded and the whole system tested. To test the memory management aspects another simple routine DMPMEM, which dumps the contents of the workspace, was written.

In similar fashion the modules ELIXIR, DBNREP and EDBVU were coded. The six modules and many of their submodules communicate only via the databases EDB and EDBn, thus localising error. Error traps, messages and tracebacks were built into the specification of each module. These were primarily intended to aid the user (KAc) but the information it provided was soon realised to be extremely valuable for debugging. Accordingly the specifications were modified to distinguish between possible user errors and errors arising from program bugs. The final test of the ELIXIR system was essentially its use in the examples of chapter five.

In the development of ELIXIR, design and quality assurance (correctness) were the dominant software engineering aspects. Cost estimation was quite crude. From a rough specification, it was decided that development of ELIXIR was feasible within say one man year and it was therefore done. The only project management issue of relevance was the scheduling of module or subroutine development and testing.

Not being a software engineer, most of the above ideas were learned during software development. Perhaps this was fortunate, because had we realised the complexity of the software required, we may never have considered developing it at all.

CHAPTER 5 : EXAMPLES

Two examples of knowledge acquisition are presented in this Chapter. The first to be presented is a more detailed account of the example, concerning NLFRAM given in the introduction to the thesis. This is an important example because it contains aspects of both boundary value problems (the finite element discretisation of the beam) and initial value problems (the response history to an incrementally applied load). NLFRAM typifies more general finite element structural analysis packages. It served as a focus for development of ELIXIR. This does not imply that ELIXIR is tailored towards knowledge acquisition for NLFRAM but it was considered the main test case. The example demonstrates how knowledge concerning an FEA package as a whole is acquired.

The second example deals with a boundary value problem. Plane frames often have curved members and the objective here concerns a proposed method of modelling the curved members using straight frame elements with rigid offsets. The question was how much offset to use to capture the effects of curvature at element level? The importance of this problem to the thesis is that an analytical proof of the validity of the knowledge could be developed. The knowledge may be used to improve a modelling procedure or algorithm (if built into an FEA modelling pre-processor).

It is very important to note that these examples are intended to demonstrate two things, namely

(i) the knowledge acquisition procedure - its concepts, operations, etc.

(ii) the use of a CBKAS (i.e. ELIXIR) - its component programs, databases, etc.

It should also be remembered that the description of these studies are 'rational reconstructions' [27-31, 35-37] of the actual studies. While clearly any actual study is far more complex than any 'rational

reconstruction', we still believe that a 'rational reconstruction' of the events in the study is the clearest way to present it. We have tried to indicate the actual complexity by including feedback and learning explicitly in certain parts of the description but obviously feedback and learning is active throughout knowledge acquisition.

5.1 ACQUIRING KNOWLEDGE ABOUT NLFRAM

The planning phase naturally introduces the problem of acquiring knowledge about NLFRAM so no additional introduction is needed.

The Planning Phase - KAP(P,G) and KAProg

The first step in knowledge acquisition was to define the knowledge acquisition problem, KAP(P,G). This required description of P and a statement of the goal G.

Description of P

Name - NLFRAM
Author - D L Hawla
Language - Fortran 77
Development Environment - CDC Cyber 174 under NOS2.3
Version - December 1984
Application - Structural analysis of plane frames
Problem types - Plane frames of composite construction especially R.C. Euler-Bernoulli beam theory. Material nonlinearity. Geometrical nonlinearity. Static loading. Time-dependent loads. Area of imbedded materials not deducted from matrix material. No internal hinges or slides. Static analysis. Transient dynamic response analysis.

Solution techniques - displacement based finite elements. Virtual work formulation. Incremental analysis with equilibrium iteration for static analysis.

Implicit trapezoidal rule for dynamic analysis. Element integration is numerical. Material point integration always from last converged configuration.

Outputs - displacements, stress resultants, history parameters, computational effort, reactions, residual forces, iteration convergence information, etc.

Goal G

"Find HK characterising an approximate Pareto optimal set for NLFRAM. The HK should ideally span the problem space of P yet be sufficiently simple".

Note how extensive the description of P is. It is important for the KAc to describe P in such detail because it provides him with the necessary perspective for designing a suitable KAProg. If only a simple KAProg is chosen, the HK obtained will need to be generalised to span the whole problem space in a qualitative way. It will then be useful to have an explicit description of this problem space.

From the description of NLFRAM, it should be evident that it is typical of structural FEA packages, so many of the knowledge acquisition concepts illustrated here should generalise easily to other such packages. In fact, they should generalise to almost any computational mechanics software.

The next step was to define a KAProg which would refine suitably the above stated goal. In this example, the KAProg contained only one

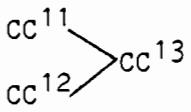
problem class. This problem class PC^1 consisted of the reinforced concrete beam problem of section 1.5. The goal G^1 was then

"Find the HK necessary to solve the reinforced concrete beam problem efficiently. The HK should include a credible approximation to the Pareto optimal set of solution techniques. The criteria for optimality (efficiency) should be based on the reliability of displacements and stress resultants and the cost of the computational effort".

This problem class was chosen for four main reasons. Firstly, one of the primary objectives of developing NLFRAM was to predict the response of reinforced concrete frames. Beams such as the one in Figure 1.5.1 form a typical component of RC frames. The HK obtained would therefore be very useful for modelling the more complex frame problems. Secondly, the beam had been tested in the University of Cape Town Structures Laboratory and there was already great interest in the problem itself. Thirdly, the problem is rich enough to demonstrate the KAS. The richness of the problem will become evident in due course. Finally, the HK would form a good base onto which HK for a more complex KAProg could be built.

After considering all the possible control options provided by NLFRAM for solving PC^1, we decided to restrict ourselves to studying the effect of only five attributes. The rest were to be fixed at some reference state. Selection of the five was based on our judgement of whether an attribute would affect both cost and reliability. If an attribute was expected to have little effect on cost, we simply fixed it so that it would produce a highly reliable solution. An example of this was the selection of the element type. If we already 'knew' (from past experience) which option was the 'best', we again simply fixed it. Selection of the equilibrium iteration algorithm and Gaussian integration order were done in this way. The five attributes finally selected for study were time step size, convergence tolerance, mesh density, cross-sectional depth integration order and the concrete tensile stress release rate (more will be said about the last attribute). However, for each attribute/variable we expected to use at least three values so at least $3^5 = 243$ experiments would be needed.

The number of experiments could hopefully be reduced by applying the following methodological heuristic. Split the five attributes into two control classes say CC^{11} and CC^{12}. First study CC^{11} and use the HK^{11} acquired as a priori HK for the study of CC^{12}. If necessary another control class CC^{13} consisting of all five could be studied using HK^{11} and HK^{12} as a priori knowledge. The relationship between the CC^{ij} can be depicted as follows.



Such an approach is always valid but is only effective if HK^{11} is relevant a priori HK for CC^{12} and HK^{11} and HK^{12} are relevant a priori HK for CC^{13}. At worst one would expect the total effort to be the same as if no split was made. The relevancy of HK^{11} to CC^{12} depends on the nature of the effect of the variables in CC^{11} and CC^{12}. We put time step size and convergence tolerance into CC^{11} and the other three into CC^{12}. We did not expect CC^{11} variables to interact with those in CC^{12} so that having found some values of CC^{11} variables which gave efficient but highly reliable solutions, we could use these values instead of the reference values in the study of CC^{12}. The variables in CC^{11} were both expected to produce quasi-random responses while those in CC^{12} were expected to produce monotonic responses. Because the quasi-random effects could have swamped the monotonic effects and we expected the variables in CC^{11} to have a greater impact on solution costs, we decided to study CC^{11} first. The combined effects were intended to be studied on CC^{13}. The process resembles a hierarchical optimisation process of performing a series of local optimizations on subsystems followed by a global optimization.

The Execution Phase for KAP(P,G^{11})

KAS_0 and $\sum^0$

First we needed to define the constituents of the object system S^0, i.e. (X^0,F^0,Y^0,I^0).

Input or problem attributes:

X^N = {beam length, beam width, beam depth,
position of tension steel, area of tension steel,
position of compression steel, area of compression steel,
concrete parameters (measured)
steel parameters (measured)
boundary conditions,
loading positions,
static load programme}

Processor controls:

F^N = { time step size, convergence tolerances}

Outputs:

Y^N = {displacements, stress resultants reactions, residual forces,
computational costs,}

Indices:

I^N = {time nodal coordinates, Gauss point coordinates}

The sets X^R, F^R, Y^R and I^R, specifying the range of attribute appearances, will often be deferred, as was done in this example until the integration phase of knowledge acquisition.

In order to collect data for studying the system a set of variables had to be defined in order to measure the above attributes of the object system. These variables formed an image system $S^0 = (X^0, F^0, Y^0, I^0)$. Many of these variables corresponded directly to attributes of NLFRAM but some were implicitly input. For example one does not input the beam length directly - one gives nodal coordinates and defines elements between these nodes. These implied quantities are

far more meaningful to the KAc and should be recorded too. The nodes and their coordinates arise out of a discretisation (solution technique) of the beam length (problem variable) and are commonly used to index many quantities, e.g. displacements are generally only given at nodes and point loads and boundary conditions may only be specified at the nodes. Nodal coordinates can therefore be classified neither as inputs nor processor parameters. In ELIXIR they are classified as indices. Indices may be space-like such as nodal or Gauss point coordinates or time-like such as the sequence of times used to index time history responses of output variables.

For the purposes of knowledge acquisition, the following variables were chosen.

X^N = {BMLEN, BMWID, BMDEP,
TSPOS, TSAREA,
CSPOS, CSAREA,
NLMAT,
BCONDS,
LODPOS,
LODVAL, TRL, T}

X^R = {all variables fixed to values shown in Figure 1.5.1}

F^N = {DELTAT,
TOL1
SMODL, SMPARS}

F^R = {(real numbers [1.0, 16.0]); (real numbers [0.1%, 16.0%])

Y^N = {GDISP, BMOM, SHF, REAK,
HIST, RESID, CONRAT, STIF, NITERS,
SRU}

Y^R = {(act real: ([0.0, 0.012), [-43.2], [-96.0, 96.0], [0.0, 96.0], [0.0, 1000.0]}

I^N = {T, COORDS, GPCOD}

The value ranges in Y^R were derived from experimental data and simple hand calculation. The β^N part of the homomorphism linking S^0 and S^0 should be obvious from the above. The definition of β^R was deferred until the integration phase.

By this point $\sum^0$ had been adequately defined and the CBKAS, that is ELIXIR, enters the scene. Recall from chapter 4 that the program FELIX is used to initialise the database EDB. This initialisation consists essentially of setting up $\sum^0$ on EDB. However, whereas the source system $\sum^0$ is concerned only with a single problem class and a single control class, the initialised EDB will usually contain information for a number of source systems. The listing of file SPNLF shown on the next page shows the data used by FELIX to initialise EDB. This in fact will be more than adequate for studying PC^1, CC^{11} and CC^{12}.

```
PROCESSOR NAME  :-
NLFRAM
PROBLEM DESCRIPTION :-
NLFRAM SIMULATION OF TWO-SPAN, MID-SPAN POINT LOADED
REINFORCED CONCRETE BEAM
*
INPUTS :-
 1 , T              * TIME-LIKE INDEX
 2 , BMLEN          * BEAM LENGTH
 3 , BMWID          * BEAM WIDTH
 4 , BMDEP          * BEAM DEPTH
 5 , BSPOS          * BOTTOM STEEL POSITION
 6 , BSAREA         * BOTTOM STEEL AREA
 7 , TSPOS          * TOP STEEL POSITION
 8 , TSAREA         * TOP STEEL AREA
 9 , NLMAT   , I    * MATERIAL REF. CODE
10 , BCONDS , I     * BOUNDARY CONDITION CODE
11 , LODPOS         * LOAD POSITION
12 , LODVAL ,, V    * LOAD VALUES
13 , TRL    ,, V    * LODVAL(I) IS LOAD AT TRL(I)
STATIC SPECIFICATION
1
10                  * I.E. ONLY BCONDS
 0                  * JXDES
*
PROCESSOR PARAMETERS :-
 1 , LTYP    , I    * ELEMENT TYPE
 2 , NELTS   , I    * NUMBER OF ELEMENTS
 4 , DELTAT         * TIME STEP SIZE
 6 , IALGOR , I     * ITERATION ALGORITHM
 7 , TOL1           * RESIDUAL FORCE TOLERANCE
 8 , NGPTS   , I    * NO. OF GAUSS POINTS (LENGTH)
 9 , NDEPTH , I     * NO. OF NEWTON-COTES POINTS (DEPTH)
10 , REFGKT         * FREQUENCY OF STIFFNESS REFORMATION
11 , CMODL  , I     * CONCRETE MATERIAL MODEL REF. CODE
12 , CMPARS ,, V    * CONCRETE PARAMETERS
13 , SMODL  , I     * STEEL MATERIAL MODEL REF. CODE
14 , SMPARS ,, V    * STEEL PARAMETERS
15 , ALPHA          * = CMPARS(1)
STATIC SPECIFICATION
5
4,7,2,9,15          * DELTAT,TOL1,NELTS,NDEPTH,ALPHA
0,0,0,0,0           * JFDES
*
INDEXING VARIABLES :-
1 , T               * TIME-LIKE INDEX
2 , COORDS          * NODAL X-COORD
3 , GPCOD           * GAUSS POINT X-COORD
STATIC SPECIFICATION
3
1,2,3               * I.E. ALL INDICES
* NOTE THAT THE NO. OF VALUES OF ALL INDICES ARE
* ONLY SPECIFIED AT RUN TIME I.E. IN NLFRAM OUTPUT
*
OUTPUTS :-
 1 , GDISP ,, V , 2  * NODAL DISPLACEMENTS
 2 , BMOM  ,,   , 3  * BENDING MOMENTS
 3 , SHF   ,,   , 3  * SHEAR FORCES
 4 , AXF   ,,   , 3  * AXIAL FORCES
 5 , REAK  ,, V      * REACTIONS
 6 , RESID           * RESIDUAL FORCE NORM
 7 , CONRAT,, V      * CONVERGENCE RATIOS
 8 , STIF  ,, V      * STIFFNESS PARAMETERS
 9 , NITERS          * NO. OF ITERATIONS
10 , SRU   , C       * COMPUTATIONAL COSTS
DYNAMIC SPECIFICATION
* OUTPUTS ARE USUALLY DYNAMICALLY SPECIFIED BECAUSE
* ONE OFTEN DISCONTINUES DATA CAPTURE FOR SOME VARIABLES
* AFTER INITIAL EXPERIMENTATION
```

SPNLF is composed of six sections of data, each of which is delimited by ':-', namely PROCESSOR NAME, PROBLEM DESCRIPTION, INPUTS, PROCESSOR PARAMETERS, INDEXING VARIABLES and OUTPUTS. The first two are obvious. Each of the last four sections contains a variable definition subsection and a variable SPECIFICATION subsection. The latter may be either STATIC or DYNAMIC. In the input section, 13 variables are defined but only one, viz. BCONDS, is specified. Although so many variables are defined, data will only be collected for those that are specified. If specification is static, the variable reference numbers are given in SPNLF; if dynamic they are given on the file which contains the output from NLFRAM that is to be captured on EDB. By using dynamic specification, as done for outputs, the KAc may modify, for each run, the specification of which variables data is to be captured. Note however, that all five control variables in CC^{11} and CC^{12} are specified - not just those of CC^{11}. This was done so that SPNLF could be used for both control classes without alteration and so that reference solutions could easily be included in EDB (for the reference solution all five need fixing).

The commentary after the '*' serves to relate the variables to those of the object system, i.e. it may be considered part of the homomorphism β.

Once EDB had been set up with $\sum^0$ on it, the next step was to add data to it to form S^D.

KAS_1 and S^D

Two points are important in data generation. Firstly, the data should be generated according to a carefully designed experimental programme. Secondly, included in this programme should be at least one run whose solution may serve as a reference (reliable, accurate) solution.

The following control values were chosen to represent the reference solution:

DELTAT = 1, TOL1 = 0.1%, NELTS = 16, NDEPTH = 13, ALPHA = 4.

