

City Research Online

City St George's, University of London

Citation: Goker, A. S. (1994). An investigation into the application of machine learning in information retrieval. (Unpublished Doctoral thesis, City, University of London)

This is the accepted version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

Permanent repository link: https://openaccess.city.ac.uk/id/eprint/29402/

Copyright and Reuse: Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to City Research Online policy.

City Research Online: http://openaccess.city.ac.uk/ publications@citystgeorges.ac.uk

An Investigation into the Application of Machine Learning in Information Retrieval

Ayşe Safiye Göker
Department of Information Science
City University
London

May, 1994

This thesis is submitted as part of the requirements for a Ph.D in Information Science in the Department of Information Science, City University, London, England.

Table of Contents

Acknowledgements i
Abstract iii
Abbreviations iv

Chapter 1

Introduction 1

Chapter 2

Developments Toward Intelligent Information Retrieval 4
2.1 A Perspective on Information Retrieval and Context 5
2.1.1 Knowledge, Information and Data 5
2.1.2 Artificial Intelligence and Information Retrieval 8
2.1.3 Models of/in Information Retrieval 11
2.1.4 The Scope and Meaning of Context 13
2.1.5 Intelligent Information Retrieval 21
2.2 Modelling the User and System 21
2.2.1 Dynamic User Models 21
2.2.2 Static User Models 22
2.3 Classification, Categorisation and Clustering 24
2.4 Expanding the Role of the Thesaurus 26
2.5 Organising the Concepts in Documents 29
2.6 Query Formulation, Expansion and Relevance Feedback 31
2.7 Natural Language Processing 36
2.8 Improving the User Interface 38
2.9 Other Methods 40

Chapter 3

Machine Learning 41
3.1 A Human-oriented Analysis of Learning 42
3.1.1 Learning by Rote 42
3.1.2 Learning by Being Told 44
3.1.3 Learning by Example 45
3.1.4 Learning by Analogy 52
3.1.5 Learning by Observation and Discovery 54
3.2 A Computation-oriented Analysis of Learning 56
3.2.1 Data and Model Driven Learners 56
3.2.2 Improving System Speed, Scope and Quality 59
3.2.3 Sub-symbolic Learning 59
3.3 Evaluation in Learning 62

Chapter 4

Learning in Information Retrieval 64
4.1 Designing Intelligent Information Retrieval Systems 64
4.2 Why some Machine Learning Applications are more suitable to Information Retrieval 66
4.3 Sources for Learning in Information Retrieval 67
4.4 The Basis for the Current Investigation 69

Chapter 5

The Context Learner 71
5.1 Description of Context 71
5.2 Use of Context 73
5.3 Inputs to the Learner 75
5.4 Description Language 75
5.4.1 The Instance Language 75
5.4.2 The Context Description Language 76
5.5 Examples of Context Learning 76
5.6 Modules and Tests 83
5.6.1 The Set of Relevant Documents 85
5.6.2 Filtering the Terms 87
5.6.3 Merging with Old Contexts 93
5.6.4 The Tests 96

Chapter 6

The Role of a Learner in Okapi 102
6.1 Description of Okapi 102
6.1.1 Installation and Access 103
6.1.2 System Use 104
6.1.3 The Retrieval Process 104
6.2 Data and Information Sources 107
6.3 Analysis of User Data 109
6.3.1 Pattern and Context of Frequent Users 109
6.3.2 Interviews to Expand Background Information 110
6.3.3 Terms Used by Individuals 111
6.3.4 Variety and Frequency of Terms Used 111
6.3.5 User Queries 111
6.4 Results of User Data Analysis 112
6.4.1 Pattern and Context 112
6.4.2 Interviews to Expand Background Information 113
6.4.3 Terms Used by Individals 114
6.4.4 Variety and Frequency of Terms 114
6.4.5 Query Analysis 114
6.5 Deficiencies and Possible Enhancements of Okapi 122
6.6 Enhancements Involving Learning in Okapi 123

6.6.1 Phrase Identification 124
6.6.2 Dealing with Homonyms 126
6.6.3 Subject/Context Linkage and Identification 128
6.6.4 Dealing with Weight-Blocks 130

Chapter 7

Application of the Context Learner to Okapi 131
7.1 Implementing the Context Learner 131
7.1.1 The Process for Gathering Input to the Learner 131
7.1.2 The Output from the Learner 132
7.1.3 Integrating the Learner with Okapi 132
7.1.4 Possible Variants 133
7.1.5 Implementing the Variations 134
7.1.6 Problems in Implementation 135
7.2 Evaluation in Information Retrieval 136
7.2.1 The Nature of Evaluation in Information Retrieval 136
7.2.2 Relevance 139
7.2.3 Measures for Evaluation 140
7.2.4 Test Collections 143
7.2.5 User-oriented Evaluation 144
7.3 Design of the Experiments 145
7.3.1 Experiment 1: Deciding which Learning Algorithm 145
7.3.2 Experiment 2: Does the Algorithm Improve Document Ranking 147
7.4 Operational Conditions of the Experiments 148
7.4.1 The System 148
7.4.2 The Users 149
7.4.3 The Procedure 149
7.4.4 Evaluation Measures 151
7.5 Results and Evaluation 151
7.5.1 Results (Experiment 1) and Analysis 151
7.5.2 Results (Experiment 1) and Analysis 158
7.5.3 Problems and Weaknesses 159
7.5.4 Overall Interpretations 160

Chapter 8

Conclusions 162

Appendices

Appendix A - Okapi System Description 166
Appendix B - Okapi Transaction Logs 178
Appendix C - Assessment of Frequent Users' Logs 184
Appendix D - Frequent User Session Summary Diagram - An Example 206
Appendix E - Term Details for Frequent Users - An Example 209
Appendix F - Term Usage - An Example 211
Appendix G - Explanation-based Learning - An Example 213
Appendix H - Evaluation of Offline Prints: Relevance Assessment Instructions 215
Appendix I - Number of Online Queries and Sessions: Evaluation 1 216
Appendix J - Number of Online Queries and Sessions: Evaluation 2 217
Appendix K - Queries: Evaluation 1 218
Appendix L - Queries: Evaluation 2 219
Appendix M - Relevance Judgement Results : Evaluation 1 220
Appendix N - Relevance Judgement Results : Evaluation 2 228
Appendix O - Precision Values at Various Cut-off Points: Evaluation 1 232
Appendix P - Precision Values at Various Cut-off Points: Evaluation 2 236
Appendix Q - Example of Relevant Documents for Queries 238

References 241

List of Figures

Fig.2.1: The entities in an Information Retrieval situation. Diagram partly based on that in Robertson (1979). 13
Fig.2.2: Control and Data Flow in the IRS 15
Fig.2.3: The mapping between queries and documents chosen relevant. 19
Fig.2.4: Order of queries and 'contexts' generated 20
Fig.2.5: A classification of retrieval techniques 31
Fig.3.1: Machine learning strategies. 43
Fig.3.2: Optimal tree diagram produced by ID3. 48
Fig.3.3: Analogical reasoning in geometry. 53
Fig.4.1: An example connection between documents, terms, queries and users. 69
Fig.5.1: The retrived document lists showing the alternative ways of changing document ordering. 73
Fig.5.2: The stages of using the relevant documents including their index terms (R), potential merge terms (M) and old/last context terms (L) for approximating the context. 85
Fig.5.3: The possible ways of forming the set of relevant documents (R). 86
Fig.5.4: Examples of documents and the terms that cover them. 91
Fig.5.5: The possibilities for a term which was in the previous 'context' and has now reoccured. 95
Fig.7.1: The components involved in applying the context learner to the Okapi IRS. . . . 133

List of Tables

Table.3.1: Examples for classification, taken from Luger and Stubblefield (1989). 48
Table.5.1: Table showing the query and context (Used and Hold) terms for e.g.1. 78
Table.5.2: The relevant document and their (stemmed) index terms for query q(1) in e.g.1. 79
Table.5.3: The relevant document and their (stemmed) index terms for queries q(1), q(2) in e.g.2. 80
Table.5.4: Table showing the used context terms for e.g.2, when using the context learner of test T2. 81
Table.5.5: Table showing the used context terms for e.g.2, when using the context learner of test T12. 82
Table.5.6: Table showing the used context terms for e.g.2, when using the context learner of test T13. 82
Table.5.7: Table showing the used context terms for e.g.2, when using the context learner of test T15. 83
Table.5.8: Table showing the tests performed with their corresponding modules. 97
Table.5.9: Summary table of the modules and variables for the Context Learner. 98
Table.6.1: The number of <New>/<Edit> sessions and queries. 115
Table.6.2: The number of potential and genuine queries for <New> and <Edit> searches. 116
Table.6.3: <Edit> and <New> Searches for the three categories of syntactic changes. 119
Table.6.4: <Edit> and <New> Searches for the three categories of semantic changes. 120
Table.6.5: Percentage of <Edit>ed and <New> Searches for the mappings made between the syntactic changes in a query and any subsequent semantic changes. 121
Table.7.1: The 2x2 contingency table (Cleverdon, 1967). 141
Table.7.2: The range of percentage values considered for each rank position for the two types of precision values, for Experiment 1. 152
Table.7.3: Precision values at various cut-off points for Context Learner L3 for Experiment 1. 153
Table.7.4: The Context Learners with precision in the top 5%. 153
Table.7.5: The modules for the Context Learners used in Experiment 2. 154
Table.7.6: Summary table of the modules and variables concerning the Context Learner versions for Experiment 2. 155
Table.7.7: The precision values at various cut-off points for the plain Okapi system, for Experiment 2. 158
Table.7.8: The precision values for context learner L18 for Experiment 2. 159

Acknowledgements

Professor Stephen Robertson, for his supervision, patience and insight in this work.

Professor Lee McCluskey, for his supervision, advice and enthusiasm.

Stephen Walker, for helping with my queries and problems with the Okapi Information Retrieval System and also for his humour.

Susan Jones, for all the tedious proof reading and support.

The Department of Information Science Administration Staff, for resolving all 'studently' needs very efficiently and sincerely.

The British Council, for financial support for part of the duration of this work.

Aarron Gull (Dr. Gull now), for providing different arguments and outlooks during this work and coming to the technical rescue.

Robin Swain (also another Dr. now), for willingly sharing his knowledge of having been down this path before and for his patience with my diagrams.

Jackie Naughton, for being such a responsible friend and checking up on my progress regularly.

Interestingly, those in A215 who provided their mutual help, comfort and experience on matters non-Ph.d.

My family, for all the support and never turning my telephone calls down however long the distance, however unearthly the time.

This work is dedicated to my sister, Selma, who I almost lost during this period.

Declaration of Copyright

I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

Abstract

There is an increasing variety of online databases available which are also evergrowing in size. In retrieving information from these sources, it is important not only to have effective and efficient retrieval techniques but also to enable some form of adaptation to users' specific needs. Frequent users, in particular, should be able to benefit from their high use of the *information retrieval system*. A *machine learning* approach can be applied to help the system adapt to users' specific needs.

It is argued that users have a particular *context* within which their queries are formed. It is likely that consecutive queries for a particular user will be related in that they will be part of the same *context*. Thus, a *context learner* is proposed.

In this investigation, the *context learner* is used for enhancing document ordering in partial match systems.

Abbreviations

AI	Artificial Intelligence
CL	Context Learner
IR	Information Retrieval
IIR	Intelligent Information Retrieval
IRS	Information Retrieval System
LAN	Local Area Network
LISA	Library and Information Science Abstracts
ML	Machine Learning
OPAC	Online Public Access Catalogue

Chapter 1

Introduction

The use of Artificial Intelligence (AI) for Information Retrieval (IR) has been debated since the early 70s and various approaches attempted. Particular emphasis has been in the area of natural language processing, expert systems, user modelling and document classifications. This thesis investigates the applicability of machine learning (ML), as a branch of AI, to IR. In doing this, a survey of the developments of both fields is presented with a view to how they may be combined. The fact that this thesis concentrates on *learning* as viewed from an AI perspective does not mean that it did not exist previously in IR. A number of statistical retrieval techniques do contain a form of learning. However, the work here involves the application of *symbolic learning.*

The purpose of IR is to help meet an information need, at a particular point in time, of a user. The argument in this thesis is that each information need has an associated *context.* Additionally, often a number of information needs will have a *common context.* Hence, the argument is that what may be learnt from meeting one information need may be of use for further information needs. Although the work here applies to the information need and context of individual users, there is also scope for applying this idea over groups of users.

The report of this thesis can be viewed in three parts: the first consisting of the background work in the fields of ML and IR (chapters 2-4); the second forming the theoretical part of the work (chapter 5); and the final being the practical part of implementing the theory on an experimental system and its evaluation using real user queries (chapters 6-7).

The work demonstrates how machine learning can be applied to IR through the assimilation of contextual knowledge by using *context* to help users with their subsequent queries. For this, the Okapi system, based on a probabilistic model of IR was used. Various themes for representing user *contexts* were tested and evaluated. Variations included the size of the *context* and rate of change of the terms in it and whether the context remained constant or dynamic.

The tests were carried out on a group of frequent users. The experiments identified which of the approaches to identifying/defining context performed better and whether it was an improvement on the existing retrieval system.

The following seven chapters present all aspects of the investigation carried out into the application of machine learning in information retrieval.

Chapter 2 gives a perspective of IR more suited towards applying ML to IR. The concepts and meaning of *information*, *knowledge*, *data* are described with a view to introducing the idea of *context*. Developments that are seen to be towards intelligent IR are surveyed and these are classified according to the way in which they may help retrieval.