The same solution will be used for CC^{12} hence the explicit inclusion of its control variables in the designation of the reference solution.

First a run of NLFRAM with the reference values was made. Then a programme of 25 experiments was run with each combination of

DELTAT = 1, 2, 4, 8, 16
and TOL1 = 1, 2, 4, 8, 16%.

The data collected from these runs forms S = (X,F,Y,I). This was written to a temporary binary file (TEMPFILE) by NLFRAM and then captured onto EDB by PRELIX. At this stage EDB contained S^D.

KAS_2, $\tilde{S}^D$ and $\hat{S}^D$

The next step was to convert S^D to $\tilde{S}^D$ by introducing a reliability variable (called RELY) and to compute the RELY data from the data Y. But first we needed to choose a means of calculating RELY.

Choosing a reliability/deviation measure

What are the options? One could use measures based on displacements, stress resultants, stresses or reactions at one or more points in the structure. Any of the measures described in Chapter 3 may be used at a particular point in time or averaged over a time period. Measures based on stress resultants or reactions are not particularly useful except to control residual force imbalances (Residual force imbalance is important because the materials are history dependent). Measures based on stresses were not considered because they depend on the stress distribution through the beam depth which depends on the material model chosen. The material models, especially for concrete, are not very well calibrated due to insufficient and/or inappropriate test data (or inappropriate model) - hence the inclusion of the material parameters in F and not X. Measures based on stress would

therefore seem inappropriate. This leaves displacements and reactions. Although it may have been worthwhile to include a reaction based reliability for this problem, only the displacement under the point load was chosen. A reliability measure based on an RMS (over time) relative displacement deviation was used, i.e. used R3/CO/DR/RMS. The reason for taking RMS values is to smooth out the random component in the reliability response caused by the discretisation and iteration.

RELY data was then calculated for all cases in S. Adding this data to S gave $\bar{S}$ and adding RELY to Y^N gave $\bar{Y}^N$ and so $\bar{S}^D$ was formed. This too was held in EDB.

The next step was to form $\hat{S}^D$ from $\bar{S}^D$. Recall that $\bar{S}^D = (\bar{X}^D, \bar{F}^D, \bar{Y}^D, \bar{I}^D)$ whereas $\hat{S}^D = (\hat{X}^D, \hat{F}^D, \hat{Y}^D)$. The idea of this step is to condense the data in $\bar{S}^D$ (which is generally linked to time- and space-like indices and so may be voluminous) down to something more manageable. Often this may be as simple as selecting $\bar{X},\bar{F},\bar{Y}$ data for some specified values of $\bar{I}$ or averaging $\bar{Y}$ data over the $\bar{I}$ values. Reliability calculation already involves similar ideas. In this example RELY is based on midpoint deflection only (i.e. a single value of a space-like variable) but is evaluated for each value of T (i.e. time-like index). To condense further, we selected RELY for T = 96.0 only (i.e. the end of the time period or at maximum load). So whereas $\bar{Y}$ contains bending moments at all Gauss points and displacements at all nodes for each value of T plus costs and reliabilities for all T values, $\hat{Y}$ contains only cost and reliabilities for T = 96.0. $\hat{X}$ and $\hat{F}$ were identical to $\bar{X}$ and $\bar{F}$. This new condensed data system $\hat{S}^D$ is stored not in EDB but in a new database called EDBn (i.e. EDB1, EDB2, etc.). The idea is that the KAc may form more than one EDBn from EDB. Also the storage format in EDBn is more suited to the data processing and manipulation involved in KAS_3.

KAS_3 and $\hat{S}^D_K$

Recall that the objective of this step is to 'partition' $\hat{S}^D$ into $\hat{S}^D_K$. $\hat{X}^D$ requires no 'partitioning' since it contains only one problem. A

'partition' of $\hat{F}^D$ involves findings subsets of DELTAT and TOL1 which are Pareto optimal. Finding Pareto optima involves studying $\hat{Y}$, i.e. the cost and reliability of the solutions. It was found that all solutions had RELY > 99 percent even for DELTAT = 16 and TOL1 = 16 percent and, as expected, cost decreased as DELTAT and TOL1 were increased. So essentially the idea was to use the largest of these values. However there were auxiliary conditions to be met. Firstly as TOL1 increases the residual forces increase. We could have calculated reliability measures based on these residual forces but instead we simply did graphical comparisons of shear force (SHF) and bending moments (BMOM) between the reference and the experimental solutions. In this way we decided that a TOL1 = 4 percent adequately controlled the residual forces. It is important to control the residual forces because the materials in the beam are history dependent. Similarly, it was decided that DELTAT = 16 did not provide adequate detail of the response so DELTAT = 8 was selected. With DELTAT = 8 and TOL1 = 4 percent the cost reduction factor over the reference solution was approximately 5.5 (from 642 to 117 SRU). It was also found that, in the range of values studied, the cost was more sensitive to DELTAT changes than to TOL1 changes.

KAS_4 and HK^{11}

HK^{11} turned out to be extremely simple, that is:

If DELTAT = 8 and TOL1 = 4 percent
then RELY > 0.99 and COST ≈ 117 SRU

Such a statement of the HK^{11} is in terms of variables and values, i.e. image system (S^0) terms. Usually it is also required to convert this to object system (S^0) terms. This was done by first setting up β^R. One suitable selection for β^R was the following:

DELTAT	Time step size	TOL1	Convergence Tolerance
1	Very small	1%	Very tight
2	Small	2%	Tight
4	Medium	4%	Medium
8	Large	8%	Loose
16	Very large	16%	Very loose

RELY	Displacement reliability	COST	Computational Cost
0.999	Very high	117	Very expensive
0.97	High	47	Fairly high
0.94	High-medium	36	High-medium
0.84	Low-medium	23	Low-medium
0.80	Low		
<0.75	Unacceptable		

Note that appearances are adjectives which qualify the attributes. Some of the elements of β^R namely those relating to RELY and COST are in fact based on the study of CC[12] which is yet to be presented. With this interpretation, the HK[11] could be restated as:

If time step size is large and convergence tolerance is medium then displacement reliability is very high and cost is expensive.

This may seem trivial in this example and hopefully it will often be so simple but it is worthwhile because it will usually make explanation (integration phase) easier or more natural. It has the advantage of distinguishing explicitly between the concepts of time step sizes or convergence tolerances and the technical details of their measurement.

The Execution Phase for $KAP(P,G^{12})$

KAS_0 and Σ^0

The only difference between Σ^0 for CC^{12} compared to that of CC^{11} was that F^0 and F^0 changed to:

F^N = {mesh density; order of cross-sectional depth integration, concrete tensile stress release rate}

F^R = will be defined later.

F^N = {NELTS; NDEPTH; ALPHA}

F^R = {(4,6,8,10,12,14,16); (4,5,7,9,11,13); (Real [4.0,12.0])}

The homomorphism β is obvious. SPNLF was identical to the one used for CC^{11} so EDB was initialised in exactly the same way. In fact, all data generated during the study of CC^{12} could have been added to the EDB already formed in the study of CC^{11}. However, we chose to use separate databases for each study.

A very important point to note is that although, in the object system, the materials are part of the problem definition, in practice, when using nonlinear material models, it is often more appropriate to consider than as processor control variables. This has been done in the image system. Clearly CMODL and CMPARS would depend on the measured concrete parameters but usually insufficient measurement data of material behaviour is available. The advantage of doing this is that they then become additional control variables and so may allow additional efficiency to be achieved. Conceptually one may consider their deviations from the measured values as artificial properties. The use of artificial viscosities or damping in some solution process is of this type.

KAS_1 and S^D

The initial programme of experiments consisted of all 27 combinations of

NELTS = 4, 8, 16
NDEPTH = 4, 7, 13
ALPHA = 4, 8, 12

plus the reference solution, a run with reference values except TOL1 = 1.0 instead of 0.1 and by mistake a repetition of one of the above 27. (We did not exclude this mistake from the presentation because, as will be seen in Table 5.1.1, cases 2 and 27, it illustrates a mild but interesting inconsistency with what was said in Chapter 3. We said that repeating a computer run would give identical results, however one can see that the costs are slightly different for cases 2 and 27 even though case 27 was simply a rerun of case 2. Presumably this difference was caused by some inconsistency in the accounting package on the host computer.) All these experiments were done with DELTAT = 8 and TOL1 = 4 percent, i.e. using HK^{11} so that the cost of experimentation was reduced considerably.

The design of additional experimental programmes will not be discussed here. They will rather be discussed in the section dealing with 'partitioning' of $\hat{S}^D$.

KAS_2, $\bar{S}^D$ and $\hat{S}^D$

The reliability measures used were the same as for CC^{11}. $\bar{S}^D$ was thus formed. Using the same selection process as before $\hat{S}^D$ was formed on EDBn. The following table shows $\hat{S}$, sorted into decreasing reliability.

IAXSET = 1 IAXIVR = 4 IARLI = 2

CASE NO.	DELTAT	TOL1	NELTS	NDEPTH	ALPHA	RELY (2)	COST SRU
30	.100E+01	.100E-02	.160E+02	.130E+02	.400E+01	1.00	642.
1	.100E+01	.100E-01	.160E+02	.130E+02	.400E+01	.999	440.
2	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	117.
27	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	116.
18	.800E+01	.400E-01	.800E+01	.130E+02	.400E+01	.977	53.2
15	.800E+01	.400E-01	.800E+01	.700E+01	.400E+01	.952	35.9
24	.800E+01	.400E-01	.160E+02	.700E+01	.400E+01	.942	81.1
28	.800E+01	.400E-01	.160E+02	.130E+02	.800E+01	.866	68.6
25	.800E+01	.400E-01	.160E+02	.700E+01	.800E+01	.860	63.8
19	.800E+01	.400E-01	.800E+01	.130E+02	.800E+01	.850	29.8
9	.800E+01	.400E-01	.400E+01	.130E+02	.400E+01	.848	22.8
16	.800E+01	.400E-01	.800E+01	.700E+01	.800E+01	.844	28.6
6	.800E+01	.400E-01	.400E+01	.700E+01	.400E+01	.831	19.5
21	.800E+01	.400E-01	.160E+02	.400E+01	.400E+01	.826	72.6
12	.800E+01	.400E-01	.800E+01	.400E+01	.400E+01	.812	32.1
20	.800E+01	.400E-01	.800E+01	.130E+02	.120E+02	.800	25.6
29	.800E+01	.400E-01	.160E+02	.130E+02	.120E+02	.800	61.5
10	.800E+01	.400E-01	.400E+01	.130E+02	.800E+01	.799	15.7
17	.800E+01	.400E-01	.800E+01	.700E+01	.120E+02	.797	22.7
26	.800E+01	.400E-01	.160E+02	.700E+01	.120E+02	.795	51.7
7	.800E+01	.400E-01	.400E+01	.700E+01	.800E+01	.790	13.2
3	.800E+01	.400E-01	.400E+01	.400E+01	.400E+01	.764	22.0
22	.800E+01	.400E-01	.160E+02	.400E+01	.800E+01	.752	49.3
11	.800E+01	.400E-01	.400E+01	.130E+02	.120E+02	.750	17.0
8	.800E+01	.400E-01	.400E+01	.700E+01	.120E+02	.743	11.3
13	.800E+01	.400E-01	.800E+01	.400E+01	.800E+01	.738	23.3
4	.800E+01	.400E-01	.400E+01	.400E+01	.800E+01	.702	13.7
23	.800E+01	.400E-01	.160E+02	.400E+01	.120E+02	.701	44.7
14	.800E+01	.400E-01	.800E+01	.400E+01	.120E+02	.697	21.9
5	.800E+01	.400E-01	.400E+01	.400E+01	.120E+02	.655	12.3

TABLE 5.1.1 : $\hat{S}$ (EDBn) after the initial experimental programme

KAS_3 and $\hat{S}_K^D$

Using ELIXIR to process/display EDBn, the following plot was produced.

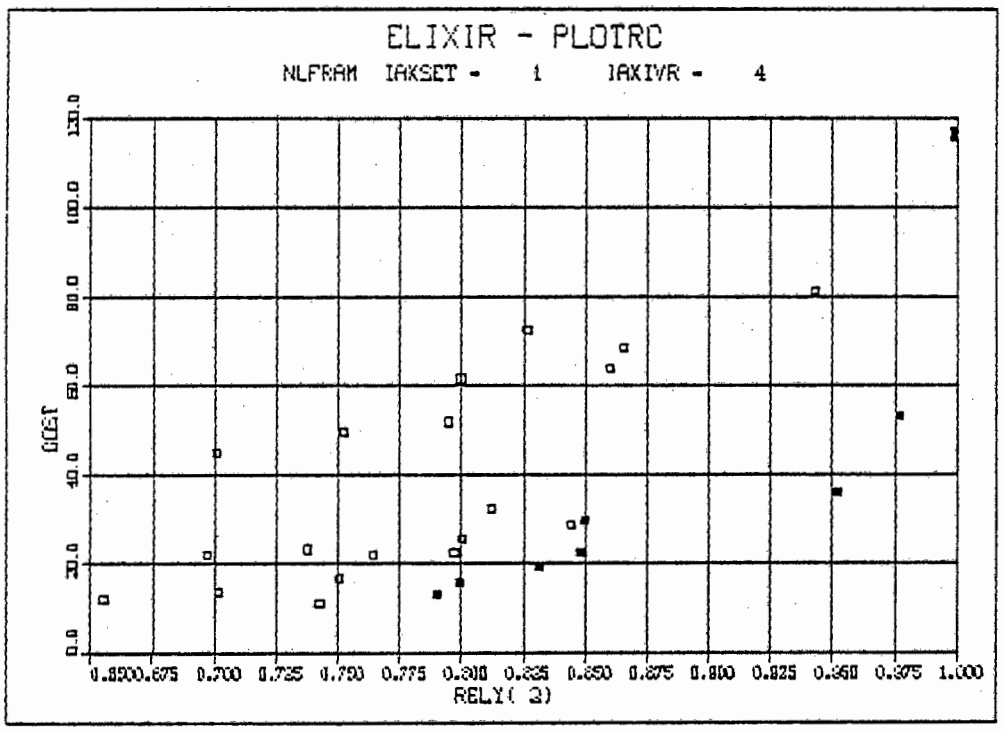


FIGURE 5.1.1 : COST versus RELY for the initial programme of experiments.

Note that in the above figure, the top two case listed in Table 5.1.1 have been omitted. The solid square markers are those which remain after filtering and correspond to those in Figure 5.1.2.

In Figure 5.1.2 only the Pareto optimal points and those with RELY > 0.75 are shown. This latter limit implies that the range of acceptable RELY was limited to [0.75, 1.0].

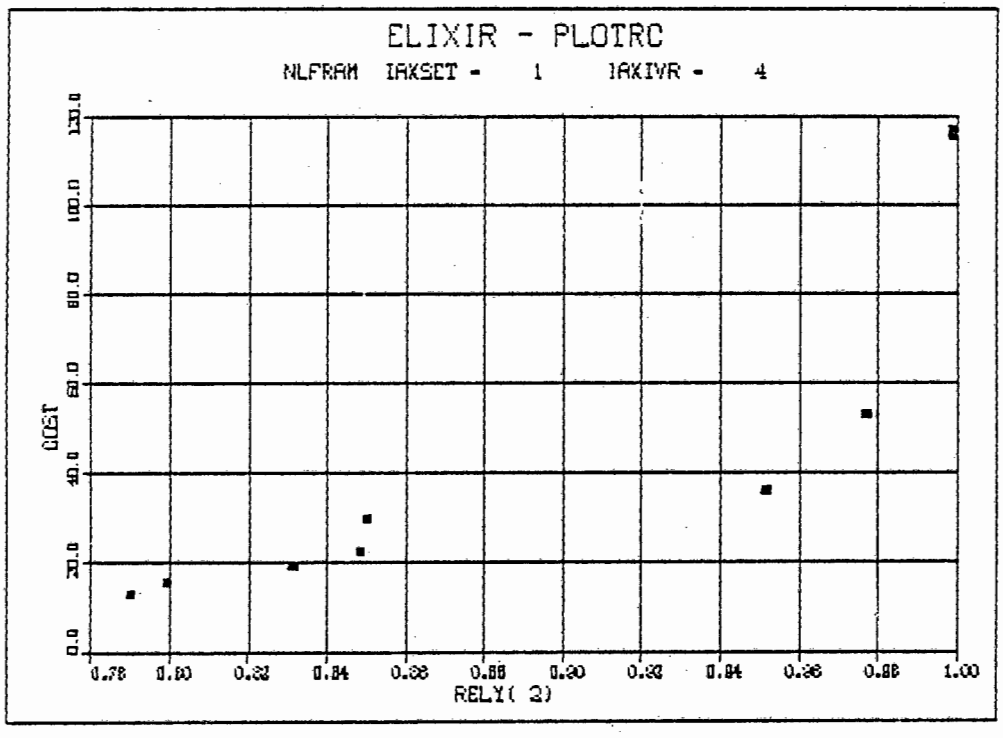


FIGURE 5.1.2 : Filtered RELY-COST data for the initial experiments

Table 5.1.2 below shows the corresponding $\hat{F}$ data.