In chapter 3, the currently available *learning* techniques are surveyed with examples, along with discussions of their ease of applicability to IR. The intention is to provide an understanding of the nature of work done by ML researchers and their applicability to IR. However, the theoretical reasoning behind the techniques are not discussed in their full logic detail. The machine learning techniques are viewed in two ways: a *human-oriented perspective* analogous to the ways in which humans learn; and a *computation-oriented analysis*.

The purpose of the introductory chapters is to provide the background in the fields of IR and ML. However, the classification made in Chapters 2 and 3 should not be interpreted too rigidly. Later, in Chapter 4, discussions regarding the nature and applicability and methods of applying ML to IR are discussed. This is done by referring to the nature of the problems, the possible sources for learning in an IRS, and some design considerations.

Chapter 5 describes and defines the theory of the *context learner* as a form of symbolic learning in helping IR and how it can be used. The possible variations are described and examples are given.

Subsequently, Chapter 6 focuses on the application of the learner on the Okapi IR system. This system and the role of the learner in it are described. Preliminary analysis of frequent users of the system and the nature of their queries is also presented. The analysis performed helped identify deficiencies of the system along with possible enhancements, although the emphasis is on those that could be improved using learning techniques. The particular problem addressed is the *weight-block* problem in which the ordering of the references shown to the user can be improved.

Chapter 7 consists of the application of the context learner, as defined in Chapter 5, to the current Okapi system. The implementation details of the various options in the context learner are discussed along with the problems in implementation. Two further experiments constituting evaluation are described with reference to their operational conditions, problems, weaknesses and results. The purpose of the first experiment was to narrow the context learner options/variations.

The second experiment was peformed on the remaining context learner versions in order to see if there was any improvement on the original system.

Chapter 8 identifies areas in which the investigation can be furthered, in addition to the conclusions that were drawn from it.

Chapter 2

Developments Toward Intelligent Information Retrieval

This chapter consists of background work for the thesis. In the first section, there is some discussion relating to the meaning of information, knowledge, data and models of/in information retrieval (IR). Following this there is an introduction into the scope and meaning of *context*, as referred to in this work. The term relates to the context of information need that users have when initiating a query on an information retrieval system (IRS).

The first section consisting of a "perspective of information retrieval and context" is intended to be a cursory summary providing a forum for deeper discussions. After this section, the developments in the field of Information Retrieval (IR) that can be seen to be "intelligent" are categorised and reviewed. The problem with heading what is intelligent IR is that although a system may appear to be "intelligent" it may not be based on an Artificial Intelligence (AI) standpoint nor is it necessarily a candidate for applying AI techniques. Nevertheless, their results show improvement in system quality and performance and indicate a level of "intelligence".

The review, in this chapter, is selective. To describe all the approaches, techniques and systems used in IR is not the scope of this thesis and others have done this in many different ways (e.g. Daniels (1986) on user modeling in IR, Robertson (1977) on theories and models in IR). Thus, the review focuses on the following two points:

- what is an advance on the practical state of the art, and
- the system that is perceived by a user as being clever or knowledgeable.

The classification given, in this chapter, is more to do with application areas in IR rather than the nature of the specific techniques employed. The work in the field does not always fall into tidy discrete units and so the classification should not be interpreted too rigidly.

In this work, the approaches have been categorised in the following way:

- modelling the user and system
- classification, categorisation and clustering
- expanding the role of the thesaurus
- organising the concepts in documents
- helping query formulation, expansion and relevance feedback
- natural language processing

- improving the user interface
- other methods

However, as mentioned above, a system referred to under one category can also be viewed in another. For example, a system containing an enhanced thesaurus may also use user models in helping with query formulations.

Evaluation of the techniques used is important and results are referred to, in this chapter, as and when they are appropriate. However, the purpose of this work is not the evaluation of IR techniques, although some further discussion is included in Chapter 7 as part of the nature of evaluation in the field.

2.1 A Perspective on Information Retrieval and Context

This section briefly discusses the meaning of knowledge, information and data, prior to addressing the connections and differences between the fields of AI and IR. This is followed by a sub-section on the types of models derived in/of IR in order to show how the idea of *context* relates to the retrieval process. Irrespective of its suitability for applying Machine Learning (ML) to IR, its validity is a separate argument of this thesis. Thus, its scope and meaning is discussed in section 2.1.4.

2.1.1 Knowledge, Information and Data

Early philosphers, in their quest to understand our environment and human nature, have often concentrated on the meaning and definition of *knowledge* and *information*. Perhaps the earliest of these was Plato (Russell, 1946) who attempted many definitions for knowledge and eventually seemed to equate it with perception. His pupil, Aristotle, related knowledge more to the mind as he saw it. According to Aristotle, the mind is a process and function working by association and not just a substance. The mind is a dynamic structure which is shaped and organised through forms and experiences of the world. This means that information which is useful in creating knowledge is organised or structured so that some association can take place. This is a more common sense approach to the Platonic view (Titus, 1953; Durham, 1989).

Descartes, perhaps taking the practical view to an extreme, enquired into everything he knew in order to identify what could be accepted as reliable knowledge. According to this view, if there is any reason to doubt then the entire category should be treated as doubtful and unreliable (although it is arguable what a category may be and what its delimiters are). With this quest for certainty, putting knowledge as derived from our senses to his test, he emerged or perhaps was left with his most famous conviction that *he existed because he thought* (Popkin et. al., 1969).

To later philosophers such as Hume, who tended to hesitate accepting anything beyond our daily experiences, the mind is nothing but an association of ideas. The mind is a collection of experiences and sensations and all knowledge comes through experience (Titus, 1953).

In what he called Critical Philosophy (a kind of scepticism), for example, Kant undertook investigations into the foundations of knowledge where his main emphasis was that whatever is

referred to as knowledge must justify itself as such. This should entail what kind of knowledge it is, indeed in what sense it is knowledge and what its scope and degree of certainty is (Randall and Buchler, 1957).

So far, the nature and characteristics of knowledge has been briefly discussed but it is also necessary to look at how it is connected to information and data. Before pursuing the meaning information and its connection to knowledge, it is useful to mention data and how it fits in with knowledge and information.

Data, according to Langefors and Sundgren (1975), are sets of symbols representing information or knowledge. One could argue whether knowledge is derived from data or data formed to represent existing knowledge. In Langefors and Sundgren's view, data could be interpreted as information or as knowledge. As *information*, it is a "**new** fact, model, procedure or goal interpreted in terms of previous facts, models, procedures or goals". As *knowledge*, however, it is the same as a "**previously acquired** fact, model, procedure or goal". The view in this thesis is one that relates to the path of data-information-knowledge, in that information can be derived from data and that knowledge can be derived from information. However, what is meant by information?

Once again refering to an early philosopher, such as Socrates, it is clear that people have contended with this definition for a long time. There are, according to Socrates' account, two main types of information that we can possess - *visible/sensible* (those which are acquired through the senses) and *intelligible*. The visible or sensible information includes images and opinions but none of this constitutes knowledge, 'because none of it is indubitable'. The intelligible information, on the other hand, deals with ideas (Platonic Ideas), and it is here that knowledge is possible. The lowest level of this is the testing of hypotheses without necessarily understanding their nature. At the highest level, complete knowledge occurs when one is fully aware of the idea in one's mind and has understood its nature (Popkin et. al. 1969).