IAXSET = 1 IAXIVR = 4 IARLI = 2

CASE NO.	DELTAT	TOL1	NELTS	NDEPTH	ALPHA	RELY (2)	COST SRU
2	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	117.
27	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	116.
18	.800E+01	.400E-01	.800E+01	.130E+02	.400E+01	.977	53.2
15	.800E+01	.400E-01	.800E+01	.700E+01	.400E+01	.952	35.9
19	.800E+01	.400E-01	.800E+01	.130E+02	.800E+01	.850	29.8
9	.800E+01	.400E-01	.400E+01	.130E+02	.400E+01	.848	22.8
6	.800E+01	.400E-01	.400E+01	.700E+01	.400E+01	.831	19.5
10	.800E+01	.400E-01	.400E+01	.130E+02	.800E+01	.799	15.7
7	.800E+01	.400E-01	.400E+01	.700E+01	.800E+01	.790	13.2

TABLE 5.1.2 : $\hat{S}$ (EDBn) after filtering.

The filter parameter ε used in the above was very small (0.02).

From Figure 5.1.2 it is clear that the data is very sparse. From Table 5.1.2, it would appear that for solutions to satisfy the filter,

NDEPTH should be restricted to (7, 9, 11, 13) and ALPHA to [4.0, 8.0]. Also since the largest gap in RELY lies in the middle of the range, i.e. 0.85 to 0.95, with NELTS = 4 satisfying the lower part and NELTS = 8 satisfying the higher part, NELTS would be restricted to (4, 6, 8).

Now FB_{31} came into play in the design of additional experiments. The second programme contained runs with NELTS = 4, 6, 8, NDEPTH = 9, 11 and ALPHA = 6, 8. The same procedures of filtering and plotting were followed. Studying these results, it was found that the $\hat{Y}$(RELY and COST) response had a significant quasi-random component. Such randomness aggrevated the sparsity problem which was still evident. It also became clear that to increase the representation (of points) in the higher part of the middle RELY range, i.e. near to 0.95, ALPHA values closer to the reference would be needed. A third programme was designed and the result is shown in Figure 5.1.3 and Table 5.1.3.

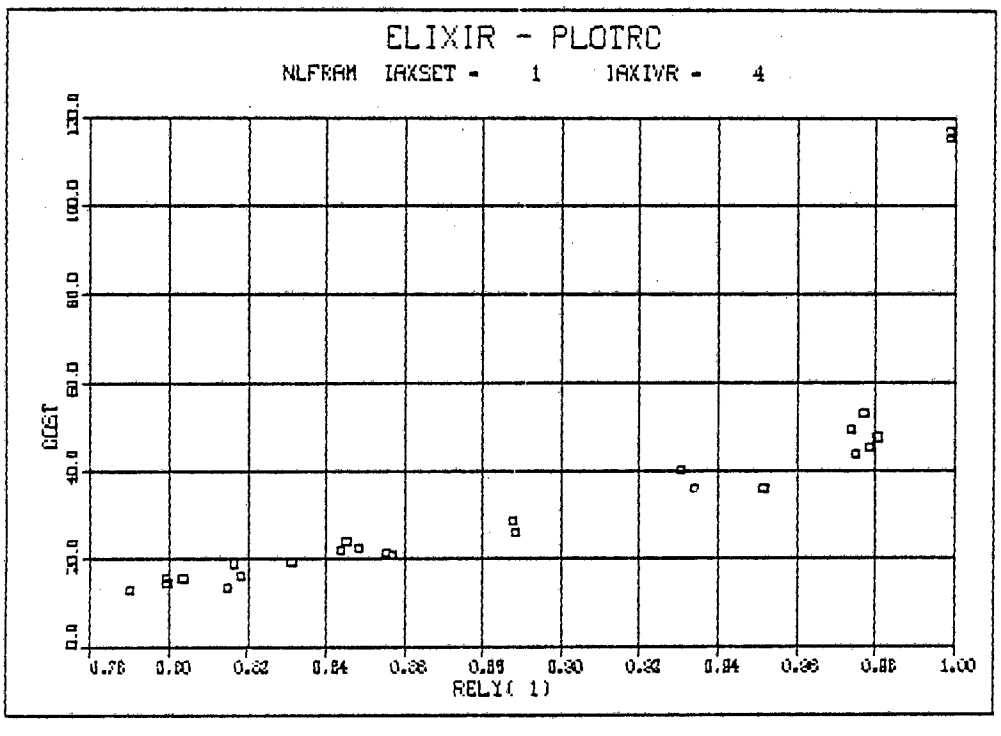


FIGURE 5.1.3 : Filtered $\hat{Y}$ data after the third experimental programme (ε = 0.20).

IAXSET = 1 IAXIVR = 4 IARLI = 1

CASE NO.	DELTAT	TOL1	NELTS	NDEPTH	ALPHA	RELY (1)	COST SRU
2	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	117.
27	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	116.
60	.800E+01	.400E-01	.800E+01	.900E+01	.450E+01	.983	41.4
61	.800E+01	.400E-01	.800E+01	.110E+02	.450E+01	.980	45.9
52	.800E+01	.400E-01	.800E+01	.110E+02	.400E+01	.980	47.6
50	.800E+01	.400E-01	.800E+01	.900E+01	.400E+01	.979	45.3
51	.800E+01	.400E-01	.800E+01	.900E+01	.500E+01	.975	43.9
53	.800E+01	.400E-01	.800E+01	.110E+02	.500E+01	.974	49.5
57	.800E+01	.400E-01	.600E+01	.110E+02	.500E+01	.954	37.1
15	.800E+01	.400E-01	.800E+01	.700E+01	.400E+01	.952	35.9
64	.800E+01	.400E-01	.800E+01	.900E+01	.550E+01	.951	36.8
56	.800E+01	.400E-01	.600E+01	.900E+01	.500E+01	.947	36.2
65	.800E+01	.400E-01	.800E+01	.110E+02	.550E+01	.946	37.5
63	.800E+01	.400E-01	.600E+01	.110E+02	.550E+01	.944	32.9
62	.800E+01	.400E-01	.600E+01	.900E+01	.550E+01	.944	29.0
35	.800E+01	.400E-01	.600E+01	.900E+01	.600E+01	.888	26.0
37	.800E+01	.400E-01	.600E+01	.110E+02	.600E+01	.888	28.7
45	.800E+01	.400E-01	.600E+01	.900E+01	.700E+01	.857	21.2
46	.800E+01	.400E-01	.600E+01	.110E+02	.700E+01	.855	21.5
59	.800E+01	.400E-01	.600E+01	.110E+02	.750E+01	.850	23.1
58	.800E+01	.400E-01	.600E+01	.900E+01	.750E+01	.850	23.3
9	.800E+01	.400E-01	.400E+01	.130E+02	.400E+01	.848	22.8
38	.800E+01	.400E-01	.600E+01	.110E+02	.800E+01	.845	24.2
36	.800E+01	.400E-01	.600E+01	.900E+01	.800E+01	.844	21.9
6	.800E+01	.400E-01	.400E+01	.700E+01	.400E+01	.831	19.5
33	.800E+01	.400E-01	.400E+01	.110E+02	.600E+01	.818	16.6
44	.800E+01	.400E-01	.400E+01	.130E+02	.600E+01	.817	19.2
31	.800E+01	.400E-01	.400E+01	.900E+01	.600E+01	.815	13.9
43	.800E+01	.400E-01	.400E+01	.700E+01	.600E+01	.804	15.9
32	.800E+01	.400E-01	.400E+01	.900E+01	.800E+01	.799	14.7
10	.800E+01	.400E-01	.400E+01	.130E+02	.800E+01	.799	15.7
7	.800E+01	.400E-01	.400E+01	.700E+01	.800E+01	.790	13.2

TABLE 5.1.3 : Filtered $\hat{S}$ data after the third experimental programme ($\varepsilon = 0.20$).

In Table 5.1.3, the $\hat{S}$ data has been split into six clusters. This was done manually. The top four really reflect the clusters in Figure 5.1.3 for medium and high reliabilities and are based on RELY and COST criteria. The bottom two are split according to NELTS. To improve the pattern without loss of information, cases 18, 9 and 6 were removed. The pattern represents the 'partition' $\hat{S}_K$ - it is really a 'partition' of $\hat{F}$ and $\hat{Y}$.

From $\hat{S}_K$ we induced $\hat{S}_K^R$ (i.e. a 'partition' of $\hat{F}^R$ and $\hat{Y}^R$) and a final program of experiments was designed simply to concentrate the representation of $\hat{Y}$ data into these $\hat{F}_{1\ell}^R$ and $\hat{Y}_{1\ell}^R$. The idea was simply to increase the empirical basis for each rule so as to decrease the uncertain effect of the quasi-random component and thereby improve the robustness of the rules.

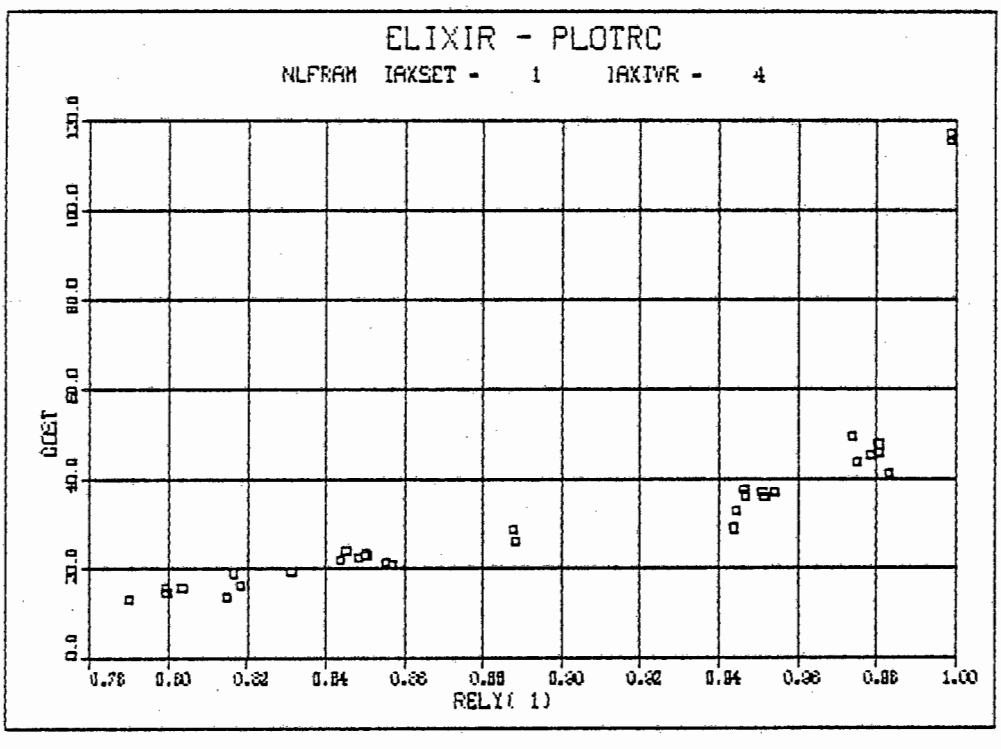


FIGURE 5.1.4 : Filtered $\hat{Y}$ data after the final programme ($\varepsilon = 0.20$)

IAXSET = 1 IAXIVR = 4 IARLI = 1

CASE NO.	DELTAT	TOL1	NELTS	NDEPTH	ALPHA	RELY (1)	COST SRU
2	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	117.
27	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	116.
52	.800E+01	.400E-01	.800E+01	.110E+02	.400E+01	.980	47.6
50	.800E+01	.400E-01	.800E+01	.900E+01	.400E+01	.979	45.3
18	.800E+01	.400E-01	.800E+01	.130E+02	.400E+01	.977	53.2
51	.800E+01	.400E-01	.800E+01	.900E+01	.500E+01	.975	43.9
53	.800E+01	.400E-01	.800E+01	.110E+02	.500E+01	.974	49.5
15	.800E+01	.400E-01	.800E+01	.700E+01	.400E+01	.952	35.9
39	.800E+01	.400E-01	.800E+01	.900E+01	.600E+01	.934	36.1
41	.800E+01	.400E-01	.800E+01	.110E+02	.600E+01	.931	40.2
35	.800E+01	.400E-01	.600E+01	.900E+01	.600E+01	.888	26.0
37	.800E+01	.400E-01	.600E+01	.110E+02	.600E+01	.888	28.7
45	.800E+01	.400E-01	.600E+01	.900E+01	.700E+01	.857	21.2
46	.800E+01	.400E-01	.600E+01	.110E+02	.700E+01	.855	21.5
9	.800E+01	.400E-01	.400E+01	.130E+02	.400E+01	.848	22.8
38	.800E+01	.400E-01	.600E+01	.110E+02	.800E+01	.845	24.2
36	.800E+01	.400E-01	.600E+01	.900E+01	.800E+01	.844	21.9
6	.800E+01	.400E-01	.400E+01	.700E+01	.400E+01	.831	19.5
33	.800E+01	.400E-01	.400E+01	.110E+02	.600E+01	.813	16.6
44	.800E+01	.400E-01	.400E+01	.130E+02	.600E+01	.817	19.2
31	.800E+01	.400E-01	.400E+01	.900E+01	.600E+01	.815	13.9
43	.800E+01	.400E-01	.400E+01	.700E+01	.600E+01	.804	15.9
32	.800E+01	.400E-01	.400E+01	.900E+01	.800E+01	.799	14.7
10	.800E+01	.400E-01	.400E+01	.130E+02	.800E+01	.799	15.7
7	.800E+01	.400E-01	.400E+01	.700E+01	.800E+01	.790	13.2

TABLE 5.1.4 : Filtered $\hat{S}$ data after the final programme ($\epsilon = 0.20$)

Now the simplicity conditions are invoked to eliminate those cases from the above table which would complicate the rules or the set of rules. In this study, cases 15, 64 and 65 were removed from the third cluster, the whole of the fourth cluster was removed, i.e. cases 35 and 37 were removed, case 9 was removed from the fifth cluster and cases 6, 33, 44 and 10 were removed from the last cluster. This resulted in the following clusters.

IAXSET = 1 IAXIVR = 4 IARLI = 1

CASE NO.	DELTAT	TOL1	NELTS	NDEPTH	ALPHA	RELY (1)	COST SRU
2	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	117.
27	.800E+01	.400E-01	.160E+02	.130E+02	.400E+01	.999	116.
60	.800E+01	.400E-01	.800E+01	.900E+01	.450E+01	.983	41.4
61	.800E+01	.400E-01	.800E+01	.110E+02	.450E+01	.980	45.9
52	.800E+01	.400E-01	.800E+01	.110E+02	.400E+01	.980	47.6
50	.800E+01	.400E-01	.800E+01	.900E+01	.400E+01	.979	45.3
51	.800E+01	.400E-01	.800E+01	.900E+01	.500E+01	.975	43.9
53	.800E+01	.400E-01	.800E+01	.110E+02	.500E+01	.974	49.5
57	.800E+01	.400E-01	.600E+01	.110E+02	.500E+01	.954	37.1
56	.800E+01	.400E-01	.600E+01	.900E+01	.500E+01	.947	36.2
63	.800E+01	.400E-01	.600E+01	.110E+02	.550E+01	.944	32.9
62	.800E+01	.400E-01	.600E+01	.900E+01	.550E+01	.944	29.0
45	.800E+01	.400E-01	.600E+01	.900E+01	.700E+01	.857	21.2
46	.800E+01	.400E-01	.600E+01	.110E+02	.700E+01	.855	21.5
59	.800E+01	.400E-01	.600E+01	.110E+02	.750E+01	.850	23.1
58	.800E+01	.400E-01	.600E+01	.900E+01	.750E+01	.850	23.3
38	.800E+01	.400E-01	.600E+01	.110E+02	.800E+01	.845	24.2
36	.800E+01	.400E-01	.600E+01	.900E+01	.800E+01	.844	21.9
31	.800E+01	.400E-01	.400E+01	.900E+01	.600E+01	.815	13.9
43	.800E+01	.400E-01	.400E+01	.700E+01	.600E+01	.804	15.9
32	.800E+01	.400E-01	.400E+01	.900E+01	.800E+01	.799	14.7
7	.800E+01	.400E-01	.400E+01	.700E+01	.800E+01	.790	13.2

TABLE 5.1.5 : $\hat{S}$ clusters after simplification - these represent $\hat{S}_K^R$

KAS_4 and HK^{12}

$\hat{S}_K^R$ could now be formulated as the following five rules.

Rule 1: If DELTAT = 8 and TOL1 = 4% and NELTS = 16 and NDEPTH = 13
and ALPHA = 4
Then RELY ≈ 0.999 and COST ≈ 117 SRU.

Rule 2: If DELTAT = 8 and TOL1 and NELTS= 8 and DEPTH ε (9,11)
and ALPHA ε [4.0, 5.0]
Then RELY ≈ 0.97 and COST ≈ 47 SRU.

Rule 3: If DELTAT = 8 and TOL1 = 4% and NELTS = 6 and NDEPTH ε (9,11)
and ALPHA ε [5.0, 5.5]
Then RELY ≈ 0.94 and COST ≈ 36 SRU.

Rule 4: If DELTAT = 8 and TOL1 = 4% and NELTS = 6 and NDEPTH ε (9,11)
and ALPHA ε [7.0, 8.0]
Then RELY ≈ 0.84 and COST ≈ 23 SRU.

Rule 5: If DELTAT = 8 and TOL1 = 4% and NELTS = 4 and NDEPTH ε (7, 9)
and ALPHA ε [6.0, 8.0]
Then RELY ≈ 0.80 and COST ≈ 15 SRU.

It should be clear that the removal of some cases from the clusters in Table 5.1.4 simplifies the rules. The fourth cluster, i.e. cases 35 and 37 were removed completely because of an inadequate empirical support and slight variations of ALPHA would seem to place the RELY and COST into the one of the neighbouring clusters. The heuristic nature of the process should also be evident.

For explanation and interpretation, the above rules needed to be restated in terms of S^0. For this purpose, tables relating S^R to S^R needed to be defined. One possible interpretation is the following.

NELTS	Mesh density	NDEPTH	Number of Newton-Cotes depth integration points
2	Very low	[4,5]	Low
4	Low	[7,9]	Medium
6	Medium	[9,11]	High
8	High	13+	Very high
16	Reference		

ALPHA	Concrete tensile stress release rate
4.0	Reference
[4.0, 5.0]	Very Fast
[5.0, 5.5]	Fast
[6.0, 8.0]	Medium
[7.0, 8.0]	Slow - medium
12.0	Slow

Many of these variables are real but only single values instead of intervals are given. This is quite acceptable because the user only needs a range of values from which to choose - the particular values are never really important. Of course this is only true for F and Y or $\hat{Y}$. For X the user needs the whole of X^R to be represented. If the HK is given only for points in X^R, it must be possible to interpolate to get HK for any other point in X^R. Interpolation of F and Y or $\hat{Y}$ is usually not valid.

With the above interpretation, HK^{12} can now be given as:

If the time step size is large and the convergence tolerance is medium, then:

Rule 1: If the reference mesh density, depth integration and concrete tensile stress release rates are used, then displacement reliability will be very high and the cost will be very expensive.

Rule 2: If the mesh density is high, depth integration high and concrete tensile stress release rate very fast then displacement reliability will be high and cost fairly high.

Rule 3: If the mesh density is medium, depth integration high and concrete tensile stress release rate fast, then displacement reliability will be high-medium and cost high-medium.

Rule 4: If the mesh density is medium, depth integration high and concrete tensile stress release rate slow-medium, then displacement reliability will be low-medium and cost will be low-medium.

Rule 5: If the mesh density is low, depth integration medium and concrete tensile stress release rate medium, then displacement reliability will be low and cost low.

In addition to this should be given the values of all other F variables that were fixed, the values used in the reference solution, the diagrams of the problem class and reference solution, all X variable values (summarised forms), a listing of the input data for at least one of the runs and an explanation of the HK.

The Integration Phase

The five rules given above already represent the integration of HK^{11} and HK^{12} into HK^{1}. The integration in this study was easy. Perhaps integrating HK^{ij} into HK^{i} will always be easy. Because the HK^{11} was eventually based primarily on auxiliary considerations rather than optimality considerations, the study of CC^{12} using HK^{11} is effectively the same as studying CC^{13}. Hence no work was done on CC^{13} itself.

A very important component of the integration phase is the explanation of the HK^{1}.

Explanation of HK^{1}

For linear analysis two elements would give exact solutions in terms of Euler-Benoulli beam theory. Although the bending moment distribution is piecewise linear, due to material nonlinearity, the curvature and axial strain distribution will be nonlinear. As the load increases this nonlinearity will be come more severe. The beam elements used have cubic transverse and quadratic axial displacement fields giving linear curvature and linear axial strain fields. Thus to capture reliably these nonlinear strain distributions requires high mesh density at high loads.

The concrete constitutive law used is highly nonlinear so a high number of Newton-Cotes points are needed for integration through the depth of the concrete cross-section, except when a low cost low reliability solution is sought.

ALPHA is part of the material specification and so should be in X^0 and not F^0. However, from the HK it is clear that using ALPHA slower than the reference value, reliable solutions can still be obtained. With ALPHA closer to the reference value, solutions become expensive. This is because the higher the stress release rate, the more strain energy needs to be redistributed through the structure for fixed time (load) step size. This will usually require more equilibrium iteration and hence higher costs. The reason one can still obtain high reliability with 'incorrect' ALPHA is that for most of the response the steel dominates the behaviour.

The recommendation (HK is always regarded as recommendation) of large time step size but not very large is based purely on the grounds of adequacy of response representation since even very large steps produced very highly reliable solutions.

Tightening convergence tolerances results in high cost while loose tolerances result in solutions oscillating about the reference solution. Use of a medium tolerance thus represents a compromise. At this level the displacement oscillations are very small indeed.

Another very important aspect of the integration phase is the generalisation of the HK to a wider problem class. Such generalisation is based on an understanding of the physics and numerical methods underlying the processor. An explanation of the HK is therefore a prerequisite to such generalisation.

Generalising the HK

HK^1 is formally only applicable to the single problem from which it was derived. Heuristically, however, it is much more widely applicable, i.e. it will be applicable to a large proportion of the ranges of the input attributes or variables. In a more detailed knowledge acquisition exercise all relevant and dangerous generalisations would need to be spelled out. Here, we will only show some typical ones.

Clearly the HK is applicable to any problem which has the same relative dimensions, steel ratios, material properties, boundary conditions and relative loading, i.e. the HK is easily nondimensionalised. However, even for wide ranges of these nondimensional variables, the HK will still apply. For example changing the relative steel content, moving the loading position, changing span to depth ratios, beam to width ratios, applying an axial load and changing the steel positions will not affect the HK significantly. From previous experience with the concrete model used, the response is almost insensitive to variations of up to 30 percent in the compression parameters. Changing the tension parameters affects only the main cracking stage's response but can significantly affect costs (This has already been explained). As long as the steel model is the same, varying the parameters will not affect the HK - if the ratio E_1/E and E_H/E (see Figure 1.5.1) are reduced, the HK may need modification due to increased nonlinearity while if they are increased, the HK will be conservative.

Changing the form or structure of the material models may well have a significant affect on the HK. Without knowing what such changes are likely to be, one cannot say much about the nature of these changes. However, the above used models capture quite well the behaviour of the reinforced concrete beam so changing to any models which are capable of similar prediction will probably not affect the HK too greatly. Certainly the above HK would be an excellent starting point for further knowledge acquisition.

Altering the boundary condition may have the most significant effect. This is because, once the concrete has cracked, the neutral axis position moves, so if the beam ends are restrained axially, significant arching action may occur. Addition of end moment restraint will also usually change the HK - probably higher mesh densities will be required. So further knowledge should be acquired for these different boundary conditions. Again the above HK will be an excellent starting point.

Once the HK is expanded to include various boundary conditions, it becomes possible to consider its application in general plane frames. For example a two-bay portal frame would be composed of at least five members. For reliability estimates, the HK may be used for each member taken separately. Cost estimates based on summing the individual member estimates will always be too low for any structure which has complex connections, i.e. where more than two members meet at a joint. In fact such estimates may be much too optimistic. Perhaps, another class of problems containing problems of varying joint complexity should be studied. This will be left to subsequent research.

Further generalisation to dynamic analysis, stability analysis and general geometric nonlinearity is unwise.

5.2 CURVED BEAM SIMULATION

The most important effect of element curvature, namely membrane-bending coupling, is neglected when straight elements are used. On the other hand, intrinsically curved elements are difficult to formulate and are usually expensive, while use of shallow arch type theories with low-order elements presents a problem called 'membrane locking'. To overcome this latter difficulty, reduced integration, modified membrane strain definitions and penalty methods have been used [71-75]. Here we take a different approach. We propose to simulate a curved element by a straight element with a constant offset. The main advantages of this approach are simplicity and no locking.

Attention has been restricted to planar beam elements.

OFFSET BEAM ELEMENT

The curved beam element shown in Figure 5.2.1 may be considered as a beam with varying offset. An approximation to this curved element is to use a straight two-noded Hermitian beam element with a constant offset as shown in Figure 5.2.2. Such an element is probably the simplest element which includes membrane-bending coupling at the element level.

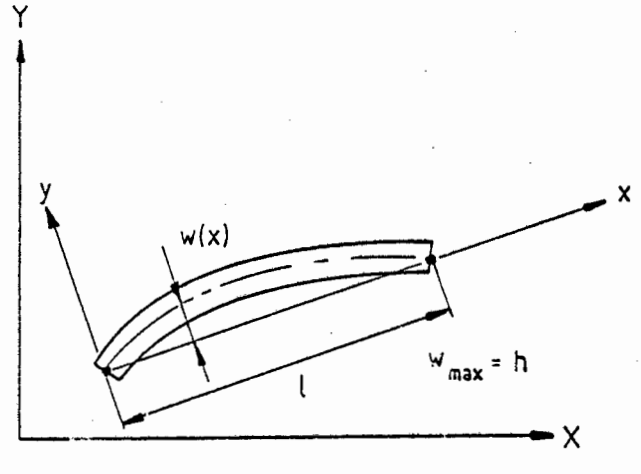


FIGURE 5.2.1 : Curved Element

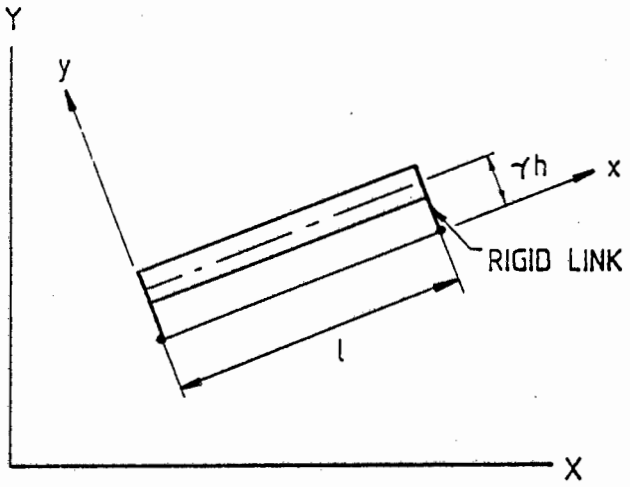


FIGURE 5.2.2 : Offset Straight Element

One suggestion for the offset is to use the average of w(x) over length ℓ. If w(x) were quadratic this would yield γ=2/3.

The Planning Phase - KAP(P,G) and KAProg

The FEA package to test the offset beam proposal was again NLFRAM. Thus P is as before.

The goal G is now to find the 'best' values for γ and the conditions under which these values are applicable. 'Best' here was interpreted as 'Pareto optimal'. Two approaches were taken, namely, the knowledge acquisition approach based on experimentation on a class of problems and a theoretical approach (This will be given later).

Figure 5.2.3 below shows the deep circular arch on which the problem class PC^1 was based.

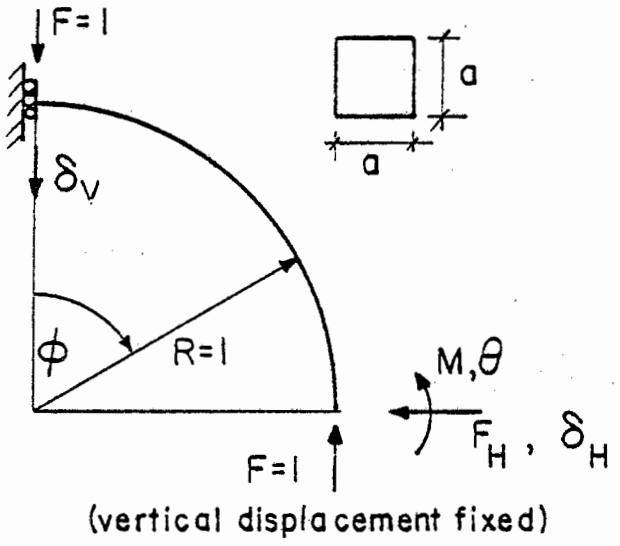
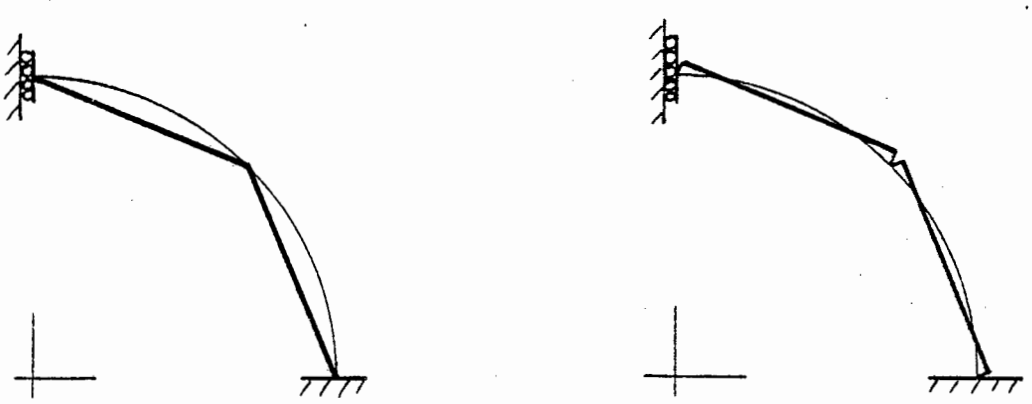


FIGURE 5.2.3 : Deep Arch Used To Define Problem Class

Furthermore, it was assumed that the boundary conditions at the right hand support were such that $M\theta=F_H\delta_H=0$ so that the only external loading was the unit vertical force at top of the arch. Typical finite element models of the arch are shown in Figure 5.2.4.



(a) Standard Straight Elements (b) Offset Straight Elements

FIGURE 5.2.4 : Typical Finite Element Models Of The Deep Arch

The control class CC[11] for this study consisted simply of the offset parameter (Y) and the mesh density (to be represented by NELTS).