Is this how information is viewed in information science? What is the definition of information in this context? It is necessary at this point to go through a clarification of ideas and analyse just what some of the above concepts, principles and beliefs mean in information science?

As Belkin and Robertson (1976) point out, there are a variety of information concepts in use in various disciplines. A definition appropriate in the context of one discipline may not necessarily be appropriate to another (e.g. Shannon's view of information in the context of telecommunications is not necessarily appropriate to the context of information science). The view of information in other fields such as computer science, cybernetics and artificial intelligence and the connection with these fields and information science is explored in depth in (Machlup and Mansfield, 1983). Belkin and Robertson's view is that "information science should concern itself with a specific, delimitable section or portion of the information spectrum" and thus they attempt to "establish a suitable specification and delimitation" for this.

In investigating the common elements of the various uses of the term "information", Belkin and Robertson found that its relation to structure was the only common element and thus have come

up with perhaps the most general definition of information as "that which is capable of transforming structure".

They have categorised the definitions of information in various contexts in terms of an information spectrum. The spectrum ranges from the "infra-cognitive" to the "meta-cognitive", with various associated structures ranging from heredity and uncertainty to formalised knowledge. Examples include the hereditary nature of genetic information at the lower end of the spectrum, noise on a channel changing the structure of the information, and semiotic structures that help form formalised knowledge at the other end of the spectrum. Though they have defined the spectrum in this way this does not necessarily mean that information science deals with the whole spectrum.

Wersig and Neveling (1975) view the real background of information science to consist of a social responsibiliy of "transmitting knowledge to those who need it". Belkin and Robertson agree with the implication that information science is concerned with information and that it is a purpose/problem-oriented discipline. However, their definition includes its **problem** to be to

> "facilitate the communication of information between human beings"

and its **purpose** to be

> "the deliberate (purposeful) structuring of the message by the sender in order to affect the image structure of the recipient. This implies that the sender has knowledge of the recipient's structure".

Thus, they relate its purpose to encompass its structural characteristics (Belkin and Robertson, 1976).

The basic concepts of information science include definitions of *text* and *information*. A text, in information science is "a collection of signs purposefully structured by a sender with the intention of changing the image-structure of a recipient". Information is "the structure of any text which is capable of changing the image-structure of a recipient" (Belkin and Robertson, 1976). The phenomena of information science addresses the relationship between text and its associated information to the sender and recipient of it. Belkin and Robertson identify three basic phenomena of information science, although the discipline has not necessarily concerned itself equally with all three. Nevertheless, they are:

- the text and its structure (the information)
- the image-structure of the recipient and the changes in that structure
- the image-structure of the sender and the structuring of the text.

The first of these has been the major concern of information science. Given the scope and context of information science, as discussed above, we are confronted with various associated questions regarding what is meant by information systems, information services, information resources, information needs and information retrieval.

An *information system*, meaning indexing/retrieval functions, often includes aspects denoted as *resources* i.e. the quality of content and coverage. The *information service* is the user's view of the information system and resources (Bawden, 1990). Information as a resource has various qualities such as being expandable (increases with use), compressible (can be summarised),

transportable and sharable (not exchangeable as it can be given away and retained at the same time). However, this is a broader view of information as often depicted in *information resource management* where information is more like a commodity with identifiable and measurable characteristics (Bawden, 1992).

An *information retrieval system* (IRS), in its broadest sense, includes question-answering systems (or fact-retrieval systems), data retrieval systems, and passage retrieval systems (Lancaster, 1978). However, the view in this thesis is that *information retrieval* is generally taken to mean "the retrieval of references to documents in response to requests for information" and that "an *information retrieval system* is a set of rules and procedures, as operated by humans and/or machines, for doing operations such as indexing, search formulation, searching, feedback, and index language construction" (Robertson, 1981). Thus, as Salton and McGill (1983) point out, information retrieval is best understood "if one remembers that the information being processed consists of documents". In this context, therefore, information retrieval deals with the representation, storage, and access to documents (document representations) and thus refers more specifically to document retrieval systems.

2.1.2 Artificial Intelligence and Information Retrieval

The purpose in this section is to highlight the way in which the problem and task of information retrieval is viewed by those in the fields of Information Science and Artificial Intelligence (AI). This difference would not be a significant concern if the two fields did not merge at any point. However, as AI applications are being developed in IR, the difference in these two viewpoints means that, in reality, they are attempting solutions for different problems based on completely different premises - often due to misconception or misunderstanding.

Artificial Intelligence

Early definitions stated that AI was the "study of ideas that enable computers to be intelligent" (Winston, 1975) or that it was "a study of intelligent behaviour" (Genesereth, 1987). More recent definitions view AI as a branch of computer science that is concerned with the automation of intelligent behaviour, a statement which emphasises the conviction that AI is a branch of computer science (Luger and Stubblefield, 1989). Unlike Winston's statement this definition does not imply that computers themselves are or will be intelligent, but that they will simulate or imitate what we perceive to be intelligent behaviour. The common problem in all these definitions is that of defining intelligence itself (Luger and Stubblefield, 1989).

This problem is not unique to the field of AI. Many psychologists and philosophers have also questioned what "intelligent behaviour" is and how it could be recognised or tested for. It is not the purpose of this thesis to define or discuss theories of intelligence in detail and the view preferred here is one which avoids the philosophical issues and has a more practical bias.

Rich (1983), proposes that AI is "the study of how to make computers do things at which, at the moment, people are better". The definition is dependent on time and the rate of progress at getting computers to "outperform people at 'difficult' tasks". However, progress towards this aim appears to be much slower than expected. In this sense, AI offers a medium and test bed for theories of intelligence.

The majority of research in AI is based on what Newell and Simon (1976) call the *physical symbol system hypothesis*. According to their hypothesis, a physical symbol system has "the necessary and sufficient means for general intelligent action". Although there is no way of proving or disproving this hypothesis other than subjecting it to empirical evaluation, the main point is that the manipulation of symbolic structures is a sufficient process for explaining intelligence (Newell and Simon, 1976; Luger and Stubblefield, 1989). The learning approach in this work also uses conceptually-based structures such as words/terms.

Thus, another "non-philosophic" but recursive definition is one where AI is defined as the collection of problems and methodologies studied by AI researchers. The types of problems which currently fall within the scope of AI are those such as machine vision, speech recognition, robotics, expert systems, knowledge elicitation/representaion, games playing, logic, theorem proving, natural language processing, machine learning and general problem solving. Of these, knowledge representation, expert systems, natural language processing and understanding have been applied most frequently in IR (Dumais, 1988).