So G^{11} is: Find HK^{11} characterising the approximate Pareto optimal set for PC^1 based on CC^{11}.

KAS_0 and $\sum^0$

The components of the object system S^0 were defined as follows:

X^N = {beam width (=depth); boundary condition}
F^N = {mesh density, Y}
Y^N = {displacements, stress resultants, reactions}
I^N = {angular position}

To measure the attributes of the object system the following image system S^0 was formed:

X^N = {BMWID, BCONDS}
X^R = {(Real [0.001,0.1]) (00, 01, 10, 11)}
F^N = {NELTS, GAMMA}
F^R = {(Integer [1,30]); (Real [0.0, 1.0])}
Y^N = {GDISP, BMOM, SHF, AXF, REAKS, KOST}
Y^R = {All real: [0.0, 15.0], [-1.0, 1.0], [-1.0, 1.0], [-1.0, 0.0], [-1.0, 1.0], [1.0, 30.0]}
I^N = {PHINOD, PHIGP}
I^R = {(Real [0.0, $\pi/_2$]; (Real [0.0, $\pi/_2$])}

The values of BCONDS constituted a code for the boundary conditions: 0 means free and 1 means fixed - the first digit representing θ and the second δ_H (See Figure 5.2.3) (i.e. 00 - δ_H and θ free; 01 - δ_H fixed; 10 - θ fixed; 11 - both fixed).

GDISP has three components per node and is indexed by PHINOD.
BMOM, SHF and AXF are indexed by PHIGP.
REAKS is a vector with component numbers being required.

The value ranges in Y^R were derived from very simple hand calculations. They provided valuable validity checks on the data captured.

The homomorphism β^N is clear from the above. Because we will not need to use attribute appearances, their ranges and the homomorphism β^N will be left undefined.

The angles (ϕ) for the two Gauss points of an element correspond to those of points on the arch found by projecting, perpendicularly to the chord, the Gauss points on the (straight) element. The cost variable KOST here is equal to NELTS because storage and processing costs will be directly proportional to NELTS in this problem class (geometric/topological simplicity and linear analysis).

Data for three reactions were captured, namely REAKS(1) = M(0), REAKS(2) = $M(\frac{\pi}{2})$ and REAKS(3) = F_H

KAS_1 and S^D

The initial experimental programme consisted of all combinations of:

X : BMWID = 0.01
BCONDS = 00, 01, 10, 11

and

F : NELTS = 1, 2, 3
GAMMA = 0.0, 0.5, 0.6667, 1.0

Added to this was the reference solutions (one for each X_α) which gives a total of 52 runs. This data forming S was stored on EDB.

KAS_2, $\tilde{S}^D$ and $\hat{S}^D$

The reference solution was chosen as a finite element solution using 30 elements without offset.

$\tilde{S}^D$ was merely the addition of deviation measure data to S^D. For this example a number of deviation measures were calculated.

RELY(1) : deviation based on δ_v

(2) : deviation based on θ

(3) : deviation based on δ_H

(4) : deviation based on F_H

(5) : deviation based on M

(6) : RMS deviation of BMOM normalised with respect to RMS of reference BMOM. In addition, the sign of the mean deviation was attached.

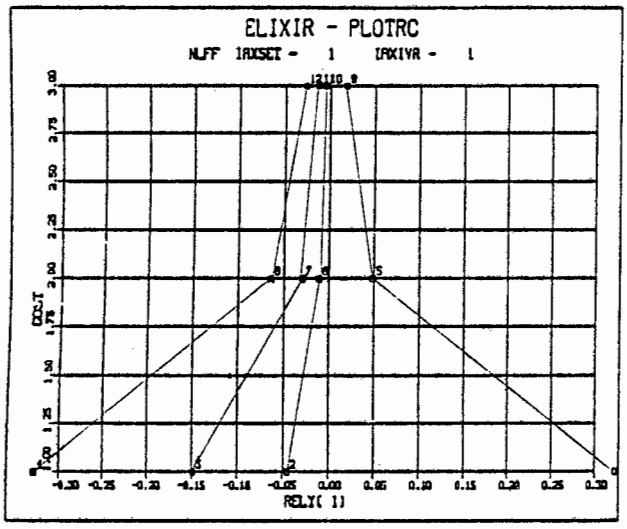
RMS values are calculated over the PHIGP values.

The selection of $\hat{S}^D$ from $\bar{S}^D$ was as follows:

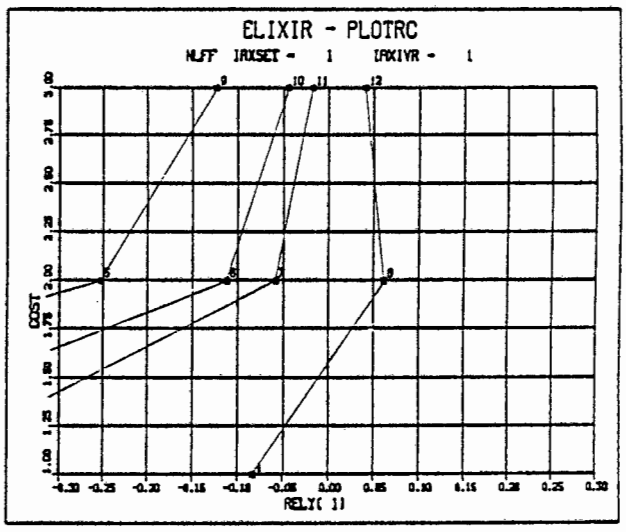
$\hat{X}^D = \bar{X}^D$, $\hat{F}^D = \bar{F}^D$ and $\hat{Y}^D$ = (RELY (1 to 6), KOST). $\hat{S}^D$ was then stored on EDBn. In fact, since the four BCONDS values induced a set of four subclasses of $\hat{X}$, each of these was stored on separate EDBn, namely TAPE41 (BCONDS=00) TAPE42 (BCONDS=01), TAPE43 (BCONDS=10) and TAPE44 (BCONDS=11).

KAS_3 and $\hat{S}^D_K$

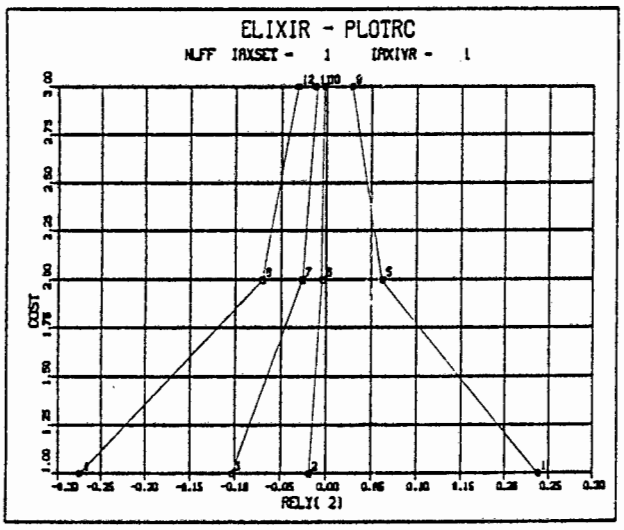
Figure 5.2.5 shows the RELY-KOST plots for the sets of cases (the reference cases have been removed). The lines joining the points have been added only to ease interpretation of the plots - they do NOT imply interpolation of the cost (which is equal to NELTS and is therefore integral). Note also that RELY(6), i.e. BMOM based reliability is not presented here. The reason is that, while RELY(1-5) are easy to calculate, it was not obvious how to calculate a BMOM based reliability. The RELY(6) as defined above was in fact only decided at a later stage, i.e after plots of BMOM versus PHIGP had been studied extensively.



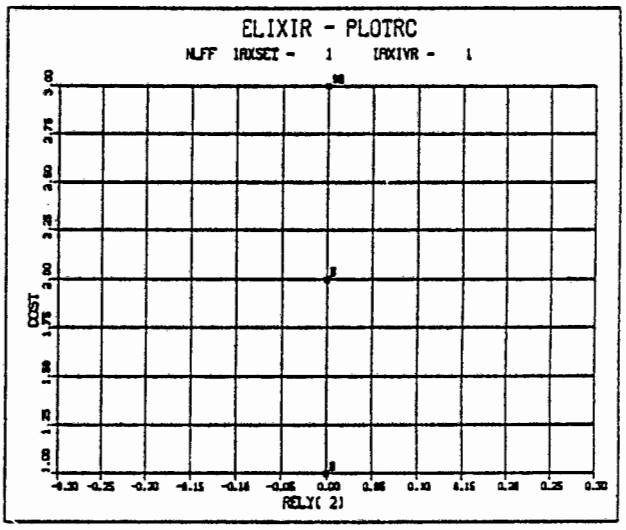
(a) (i) RELY(1) : δ_V based



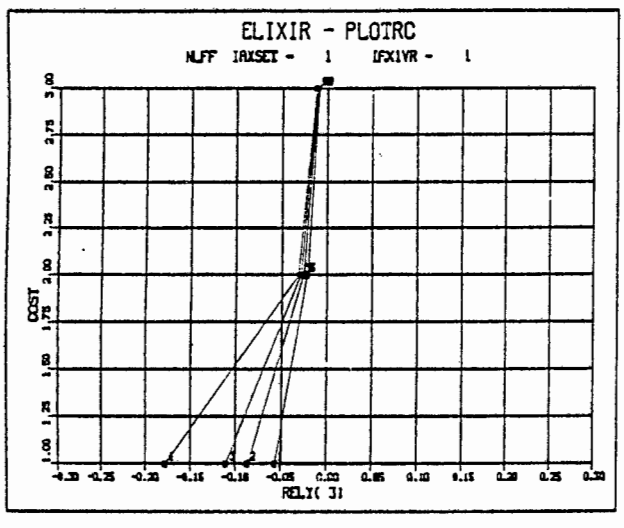
(b) (i) RELY(1) : δ_V based



(a) (ii) RELY(2) : θ based

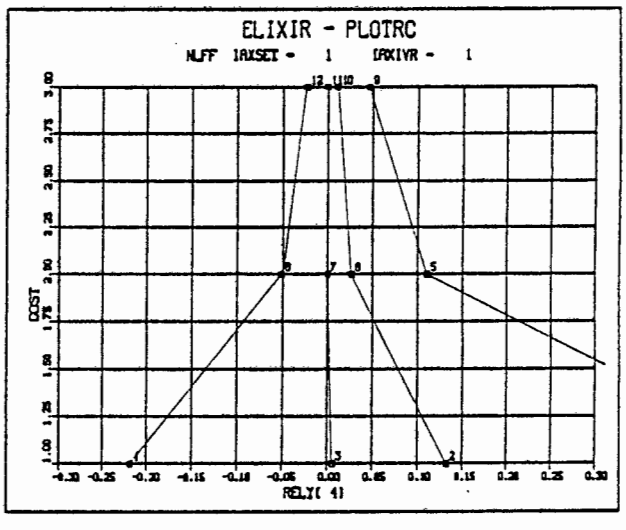


(b) (ii) RELY(2) : θ based



(a) (iii) RELY(3) : δ_H based

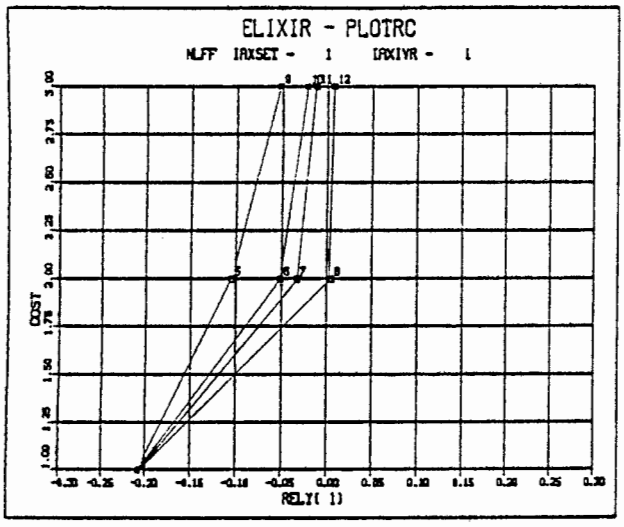
(a) θ and δ_H free (BCONDS=00)



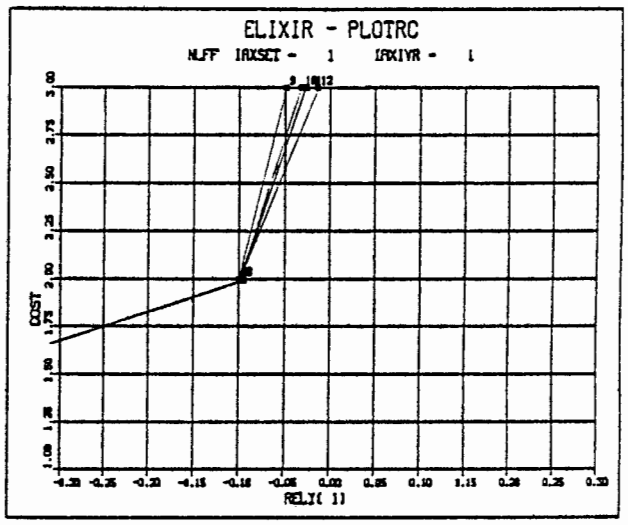
(b) (iii) RELY(4) : F_H based

(b) θ free, δ_H fixed (BCONDS=01)

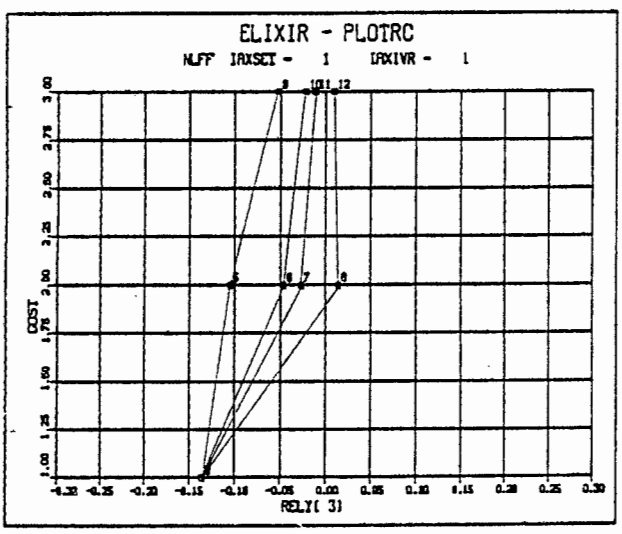
FIGURE 5.2.5 : RELY-COST plots for the various reliability measures and boundary conditions



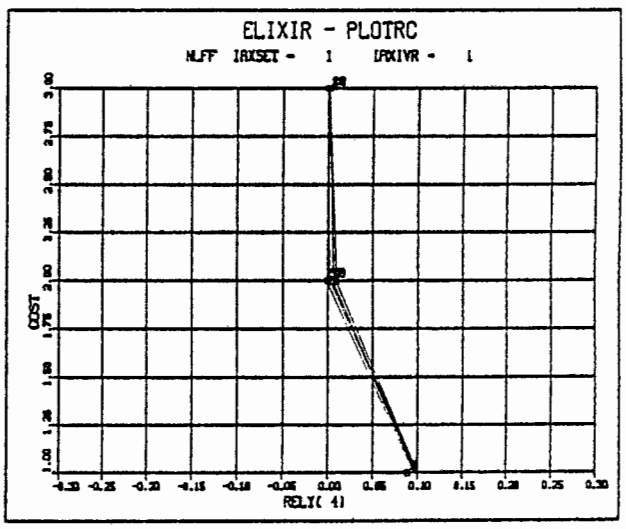
(c) (i) RELY(1) : δ_V based



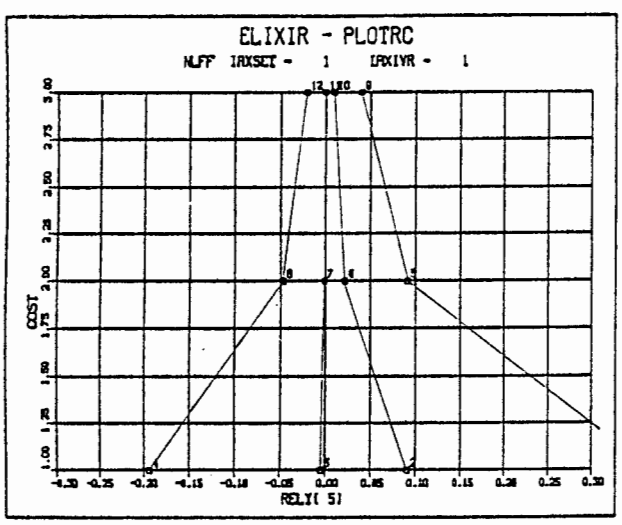
(d) (i) RELY(1) : δ_V based



(c) (ii) RELY(3) : δ_H based

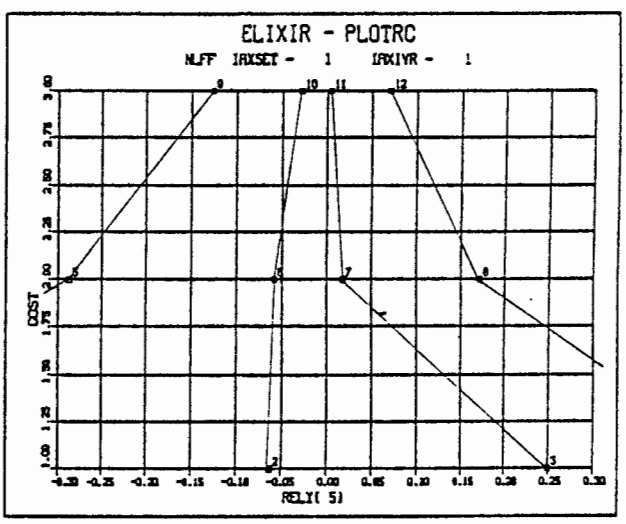


(d) (ii) RELY(4) : F_H based



(c) (iii) RELY(5) : M based

(c) θ fixed, δ_H free (BCONDS=10)

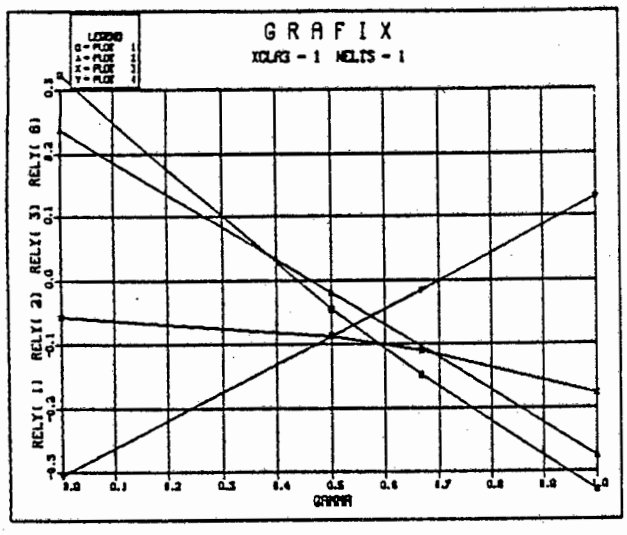


(d) (iii) RELY(5) : M based

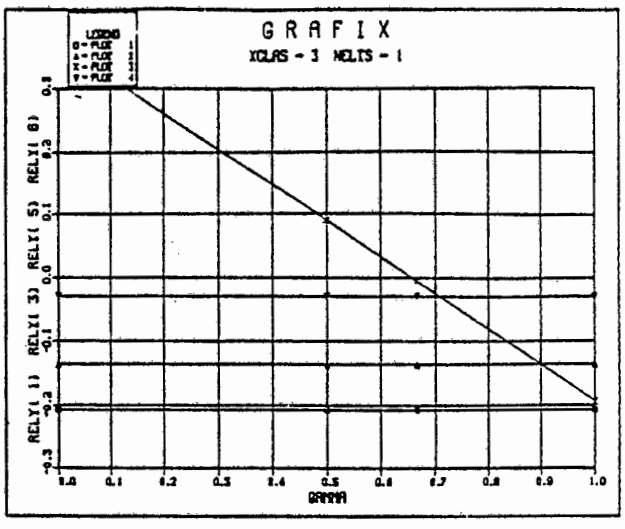
(d) θ and δ_H fixed (BCONDS=11)

FIGURE 5.2.5 : Continued

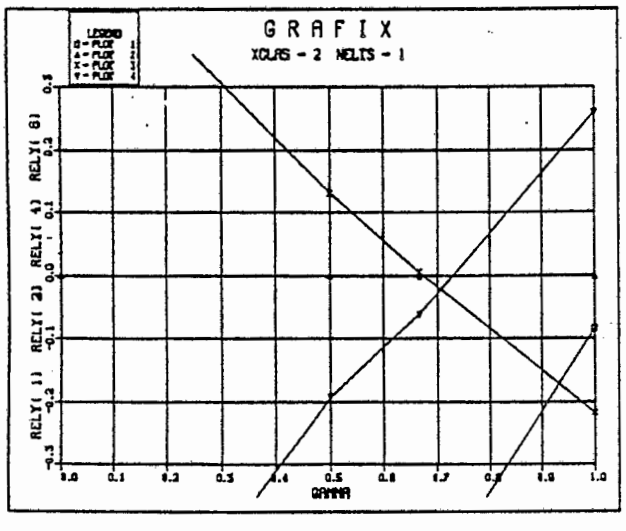
Each graph shows the reliability against cost for a particular reliability measure and BCONDS setting. Clearly the points may be clustered according to KOST(=NELTS). Within each of these clusters the reliability varies with γ. Mostly the sensitivity of the variation decreases with increasing NELTS (All graphs are to the same scale). This is as expected. However, the quantity of data, while still very sparse (only four values of γ) is still too much to apprehend in its present form. If filtering to reduce the data is done on such a sparse sample one would remove nearly all the data. Thus more points based on different γ are needed plus a means of summarising the presented data. In order to discover an appropriate means to achieve both these aims, the presentation of the data was changed to the one in Figure 5.2.6.



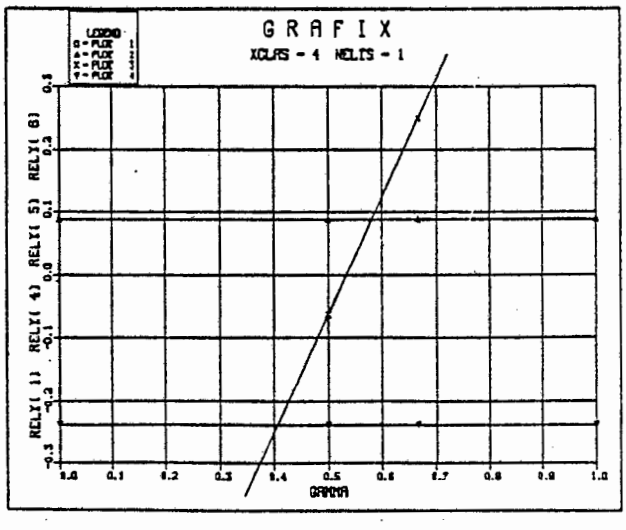
(i) θ and δ_H free (BCONDS=00)



(iii) θ fixed, δ_H free (BCONDS=10)

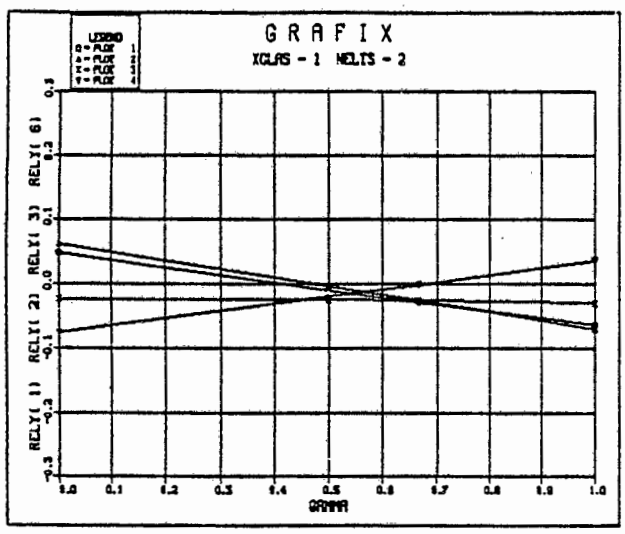


(ii) θ free, δ_H fixed (BCONDS=01)

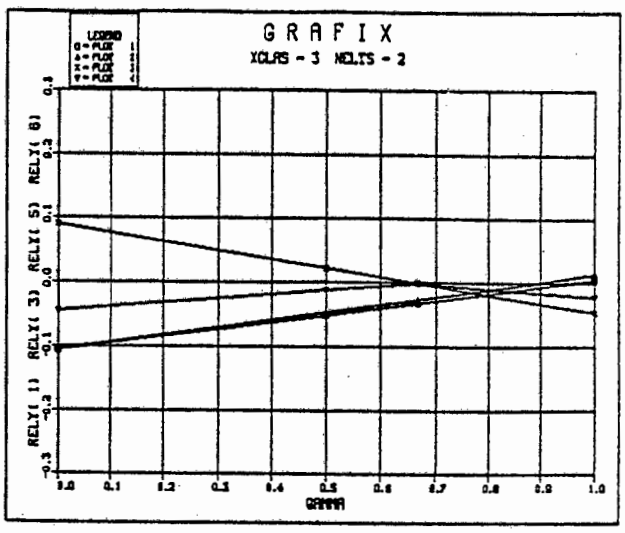


(iv) θ and δ_H fixed (BCONDS=11)
(Note RELY(1)=-1.0)

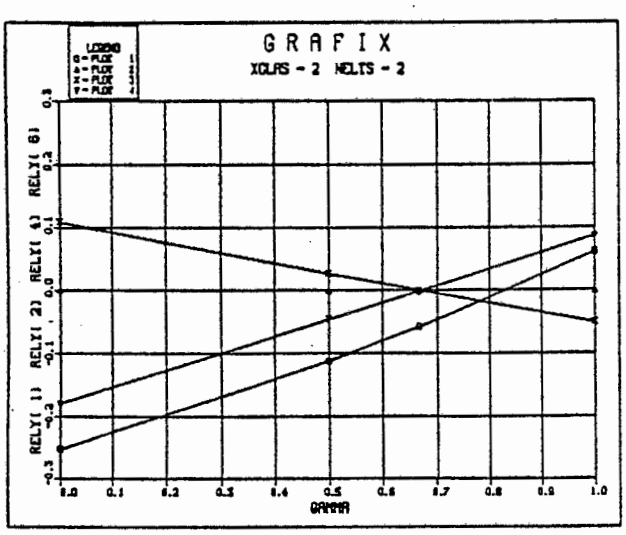
FIGURE 5.2.6(a) : γ versus RELY(i) for NELTS = 1



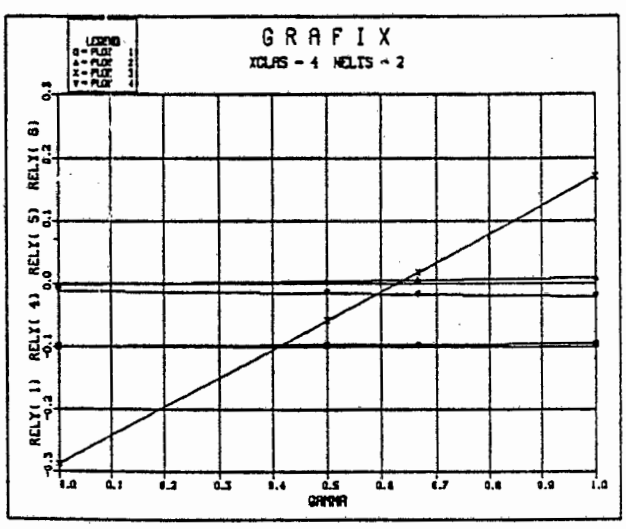
(i) θ and δ_H free (BCONDS=00)



(iii) θ fixed, δ_H free (BCONDS=10)

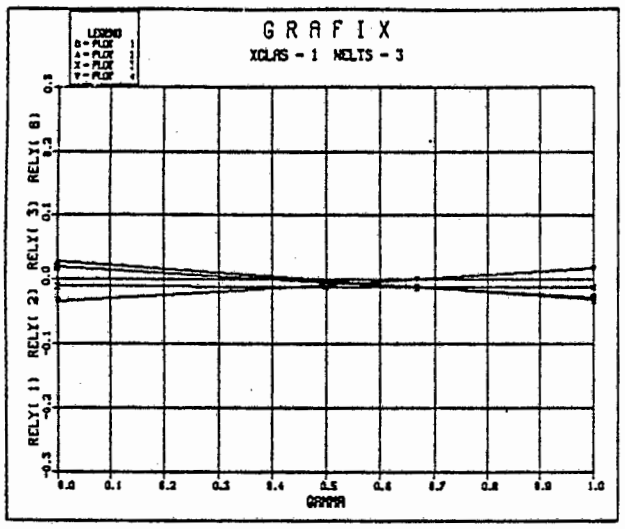


(ii) θ free, δ_H fixed (BCONDS=01)

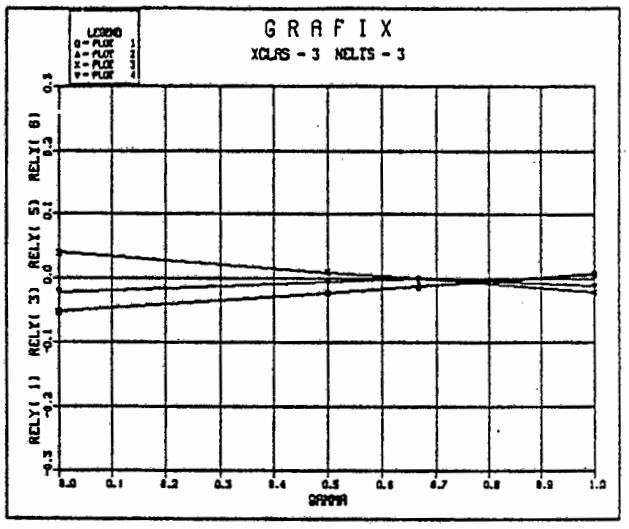


(iv) θ and δ_H fixed (BCONDS=11)

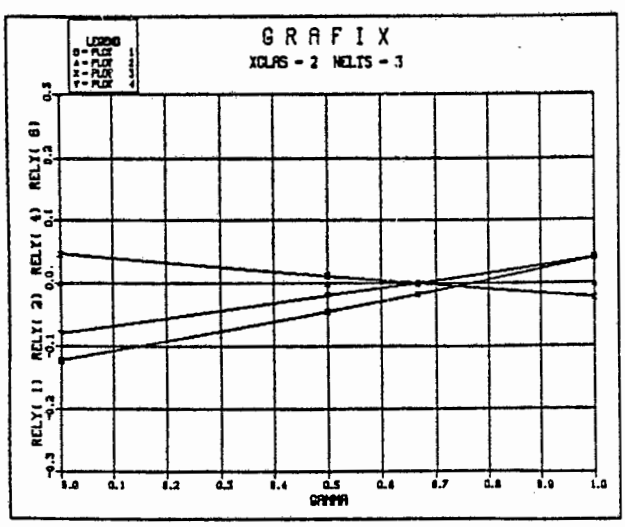
FIGURE 5.2.6(b) : γ versus RELY(i) for NELTS = 2



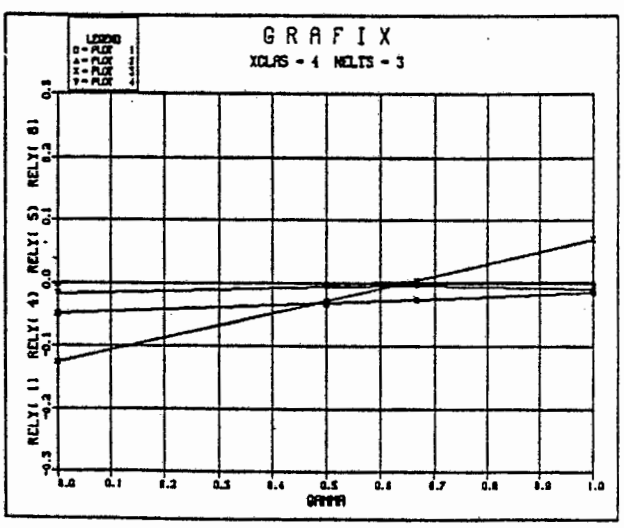
(i) θ and δ_H free (BCONDS=00)



(iii) θ fixed, δ_H free (BCONDS=10)



(ii) θ free, δ_H fixed (BCONDS=01)



(iv) θ and δ_H fixed (BCONDS=11)

FIGURE 5.2.6(c) : Υ VERSUS RELY(i) for NELTS = 3

These show clearly the almost linear dependence of each RELY measure on γ and the decrease in sensitivity with increasing NELTS. The fact that the lines tend to cross indicates that for each NELTS and BCONDS there is an optimum γ. Care must be taken in seeking this optimum γ because the scaling of each RELY measure implies a relative importance factor. Changing the importance factors would give different optima in this situation because the crossings do not occur at zero.