AI and IR

With the above definitions, it is perhaps understandable why there is such a tendency within the AI community to view the purpose of information retrieval systems to be that of answering questions (Schank et. al., 1981; Michie, 1980). Schank takes this further to say that questions should be able to be asked in natural language and their meanings extracted by the system to give appropriate answers. In discussing the use of knowledge-based systems in information retrieval, Michie also views one of the purposes of such a system to be to provide answers to problems[1].

Anwering questions is quite different to retrieving documents[2]. An additional difference in the viewpoints of the two fields is how they view information need and the purpose of an IRS in this context. Here, the difference is that the aim of an IRS is to answer the information need itself or that it is "to find information items relevant to an information need" (Bartschi, 1985).

According to Sparck Jones, the AI approach is fundamentally misconceived precisely because it is based on the wrong general model of information retrieval as question answering. Her argument is that there is an underlying presumption that there is "an amount of definiteness in the perception and characterisation of user needs". She also points to a more extreme end of the AI claim that the text base should be replaced by a knowledge base. In this claim, the assumption is that there could be a representation of the text (other than the text language itself) which captures all its meanings and expressive properties in an unambiguous and explicit manner (Sparck Jones, 1990).

Thus, if an IRS is to contain meanings and properties explicitly then a good IRS must have the ability to change its categorisation system as necessary. According to Schank (Schank et. al., 1981), for a database that is organised by indices that do not express meanings, this may seem

[1]The implication being that the problem can be translated into an explicit question.

[2]Documents themselves may or may not be able to provide answers to some questions.

like a goal that is beyond hope. And perhaps it is for such systems. However, his suggestion is that when the basic data are meaning, and the memory organisation used is dependent on a program that attempts to generalise meaning units in an attempt to form new ones (i.e. inference), then such a system is within the bounds of plausibility.

On the other hand, if we view information retrieval as the retrieval of documents associated with various requests/needs for information then the technology of information retrieval (document retrieval) is based on the Aristotelian principle, mentioned earlier, that the human mind works by association (Durham, 1989). To say that the documents can be represented by facts alone is ignoring the importance of the recipient and their perception or perspective. If we accept that the human mind works by association then documents are predominantly useful within this association. Therefore, the content of a document can only be meaningful within this *associative framework* or *context.*

On the question of information comprising of 'meaning units' there is also the issue of how it can be manipulated. Belkin and Robertson (1976) point out that fact-retrieval systems view information as an atomic phenomenon whose basic units can be singled out, taken out of context and combined in different ways. An implication of this is that these 'discrete' units can in some way form a 'universal' knowledge structure. The importance of context in information retrieval is one of the main motivations of the research in this thesis, as discussed later in section 2.1.4 and 5.

Mooers (1960)[3] also had made some predictions regarding the role of inference in an IRS. Mooers' belief was that given solved indexing or translation samples, an inferencing mechanism could derive rules which might have been used to perform the corresponding indexing or translation tasks. Salton (1987) clearly points out the immense complexity and difficulty of this task, which still applies.[4]

So, what is the argument for AI in developing IRSs? The argument according to Sparck Jones (1990), is not the question answering in a single domain but that it is required to support the integrated information management system of the future. Thus, the system should have something like an "intelligent catalogue" with enough internal knowledge that will enable it to relate to its resources and reduce the load on the user thinking about which resources might be help him (Sparck Jones, 1990). An additional argument, put forward in this thesis, is that AI offers a wide range of tools of which some have the potential of being useful. It is the work of this thesis to see if such an approach is not only possible but also useful (to the user or recipient). In investigating this, rather than concentrating on 'intelligent cataloguing or classification' the work focuses on 'intelligent ordering of documents'.

---

[3]Mooers made, in the early sixties, predictions concerning text processing and IR for the next two decades. Most of his forecasts were "unusually" accurate as Salton and McGill (1983) state - except perhaps the one mentioned here).

[4]In this context, Salton states that "In the normal document collection environment one must necessarily deal with different types of information and different contexts. In these circumstances, the inferencing step needed to produce a valid system of rules for a given task is most hazardous".

2.1.3 Models of/in Information Retrieval

Having discussed the various views on information, information need and retrieval we can now view some information retrieval paradigms (bearing in mind the overall goal of research in IR as being to understand and facilitate communication of desired information between a human generator and user - Belkin).

One way of viewing research in information retrieval is in the following three categories: the system-driven approach, the user modelling view and the cognitive paradigm. The first of these deals with the effectiveness and quality of retrieval techniques, indexing, information system design and analysis. The second is concerned with the idea of desired information, information in human, cognitive communication systems, and the relationship between information and user. The cognitive approach combines the two previous approaches mainly studying knowledge-based, interactive information retrieval processes and systems (Ingwersen, 1987). The role of physical and cognitive paradigms in IR research is discussed further by Ellis (1992) who believes neither provides a basis for a powerful paradigm directed science.

In the absence of a generally accepted theory of information retrieval, Robertson (1977) highlights that "any theory that appears to deal with or relate to any part of the storage-and-retrieval process is potentially part of a theory". He refers to classification theories, linguistic theories, psychological approaches and mathematical theories as examples. However, later in his discussion about the future of IR, Robertson (1990) mentions five models which are to be viewed as complementary to each other. Namely, technical models, conceptual models, interactional models, cognitive/behavioural models and social/organisational models.[5]
Fields such as machine learning can be more applicable in the technical and conceptual models whlist human-computer interaction (HCI) has more to offer the interactional models. The more interesting aspect Robertson's models is that he identifies their links. Following is a list of some possible links along with their descriptions:

- Technical/Interactional: This include models developed for the technical area also having an impact on the ways in which they are interacted. For example, in using the Boolean approach the searcher's major task is to identify a suitable size set. Other links are those encompassing relevance feedback where searcher is required to distinguish between relevant and non-relevant items.

- Conceptual/Interactional: Encompasses systems based on topic menus in which the approach is to start from a model of knowledge as indicated by a formal classification scheme and then produce a model of interaction. E.g. Cansearch[6] - this is how it is therefore this is how the user should see it.

---

[5]The models concerned represent the following:

- Technical models: Those dealing with the technical aspects of IR, such as Boolean logic, clustering, probabilistic models and vector space models.
- Conceptual models: Models for structuring knowledge (hierarchies, facets etc.).
- Interactional models: Models which concentrate on the importance of searcher-system interaction.
- Cognitive/behavioural models: Models relating to user information-seeking behaviour such as ASK (Belkin). The particular emphasis is on user behaviour in relation to the perceived problem.
- Social/organisational models: Those models concerned with the social patterns of information transfer (such as citation studies) and those concentrating on the social aspects of information transfer.

[6]This is also referred to later in this chapter.

- Technical/Conceptual: Traditional links especially concerning some aspects of clustering and statistical indexing.

- Interactional/Cognitive: Systems which have a strong cognitive base and build the system-user interaction on this. E.g. Distributed Expert System[7] in which the system is able to build internal models of the user, domain, search topic etc. because it takes on cognitive activity normally done by human intermediaries.

- Cognitive/Social: Emphasis on placing models of information-seeking behaviour in a social context.