It was also already clear that for BCONDS=11 at least two elements were necessary.

To expand the database with respect to γ it was decided that instead of running many more cases on NLFRAM it was legitimate to interpolate RELY with respect to γ. For each of the NELTS and BCONDS settings, RELY(i) was interpolated for 151 points on $\gamma\varepsilon[0.5, 1.0]$. Figure 5.2.6 shows that all γ optima would lie in this interval. Then for each γ point the RELY data was summarised as follows:

$$\mu(\gamma,\text{NELTS}) = \max_{i,\text{BCONDS}} \left|\text{RELY}_i(\gamma,\text{NELTS},\text{BCONDS})\right|$$

except that for NELTS=1, BCONDS=11 is excluded.

This formula implies that all RELY measures are important and that we wish to have a rule applicable to all BCONDS, i.e. the whole of $\hat{X}^R$. The following Figure 5.2.7 shows μ parameterised by NELTS versus γ.

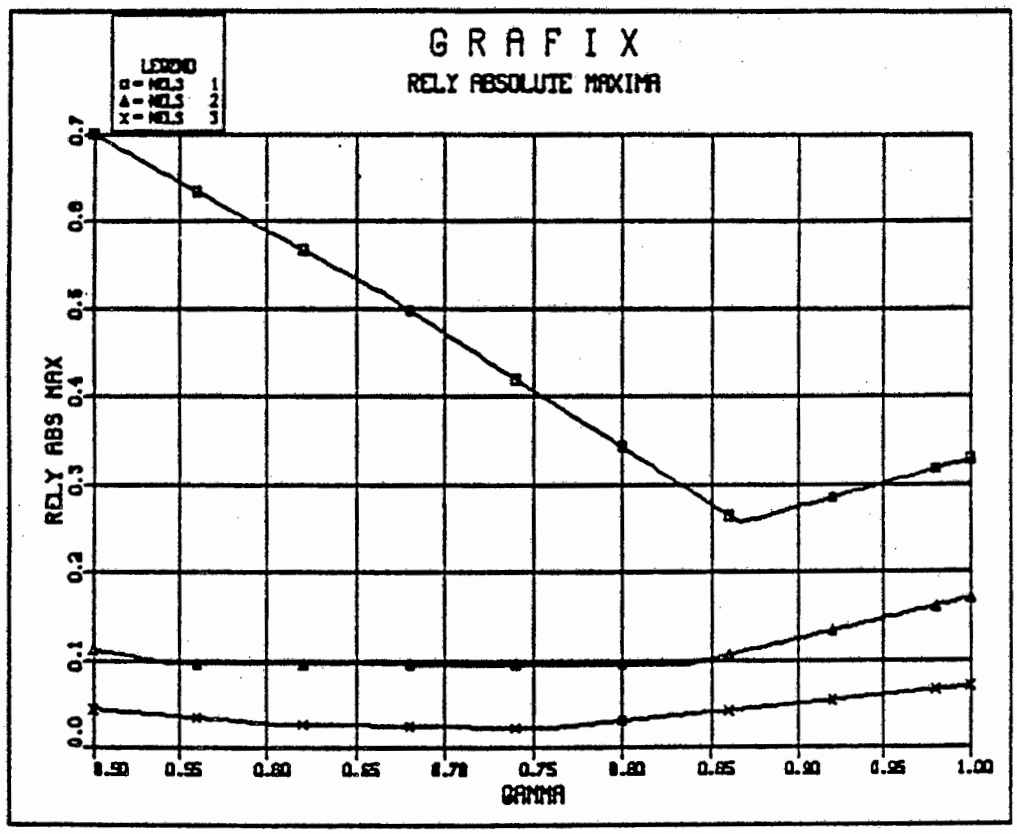


FIGURE 5.2.7 : Y versus μ with NELTS as a parameter (For NELTS = 1, BCONDS = 11 is excluded from μ calculation).

Applying a filter based strictly on the Pareto principle to all data points in μ-cost plane would correspond to taking the minimum points of each curve in the above figure. However, in most cases it is advantageous to have an idea of the sensitivity of the performance to the influencing variables. This may be achieved via an approximate Pareto filter, i.e. a filter which passes not only the optimum but also those points sufficiently close to it. (If the function were sufficiently smooth one could take derivatives). We could have used the filter in ELIXIR but, here, we simply read off the following rules to form the heuristic knowledge (i.e. induced $\hat{S}_K^R$ and formulated the rules).

KAS_4 and HK^{11}

HK^{11} expressed in terms of the image system is as follows.

Rule 1: If NELTS=1 and BCONDS≠ 11
then if $\gamma = 0.86$ then $\mu_{opt} = 0.26$
if $\gamma\varepsilon[0.85, 0.90]$ then $\mu\varepsilon[0.26, 0.27]$

Rule 2: If NELTS=2
then if $\gamma = 0.84$ then $\mu_{opt} = 0.098$
if $\gamma\varepsilon[0.55, 0.84]$ then $\mu\varepsilon[0.098, 0.010]$

Rule 3: If NELTS=3
then if $\gamma = 0.75$ then $\mu_{opt} = 0.023$
if $\gamma\varepsilon[0.59, 0.80]$ then $\mu\varepsilon[0.023, 0.030]$

Note that each rule above contains two subrules: one dealing with optimal control and the other with approximate optimal control. While a user would presumably select an optimal control, the information contained in the approximate optimal control gives a good idea of the sensitivity of the control near the optima.

In this example very little advantage is to be gained by expressing HK^{11} in terms of the object system, so it was not done.

The value of these rules may be judged from the figure below.

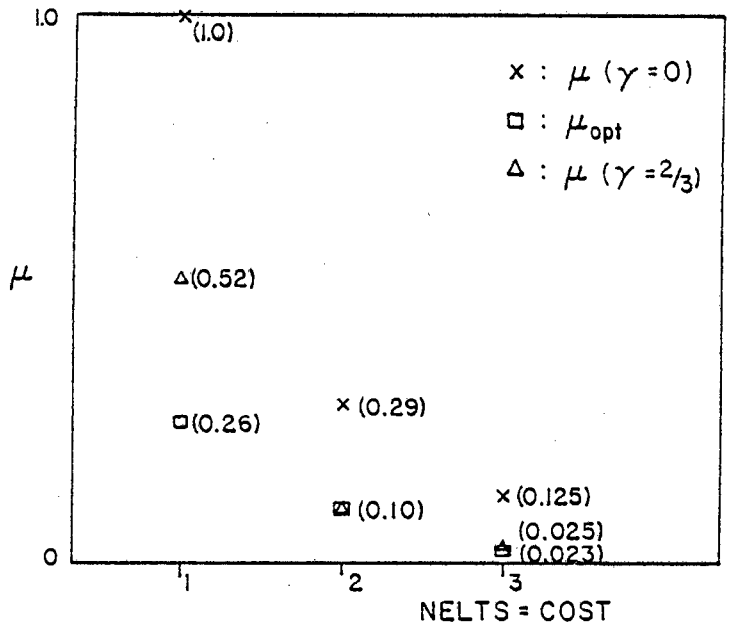


FIGURE 5.2.8 : The effect of 'optimal' γ on solution reliability

By looking back to Figure 5.2.6 one can determine which RELY measures and BCONDS are active at the μ_{opt}. One finds that BCONDS = 11 is nearly always the worst case and usually when RELY(1) and RELY(5) are active. This knowledge would be useful as a guide to someone who would refine the knowledge dependence on the boundary conditions.

The Integration Phase

In this example the justification of the heuristic knowledge follows directly from the experimental data analysis and the initial inituitive idea of using an offset with $\gamma = {}^{2}/_{3}$. However, we felt that in this instance it would be possible and worthwhile to find a theoretical explanation for this belief. Therefore, in parallel to the above knowledge acquisition process, a theoretical investigation was carried out. A stiffness matrix for a linear elastic, constant cross-section, circular arch element with arbitrary subtended angle ϕ_s was derived via the principle of complementary virtual work. The degrees of freedom for this stiffness matrix were the displacements at the ends, in directions parallel and perpendicular to the tangent of

the arch mid-surface, and the end rotations. The stiffness matrix was then transformed to relate to degrees of freedom parallel and perpendicular to the chord of the arch. Then by letting $R\phi_s \to \ell$, i.e. a limiting process from deep to shallow arches, and neglecting higher order terms, the following stiffness matrix results.

$$\begin{bmatrix}
\frac{EA}{\ell} & 0 & -\frac{2h}{3}\frac{EA}{\ell} & -\frac{EA}{\ell} & 0 & \frac{2h}{3}\frac{EA}{\ell} \\
 & \frac{12EI}{\ell^3} & \frac{6EI}{\ell^2} & 0 & -\frac{12EI}{\ell^3} & \frac{6EI}{\ell^2} \\
 & & \frac{4EI}{\ell}+\frac{4h^2}{9}\frac{EA}{\ell} & -\frac{2h}{3}\frac{EA}{\ell} & -\frac{6EI}{\ell^2} & \frac{2EI}{\ell}-\frac{4h^2}{9}\frac{EA}{\ell} \\
 & & & \frac{EA}{\ell} & 0 & -\frac{2h}{3}\frac{EA}{\ell} \\
 & & \text{symmetric} & & \frac{12EI}{\ell^3} & -\frac{6EI}{\ell^2} \\
 & & & & & \frac{4EI}{\ell}+\frac{4h^2}{9}\frac{EA}{\ell}
\end{bmatrix}$$

This is identical to the stiffness matrix of a straight element with an offset of 2h/3. Figure 5.2.8 shows that except for NELTS=1, using $\gamma=2/3$ results in μ very close to μ_{opt}. It is expected that $\mu_{opt} \to 2/3$ as NELTS$\to\infty$.

Generalisation of the HK

The heuristic knowledge HK[11] can quite easily be generalised to PC's with broader definition than PC[1]. If, instead of using NELTS as a measure of mesh density, one used the reciprocal of the angle subtended by a single element as a mesh density measurement, then conservative generalisation of HK[11] to problems with subtended angles (for the whole arch) other 90° would be straightforward. Incidentally, note how this generalisation illustrates the importance of separating the concept of mesh density from its measurement and hence the importance of defining object and image systems. By assuming that the HK is applicable to members defined between boundaries and/or point

loads, generalisations to more complex sets of point loads is also easy. Distributed loading conditions would require further knowledge acquisition but we would not expect the HK to change significantly. If the original straight element can deal with nonlinear materials then so will the offset element. However, more knowledge would need to be acquired because mesh densities (with or without offsets) would need to be increased. For geometrically nonlinear problems we believe that by including the mid-element transverse deflection in h (i.e. making h deflection dependent) the benefits would exceed those for linear analysis. Again, however, more knowledge would need to be acquired. Some generalisations to dynamic analysis and 3-D beams is also possible but these would also require further knowledge acquisition. In all generalisations or extensions of PC^1, the HK^{11} will serve as very valuable a priori knowledge.

CHAPTER 6 : CONCLUSION

Effective and efficient use of engineering software requires knowledge of the Pareto optimal set of the software. At present very little such knowledge is available to users. Complete knowledge of the Pareto optimal set is impossible to acquire due to the complexity of the problem and control spaces of the software and HK (Heuristic Knowledge) characterising an approximate Pareto optimal set must be acquired instead. Such HK may be acquired by making use of the knowledge acquisition procedure presented in Chapter 3. The procedure consists of three main phases, namely a planning phase in which a set of problem and control classes is defined, an execution phase in which HK for each problem and control class is acquired and an integration phase in which the separately acquired HK is explained and integrated into a single body of HK. The problem and control classes are simplifications of the problem and control spaces. The execution phase was further shown to consist of five main steps, namely, system definition, experimentation, data transformation, 'partitioning' and formulation of HK. These five steps correspond to the components of a KAS (knowledge acquisition system). While the planning and integration phases are labour intensive, i.e. they are done by the KAc (knowledge acquirer), by computerising most of the data management aspects of the KAS, the execution phase can be partially automated. Computerisation aspects were then discussed. Finally two examples of knowledge acquisition were given. These illustrated the use of the knowledge acquisition procedure and a computer-based KAS called ELIXIR. They also showed that, even for a fairly small software package, significant savings in computational effort may be achieved.

APPENDIX A SOME CLUSTERING TECHNIQUES

Let $X = \{x_i \mid i = 1,2,\ldots,n\}$ be a set of data points which are to be clustered or partitioned. For K hard clusters C_j, $j = 1,2,\ldots,K$

we have $\bigcup_{k}^{K} C_k = X$ and $C_i \cap C_j = \emptyset$ if $i \neq j$. (A1)

Let u be a K x n membership matrix which assigns the x_i to clusters C_k, i.e.

$$\text{if } x_i \in C_k \text{ then } u_{ki} = 1 \quad \text{else } u_{ki} = 0 \tag{A2}$$

For hard clustering, each x_i can only be assigned to one cluster but must be assigned to at least one. This is implied by (A1). Clearly this results in the restriction on u that one and only one element of each column in u must be 1, the others all being zero.

The number of possible K-partitionings on n points is given by [20] as

$$N_{Kn} = \frac{1}{K!} \sum_{j=1}^{K} (-1)^{K-j} \binom{k}{j} j^n \tag{A3}$$

N_{Kn} grows very quickly with increasing K and n. For example, for 5 clusters on 100 points, $N_{5,100} \approx 10^{68}$; $N_{3,15} = 2.3 \times 10^6$.

In general the points x_i may be vectors but consider for the moment the case where each x_i is a scalar. Also assume that the data are ordered, i.e.

$$x_1 \leq x_2 \leq \ldots\ldots \leq x_n, \tag{A4}$$

and that the elements of a cluster form contiguous sets, i.e. if a cluster begins with x_i and has ℓ points, then it consists of

$$\{x_i, x_{i+1}, \ldots, x_{i+\ell-1}\} \tag{A5}$$

Under these conditions (A3) becomes

$$N'_{Kn} = \binom{n-1}{K-1} \tag{A6}$$

$N'_{Kn} << N_{Kn}$, for example $N'_{3,15} = 91$. Compare this to $N_{3,15}$ above. A typical value expected in knowledge acquisition would be

$K = 4$, $n = 50$ in which case $N'_{4,50} = 230\ 300$.

Thus even here complete enumeration of the possibilities is probably uneconomical. Although dynamic programming can speed up the enumeration, Späth [20] suggests that, for solutions in reasonable time, one must use heuristic algorithms. He also points out that cluster algorithms are superior to visual classification only when the vectors x_i are one or two-dimensional. However, many clustering/partitioning problems in knowledge acquisition are likely to involve vectors of one or two dimensions in the clustering criteria so visual classification will often be adequate. This again stresses the importance of good graphical display of data. However the clustering algorithms may still be useful for organising the data because further processing will usually be done on this data. If it is organised properly, it will usually make such processing simpler and more efficient.

Now let $d_{ij} = d(x_i, x_j)$ be a distance function. Typically

$$d_{ij}^2 = \|x_i - x_j\|^2$$

where $\|\cdot\|$ is the Euclidean norm (assuming x is vector).

The mean or centroid $\bar{x}$ of a set of points X is that value $\bar{x} \in X^R$ (X is a sample of points in X^R)

$$\text{which minimises } \sum_i^n d^2(x_i, \bar{x}).$$

If d is based on the Euclidean norm

$$\bar{x} = \frac{1}{n} \sum x_i$$

Similarly one can define centroids for clusters as

$$\bar{x}_k = \{\bar{x} \mid \min_{\bar{x} \in X^R} \sum_{i=1}^n u_{ki} \, d^2(x_i, \bar{x})\} \tag{A7}$$

or for the Euclidean case

$$\bar{x}_k = \frac{1}{n_k} \sum_i^n u_{ki} x_i$$

where $n_k = \sum_i^n u_{ki}$

Many cluster algorithms are designed to minimise an objective function. The most popular of these is the sum of the sum of the squared distances of cluster members from their cluster centroids, i.e.

$$J(K) = \sum_{k=1}^K \sum_{i=1}^n u_{ki} \, d_{ik}^2 \tag{A8}$$

where $d_{ik} = d(x_i, \bar{x}_k)$

[Aside: The advantage of the form of (A8) and the use of the membership matrix u_{ki} is that it can easily be generalised to a fuzzy objective function

$$J_m(K) = \sum_{k=1}^K \sum_{i=1}^n (u_{ki})^m \, d_{ik}^2 \tag{A9}$$

where now $0 \leq u_{ki} \leq 1$. u_{ki} is called the membership grade of x_i in cluster k. The integer m defines the fuzziness of the clustering. The higher m the harder the partition. See [11]]

One of the simplest and most popular clustering algorithms, the KMEANS algorithm [12,20], which minimises J(K) (locally not globally - only complete enumeration can produce a global minimum) is as follows. Given an initial partition, one takes each x_i and re-assigns it to another cluster if, in so doing, J(K) is reduced. In fact one assigns it to the cluster which will reduce J(K) the most. Quite simple efficient formulae exist to test such conditions. This is repeated until no further reduction can be achieved. To check global optimality one will need to start the algorithm with a number of different initial partitions. One simple way to find an initial partition is to select K elements of X as an initial set of cluster centroids $\bar{x}_k$. In one or two dimensions this should be easy if based on visual information. Then simply assign the rest of the x_i to the cluster with nearest centroid.

A very serious problem with clustering is the selection of K. Clearly as $K \to n$, $\min J(K) \to 0$ but for abstraction purposes one requires a small K. Where clustering is used in knowledge acquisition, the selection of K should however be quite easy if visual inspection of the data is made. For simple HK one wants the minimum K which gives 'natural' clusters. Usually one can simply repeat the clustering for a few choices of K and then select the one which gives the 'best' result.

Another important problem in clustering is to choose an appropriate distance function. The Euclidean norm will usually give the most efficient algorithms with others often requiring considerable effort. Use of the L_∞ norm, i.e. $\max_\alpha |a_\alpha - b_\alpha|$ is apparently almost unusable according to [20]. Unfortunately minimisation of an objective function of the form

$$H(K) = \max_k (\max_i (u_{ki} d_{ik})) \qquad \text{(A10)}$$

may, in fact, be the most appropriate to knowledge acquisition. If the distance function measures distances between solution features $\hat{Y}$, then (A10) says that one wants a partition which minimises the worst error a user of P would make if he based his decisions on the HK derived from the partitioning.

In all of the above, we have considered only partitions of single level, that is to say that no further partitioning of partitions was desired. However, it will often be convenient to have such a double level with the first level being a partition of $\hat{X}^R$ and the second level an approximate partition of $\hat{Y}^R$. The process is more one of multi-objective function clustering rather than hierarchical clustering.

REFERENCES

Books on Artificial Intelligence/Expert Systems

1 NILSSON, N.J. 'Principles of Artificial Intelligence', Springer-Verlag, 1982.

2 PEARL, J. 'Heuristics: Intelligent Search Strategies for Computer Problem Solving', Addison-Wesley, 1984.

3 BARR, A. and FEIGENBAUM, E.A. (Eds) 'The Handbook of Artificial Intelligence', Vol. I & II. , Pitman, 1982.

4 COHEN, P.R. and FEIGENBAUM, E.A. (Eds) 'The Handbook of Artifical Intelligence', Vol. III, Pitman, 1982.

5 WATERMAN, D.A. 'A Guide to Expert Systems', Addison-Wesley, 1986.

6 HAYES-ROTH, F. WATERMAN, D.A. and LENAT, D.B. (Eds) 'Building Expert Systems', Addison-Wesley, 1983.

7 BRATKO, I. 'Prolog Programming for Artificial Intelligence', Addison-Wesley, 1986.

8 HARMON, P and KING, D. 'Expert Systems - Artificial Intelligence in Business', John Wiley & Sons, Inc., 1985.

9 WATANABE, S. 'Pattern Recognition : Human and Mechanical', John Wiley and Sons, 1985

Books on Modelling/Mathematics/Statistics

10 KANDEL, A. 'Fuzzy Mathematical Techniques with Applications', Addison-Wesley, 1986.

11 BEZDEK, J.C. 'Pattern Recognition with Fuzzy Objective Function Algorithms', Plenum Press, 1981.

12 ANDERBERG, M.R. 'Cluster Analysis for Applications', Academic Press, 1973.

13 MYERS, R.H. 'Response Surface Methodology', Allyn and Bacon, Inc., 1971.

14 DAS, M.N. and GIRI, N.C. 'Design and Analysis of Experiments', Wiley Eastern Limited, 1979.

15 KLEIJNEN, J.P.C. 'Statistical Techniques in Simulation, Part I', Marcel Dekker, Inc., 1974.

16 KLEIJNEN, J.P.C. 'Statistical Techniques in Simulation, Part II,'Marcel Dekker, Inc., 1975.

17 SZÜCS, E. 'Similitude and Modelling', Elsevier Scientific Publishing Co., 1980.

18 TAHA, H.A. 'Operations Research - An Introduction', Third Edition, Macmillan Publishing Co., 1982.

19 COOK, T.M. and RUSSELL, R.A. 'Introduction to Management Science', Second Edition, Prentice Hall, 1981.

20 SPäTH, H. 'Cluster Analysis Algorithms for data reduction and classification of objects', Ellis Horwood, 1980.

21 MEES, A.I. 'Dynamics of Feedback Systems', John Wiley & Sons, 1981.

Books on Databases

22 DATE, C.J. 'An Introduction to Database Systems', Addison-Wesley, Third Edition, 1981.

23 MARTIN, J. 'Principles of Data-Base Management', Prentice-Hall, 1976.

24 MARTIN, J. 'Managing the Data-Base Environment', Prentice-Hall, 1983.

25 WEIDERHOLD, G. 'Database Design', McGraw-Hill, 1983.

Books on FEA/Numerical Analysis

26 National Agency for Finite Element Methods and Standards (Great Britain) 'Guidelines to Finite Element Practice', 1984.

Books on Philosophy/Current Affairs

27 KUHN, T.S. 'The Structure of Scientific Revolutions', The University of Chicago Press, 1962, 1970.

28 FEYERABEND, P.K. 'Against Method', Verso, 1978.

29 FEYERABEND, P.K. 'Science in a Free Society', Verso, 1982.

30 FEYERABEND, P.K. 'Realism, Rationalism and Scientific Method', (Philosophical Papers, Vol. I), Cambridge University Press, 1981.

31 FEYERABEND, P.K. 'Problems of Empiricism', (Philosophical Papers, Vol. II), Cambridge University Press, 1981.

32 ABERCROMBIE, M.L.J. 'The Anatomy of Judgement', Penguin Books, 1960.

33 ARNHEIM, R. 'Visual Thinking', University of California Press, 1969.

34 MICHIE, M. AND JOHNSTON, R. 'The Creative Computer', Viking, 1984.

35 LAKATOS, I. 'Proofs and Refutations', Cambridge University Press, 1976.

36 LAKATOS, I. and MUSGRAVE, A. (Eds) 'Criticism and the Growth of Knowledge', Cambridge University Press, 1970.

37 NEWTON-SMITH, W.H. 'The Rationality of Science', Rontledge & Kegan Paul, 1981.

38 SUNTER, C. 'The World and South Africa in the 1990's', Human & Rousseau Tafelberg, 1987.

Papers on Artificial Intelligence/Expert Systems

39 TAIG, I.C. 'Expert Systems and Finite Element Analysis, Part I Personal Experiences from the FEASA Project', Finite Element News, No. 4, Aug. 1986, 28-32.

40 TAIG, I.C. 'Expert Systems and Finite Element Analysis, Part II Some Thoughts on Developing Practical Systems', Finite Element News, No. 6, Dec. 1986, 26-30.

41 WIELINGA, B.J. and BREUKER, J.A. 'Interpretation of Verbal Data for Knowledge Acquisition' in 'Advances in Artificial Intelligence', O'SHEA, T. (Ed), North-Holland, 1985.

42 BOOSE, J.H. 'A Knowledge Acquisition Program for Expert Systems based on Personal Construct Theory', Int. J.of Man-Machine Studies, 23, 1985, 495-525.

43 COOKE, N.M. and MACDONALD, J.E., 'A Formal Methodology for Acquiring and Representing Expert Knowledge', Proceedings of the IEEE, Vol. 74, No. 10, Oct. 1986, 1422-1430.

44 MACCALLUM, K.J. and DUFFY, A. 'An Expert System for Preliminary Numerical Design Modelling', Adv. Eng. Software, Vol. 8, No. 4, 1986, 217-222.

45 DYM, C.L. 'Expert Systems: New Approaches to Computer-Aided Engineering', Engineering with Computers, 1, 1985, 9-25.

46 SRIRAM, D., MAHER, M.L. and FENVES, S.J. 'Knowledge-Based Expert Systems in Structural Design', Computers and Structures, Vol. 20, No 1-3, 1985, 1-9.

47 RISSLAND, E.L. 'The Ubiquitous Dialectic' in 'Advances in Artificial Intelligence', O'SHEA, T. (Ed), North-Holland, 1985.

48 ARORA, J.S. and BAENZIGER, G. 'Uses of Artificial Intelligence in Design Optimisation', Comp. Meth. Appl. Mech. Eng. 54, 1986, 303-323.

49 BAENZINGER, G and ARORA, J.S. 'Development of an Artificially Intelligent Nonlinear Optimisation Expert System', AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference, Texas, May 1986.

50 MITCHELL, T.M., KELLER, R.M. and KEDAR-CABELLI, S.T. 'Explanation-Based Generalisation : A Unifying View, Machine Learning 1, 1986, 47-80.

Papers on Software/Computer Science

51 BELL, K. 'Some Thoughts on Design, Development and Maintenance of Engineering Software', Adv. Eng. Software, Vol. 8, No. 2, 1986, 66-72.

52 BENNET, J.L. 'Tools for Building Advanced User Interfaces', IBM Systems Journal, Vol. 25, No's 3/4, 1986, 354-368.

53 GOLDBERG, R. 'Software Engineering : An Emerging Discipline', IBM Systems Journal, Vol. 25, No's 3/4, 1986, 334-353.

Papers on Modelling/Maths/Statistics

54 PESCHEL, M. 'Engineering Cybernetics' in Cybernetics - Theory and Applications', TRAPPL, R. (Ed), Hemisphere Publishing Co., 1983.

55 KLIR, G.J. 'General Systems Theory' in 'Cybernetics - Theory and Applications', TRAPPL, R. (Ed), Hemisphere Publishing Co., 1983.

56 BALCI, O. and NANCE, R.E. 'Formulated Problem Verification as an Explicit Requirement of Model Credibility', Simulation, 45:2, Aug. 1985, 76-86.

57 GASS, S.I. 'What is a computer-based mathematical model?', Mathematical Modelling, Vol. 4, No. 5, 467-472.

58 GASS, S.I. 'Concepts of Model Confidence', Comput. & Ops. Res., Vol. 8, No. 4, 1981, 341-346.

59 GASS, S.I. 'Decision aiding models : validation, assessment and related issues for policy analysis', Operations Research, Vol. 31, No. 4, July-Aug. 1983, 603-631.

60 GASS, S.I. 'Evaluating Complex Models', Comput. & Ops. Res., Vol. 4, 1977, 27-35.

Papers on Databases

61 BENAYOUNE, M. and PREECE, P.E. 'Methodology for the Design of Databases for Engineering Applications', Computer-Aided Design, Vol. 18, No. 5, June 1986.

62 MURTHY, T.S., SHYY, Y-K and ARORA, J.S. 'MIDAS : Management of Information for Design and Analysis of Systems', Adv. Eng. Software, 1986, Vol. 8, No. 3, 149-158.

63 ARORA, J.S., LEE, H.H. and YAO, S.Y. 'SMART : Scientific database Management and engineering Analysis Routines and Tools', Adv. Eng. Software, 1986, Vol. 8, No. 4, 194-199.

64 FELIPPA, C.A. 'Database Management in Scientific Computing -I. General Description', Computers & Structures, Vol. 10, 1979, 53-61.

65 FELIPPA, C.A. 'Database Management in Scientific Computing -II. Data Structures and Program Architecture', Computers & Structures, Vol. 12, 1980, 131-145.

66 FELIPPA, C.A. 'FORTRAN-77 Simulation of Word Addressable Files', Adv. Eng. Software, Vol. 4, No. 4, 1982, 156-162.

67 MURTHY, T.S. and ARORA, J.S. 'Database Design Methodology for Structural Analysis and Design Optimisation', Engineering with Computers 1, 1986, 149-160.

68 WIEDERHOLD, G. 'Knowledge and database management' IEEE Software, Vol. 1, No. 1, Jan. 1984, 63-73.

Papers on FEA/Numerical Analysis

69 HIBBIT, H.D. and BASHYAM, G.R. 'Some Issues Associated with the Validation of Finite Element Analysis', Finite Elements in Analysis and Design, 2, 1986, 119-124.

70 MACNEAL, R.H. and HARDER, R.L. 'A proposed Set of Problems to Test Finite Element Accuracy', Finite Elements in Analysis and Design, 1, 1985, 3-20.

71 STOLARSKI, H and BELYTSCHKO, T. 'Membrane Locking and Reduced Integration for Curved Elements'. J.Appl. Mech., Vol. 49,. March 1982, 172-176.

72 JETTEUR, P.H., CESCOTTO, S., DE VILLE DE GOYET, V. and FREY, F. 'Improved Nonlinear Finite Elements for Oriented Bodies using an Extension of Marguerre's Theory', Computers & Structures, Vol. 17, No. 1, 1983, 129-137.

73 OLESEN, J.F. 'Field Redistribution in Finite Elements - A Mathematical Alternative to Reduced Integration', Computers & Structures, Vol. 17, No. 2, 1983, 157-159.

74 DE VILLE DE GOYET, V and FREY, F. 'Use of Marguerre Theory in the Nonlinear Analysis of Beam and Plate Structures' in 'Accuracy, Reliability and Training in FEM Technology', see Book References.

75 NOOR, A.K. and PETERS, J.M. 'Penalty Finite Element Formulation for Curved Elastica', J. Eng. Mech., Vol. 110, No. 5, May 1984, 694-712.

76 MACNEAL, R.H. 'The Reliability of Finite Element Tools', Finite Elements in Analysis and Design 2, 1986, 249-257.

77 MAIR, W.M. 'The Objectives of the National Agency for Finite Element Methods and Standards', Computers & Structures, Vol. 21, No. 5, 1985, 875-879.

78 ROUSE, N.E. 'FEA for the Masses', Machine Design, July 25, 1985, 62-68.

79 HAWLA, D.L. 'NLFRAM User's Manual', NIAST 84/85 Report, July 1984.

80 CHENG, S-I. 'Asymptotic Behaviour and Best Approximation in Computational Fluid Dynamics', Mathematics and Computers in Simulation, XXIV, Feb. 1982, 37-48.