- Cognitive/Conceptual: Possibilities of work on users' conceptual structures for/during information seeking and how specific they may be to a particular user's knowledge structures or how they may be generalised to encompass other users.

The work for this thesis consists of applying a *learner* to help in information retrieval. The learner is based on a model of the IR situation which includes the idea of *context*. For the moment, we will define *context* to be "the subject area in which the query of a particular user is concerned with". This is addressed in greater detail in the next section.

Therefore, regarding the above categories, this work fits more into the first and last of the above categories. An existing technical model is enhanced to help improve the order in which users see document references. The second (cognitive/conceptual link) in this work is slightly weaker. It refers to the argument that each user does have a context for his/her problem and that this affects their queries. In addition, groups of users (perhaps working on the same project) may have a higher degree of similarity in their contexts.

The purpose, of this section, is to provide the framework for the argument of *learning using context knowledge* in this thesis. The framework referred to here is Belkin's model of the IR situation (Belkin, 1990).

According to this model, there are the following three stages in an IR situation:
1) A person with goals, intentions or a problem to resolve, finds him/herself in a problematic situation. This occurs when a person recognises that his/her internal resources are in some way inadequate for; the resolution or management of some problem; or the achievement of some goal; or the realising of some intent.
2) A characteristic of the problematic situation is the person's Anamalous State of Knowledge (ASK). In order to manage the problem (or achieve the goal or realise the intent) this person attempts to resolve the ASK.
3) In order to resolve the ASK, the person has recourse to some knowledge resource external to him/herself. This being a collection of texts, organised and represented in some way.

A possible response to an ASK is an information need and this in turn can initiate a query. Generally, Users' needs and demands from information services (more specifically an IRS) fall into two categories[8] (Lancaster, 1978). The first is one in which the user wants to obtain a

[7]The work of Belkin, Daniels and Brooks -ref?

[8]These two categories do not completely cover information needs which may consist of browsing, for example.

particular copy of a particular document. In the second, there is the need to locate documents dealing with a particular subject (current awareness where the information may initiate action) or those that will help answer/solve a particular question or problem. Problem-solving information includes the need for a single item of factual data, the need to have documents discussing a particular subject (rather than complete current awareness) and the need for a comprehensive search.

In an IRS, the query which results from an information need is expressed in a search statement. The process beginning from the problem situation to the eventual search statement can be seen in Fig.2.1.

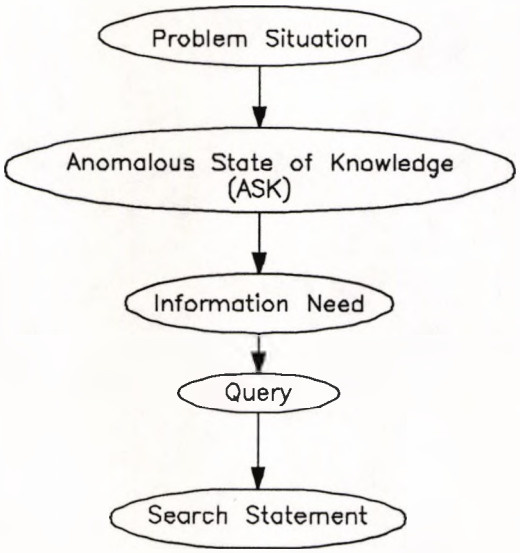


Fig.2.1: The entities in an Information Retrieval situation. Diagram partly based on that in Robertson (1979).

The application of machine learning to IR, in this work, is performed through what is termed as a *context learner*. The *problem situation* can be said to represent the *context* (in which the query occurs) and the *context learner*'s aim is to help resolve the ASK. The scope and meaning of context is discussed more fully in the next section.

2.1.4 The Scope and Meaning of Context

Standard Information Retrieval Systems (IRS) can be used to retrieve information in response to specific requests, but they have no powers of adaption to particular users over repeated sessions. Here, a learning system is described which uses relevance feedback from a probabilistic IRS to incrementally evolve a *context* for a user, over a number of online sessions. The learning implementation is demonstrated with examples (chapter 5), and it is argued that this can help an IRS adapt to a user's specific needs by using this *context* to influence document display and selection.

The scope of Context

Adaptive techniques are applicable to those computer systems that are subject to repeated use

by particular users. The user's needs will be quite specific, whereas the system will be generally applicable. Such is the case in an on-line IRS; a user can retrieve documents in any subject area through interactive sessions, and may use the system several times to satisfy a specific information need. The standard IRS does not, however, adapt to a user's need over repeated sessions: each time the system is used, the IRS starts from its general stance. The work here is aimed at overcoming this failing.

IRSs vary in their types of input and output: a common but somewhat rigid type of input is for users to represent their information need in a binary form through boolean expressions. The output from these systems tends to be an unordered list of documents and the query formulation requires knowledge of boolean logic.

Systems accepting natural language input are more flexible when accepting user queries. However, although the mapping of the information need to the query may be easier for the user, interpreting and processing queries in this form is not as straight forward. Unlike boolean queries, the relationship between terms in the query is not explicit. A *probabilistic* IRS which accepts natural language input can rank the output documents by attaching scores (numerical measures) to the documents in the collection, based on the current query. The order of the references output reflects their probability of relevance, where relevance can be seen as a relationship between the document and the need. To improve the performance of a Probabilistic IRS further, *relevance feedback* is sought from a user. This permits users to choose documents which are relevant to their need, and allows the system to expand the original query (query expansion) using additional terms from the relevant documents.

A probabilistic IRS is taken one step further to introduce a learning component which adapts to a user over a period of online sessions. Relevance feedback from these sessions can be input to an incremental context learner. The meaning and interpretation of *context* as referred to in this thesis is what is addressed in this section and below is a diagram which places this adaptive technique into an IR system with relevance feedback. Later (section 5.2), it will be described how the learner can influence the ordering or inclusion of documents output in later sessions. The control and data flow of the probabilistic IRS incorporating the context learner is shown in Fig.2.2.

Users will tend to repeat searches or conduct a series of closely related searches over a period (Walker and Hancock-Beaulieu, 1991). Although *each search must be regarded as representing a different information need, they can all be assumed to have a common* ***context****.* A document that is judged relevant to the need which prompted one search will not necessarily be relevant to the next need, but is relevant to the context in which the need falls.

Additionally, although two users may retrieve the same documents, this does not mean that they have the same *context*. Furnas et. al. (1987) highlight how context dependent vocabulary usage is and how it may differ between users. They point out that the same objects can be refered to differently and different words can mean the same thing.

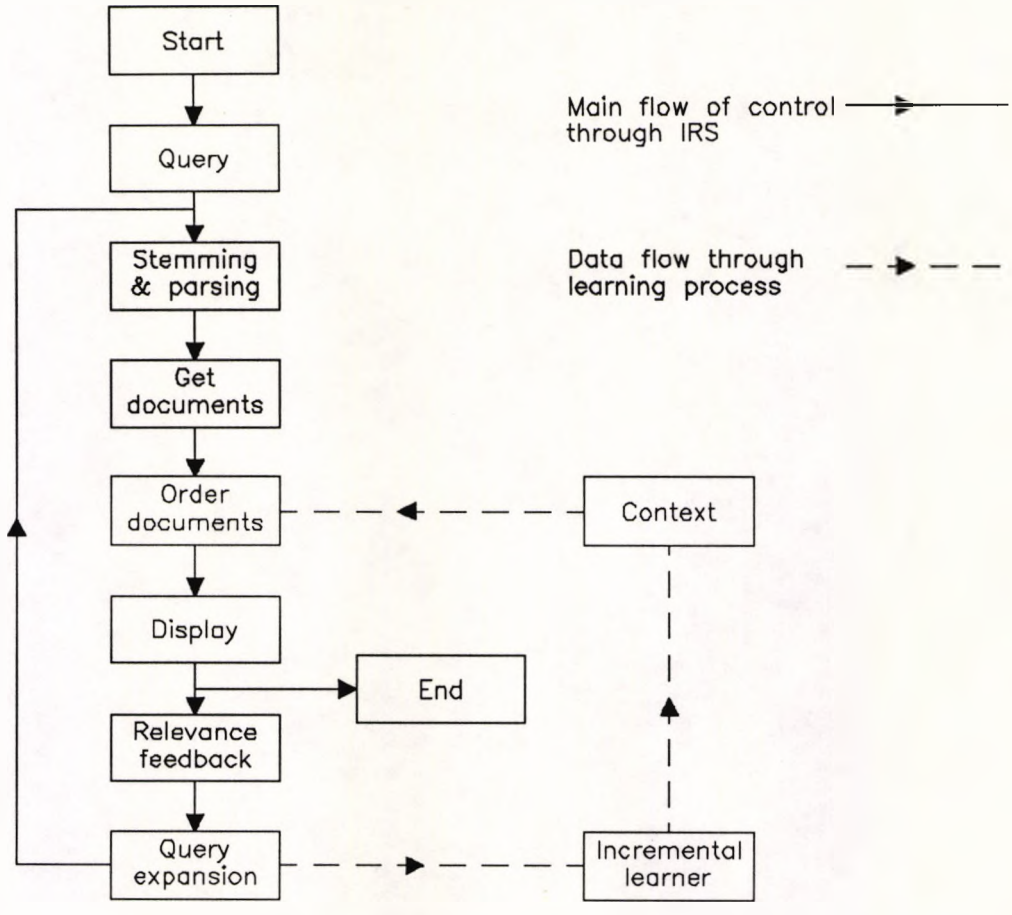


Fig.2.2: Control and Data Flow in the IRS

The meaning of Context

In the most general sense we can take *context* to mean the subject area or sub-area that the user has in mind when forming a query. However, the boundaries between these "areas" may not always be so clear. Defining the boundaries of sub-areas even if we were to assume they did not merge, is a complex task. The point here is that some core clusters of terms (identifiable as 'sub-areas') exist (as presented in chapter 6) and that it is reasonable to investigate if they are usefully linked.

Users (in such IR environments[9], whether they are aware of it or not) tend to have two or at most three subject areas in mind (sections 6.4, 6.5). Their queries are not independent of each other. They appear, on the whole, to be related.

The notion of a 'subject' or 'subject matter' is discussed in detail by Hjorland (1992). He points out that the conceptions of 'subject' in the literature is generally not explicit but implicit. There is a difference between regarding 'subject' as a fixed property of documents and viewing it from the point of view of those working in the field or the users.[10] In his analysis of the theory of

[9]The IRS was in an academic institution.

[10]Viewing it as a fixed property would be somewhat analogous of Schank's view of information described in the previous section.

'subject', Hjorland includes the points of view of subjective idealism, objective idealism, pragmatism and realism/materialism. These are explained in considerable detail as the extensive study helps clarify differing viewpoints regarding *subject* and *context*.

Subjective Idealism

In relation to the subjective-idealistic theory of 'subject matter' it seems there is no certain or objective knowledge about the subject of a document nor can the concept of a *subject* be defined. In this theory the emphasis is on structuring and ordering the contents of documents according to users' subject definitions.

Those who argue that subjects should be ordered according to each user's subjective reading are inclined to build on psychological investigations of the users' perceptions of the subject and their 'knowledge structures'. Tailoring the description of the subjects to the user's subjective perceptions can be useful in some cases such as in childrens' libraries. To a certain extent these are 'paternalistic' viewpoints because someone or the system assumes responsibility for the direction of users' searches.

Objective Idealism

At the other end of the spectrum, objective idealism emphasises an absoluteness in identifying and arriving at a subject. According to this theory, if two separate persons perform the correct analysis they should arrive at the same subject for a given document. In the extreme sense, objective idealism implies that document classification can be done independently of the context in which the classification is being used.

Pragmatism

Looking at the idea of *subject* from a pragmatic point of view, a user has a particular need for information, a problem to be solved for which information is required. The documents, which contain this information, are registered by subjects and indexed so that they can be retrieved.

According to this view, two types of indexing exist. These are *content oriented indexing* and *request oriented indexing* (or *user-oriented indexing*). In the first, documents are described by refering to their contents (E.g.: this document mentions... therefore it is categorised under...). In the second, however, the connection between the properties of a document and a user is emphasised. Classification is done according to the need of a particular user or target group.

Realism / Materialism

In this context, Hjorland makes no distinction between 'realism' and 'materialism'. Here, things exist objectively and must be represented as such. Therefore, although a document contains the (subjective) viewpoint of the author it does also have some objective properties. This is a similar idea to that embodied in *belief systems* in artificial intelligence. Belief systems attempt to overcome the problem of uncertainty in 'facts' or in knowledge-base systems typically. For example, if there was an uncertainty associated with the assertion "all students are poor" then this might be overcome by asserting that "Mary Jo Poor believes that all students are poor" or

that "some people believe that all students are poor".[11]

What is meant by the properties of a document and which of these are relevant or useful for a 'subject' description is an essential question in this view of 'subject' theory. Indeed are the properties of a document also an attribute or function of the document? Properties of documents emerge when they are actually used.

This begs questions as to what the objective criteria for the subject of a document are? Are subjects perceptions or 'ideas' in peoples minds or something else? And what is meant by the statement 'document A belongs to subject category X'? Curiously enough, objectivity may not always be in agreement with reality e.g. if the majority of people really believed that all students were poor then does this make it real? Or just because the majority of users may choose to classify a document in a certain way is that necessarily the most accurate way - as might be done by an information retrieval specialist. Does this make the user the specialist in his problem? A solution is to put the classifications and documents to the test, thus, history in the end, enables us to gain some objectivity. In this sense, the work described in this thesis also depends on the historical information gained from users' queries in the system.

Regarding the above categories, this work on *context* starts off from a *pragmatic* point but inorder to retain some objectivity a *realist/materialist* approach is added. The idea of *context*, though not necessarily referred to as such, can also be found in work on classification, clustering, SDI (Selective Dissemination of Information) and cognitive modelling of complex (control) tasks. A more detailed explanation with particular reference to these four categories follows.

According to *subjective idealism* the structure/order of contents of documents is defined by users' subject definitions. In the context of this work, 'if' or 'where' the user acquires categories from is irrelevant. The point is that there appears to be a common theme across a user's queries.

This work is based on 'observing' what users actually do and although this is not a completely contrary idea to subjective idealism there is an emphasis on structuring/ordering the contents according to users' subject definitions i.e. no statement is made about their content. The work here does not try to define the contents of a document but having identified the documents, extractions are made.

Paternalism as can be seen in *subjective idealism* does exist to a certain degree. This mainly occurs in the decision-making of reordering documents presented to a user. Section 5.2 describes the various ways in which the Context Learner can be put to use with varying amounts of 'interference' with the user's original queries, though the preference is to keep this to a minimum.

---

[11]Subjects could also be objectively stated as potentials of documents e.g.: 1) Penicilin was always an antibiotic even before this was discovered to be so. 2) Artificial intelligence was of potential use to IR even before people started to address/refer to the possible connection - this is different to saying it is a useful connection.

Objective idealism is not appropriate in this thesis. To refer to Hjorland: "subject descriptions based on objective idealism have an abstract relationship to the needs for subject description and the *contexts* in which they are used". i.e. there appears to be no connection between the general and the specific, the general exists outside and independent of the particular. Thus, there is not much scope for applying whatever characteristics of *context* may be learnt at a general level to specific users.

The idea of *context* developed here is in between the two forms of idealism described so far. Being idealistic theories, both subjective idealism and objective idealism assume that the true subjects in documents can be identified (through some form of abstract analysis or fixed procedure). However, neither the documents' potential use nor actual usefulness is addressed.

The *pragmatic* view of *subject* is the closest to the one of *context* as described in this thesis. In relation to *content-oriented indexing* and *user-oriented indexing*, the work here could probably be more accurately described as *user/need-oriented ranking*. Classifications do not remain static and thus have a temporary nature. The idea of (subject) classification representing a relation, as it may exist at a certain point in time, is parallel to the idea of a user *context* changing with increased usage of the system. There is a delicate balance between encompassing user-needs to make a system adaptable and a possible over emphasis on individual user variations (to the extent where this may degrade overall system performance).

Like the final theory of *realism/materialism*, the purpose here is not to fully define the properties of documents retrieved. The aim is to identify some characteristics which might make them more 'useful' to a particular user/need and this precisely depends on the *context*.

A slightly different perspective of *context* is discussed by Grant (1990) where he applies this to cognitive task analysis. His views are also based on the notion that human data processes depend greatly on the situation and individual (Rasmussen, 1980) which may affect information requirements and priorities.

A context, according to Grant, is represented by a 'package' of rules[12] and information requirements - a view derived from cognitive modelling of complex control tasks. *Context* is a particular *stage* of a task, along with the rules and the information that are being used during this stage. Thus, it can be likened to a frame. Higher-level rules for switching from one context to another could be seen either as a property of individual contexts, or as a property of the representation as a whole. When there is a switch from one context to another there can also be a 'reorientation' context during the transition. In this view, context is an entity/ or a point of reference. The emergence of such contexts where tasks and system performance are situation and person dependent are also seen in IR.

Obtaining the *context* structures can be done in the following ways:

- ask subjects what they perceive it to be (subjective idealism)
- examine information they use and look for patterns

[12]In Grant's context, rules are induced based on a few examples (attributes).

- derive it from the application of rules

The second of these is more closely related to the approach taken for the work in this thesis.

The Belkin (1990) model of an IR situation described (Section 2.1.3) a general sequence of events in which a problem situation results in an information need which is translated into a query. In relation to *context*, we can say that a problem occurs in a particular *context* and the query results from that problem. It may well be that context relates to a solution, if it exists. The notion of context can be discussed in terms of such models and theories. However, for the purposes of retrieval it is important to make some interpretation of context in relation to queries and documents, as opposed to problems and solutions. There is a many-many relation between queries and documents chosen relevant (Figure 2.3). One query can result in the retrieval of one or more relevant documents and a particular document chosen relevant for one query can also be relevant to others. Thus, it is reasonable to extract terms from relevant documents and apply them to other documents when trying to project the context. Context information derived in this way can be used to improve retrieval as also discussed later in Sections 5.1, 5.2.

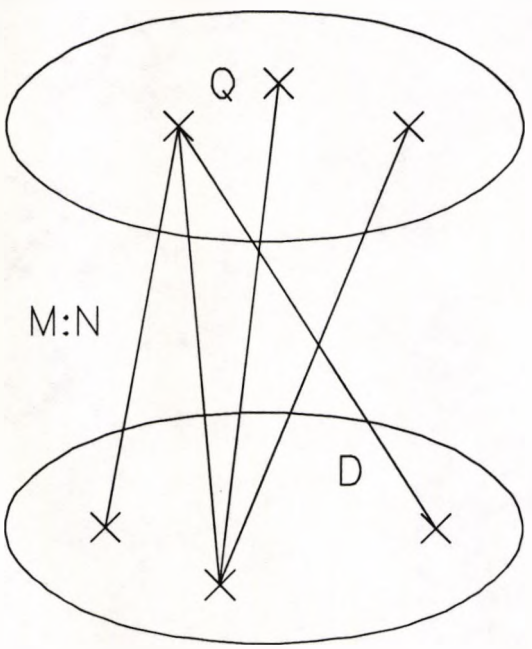


Fig.2.3: The mapping between queries and documents chosen relevant.

The Application of Context

An emphasis of this work is the functionality of having a context structure. The argument is that context C1 derived from query Q1 will be useful for Q2. This is based on Q2 being related to Q1, for the same user. The aim was to see and if possible show that Q2 is indeed related to Q1 and that C1 is somehow useful for C2. Ideally, the resulting algorithm would help if there is a connection but not damage if not. Figure 2.4 shows the order of the queries and their associated contexts. Chapter 5 discusses the approximations to context in more detail.

Later (in Chapter 6) there is a more detailed description and discussion of a particular problem in an IRS, called the Weight-Block problem. This occurs when a document ranking IRS has too

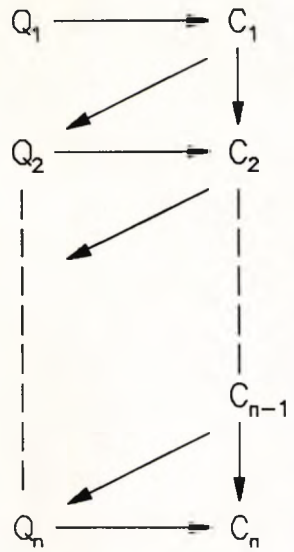


Fig.2.4: Order of queries and 'contexts' generated

many documents which have the same document weight/score. In such a situation, it appears as though there is no document ranking scheme. Hence, the necessity to somehow break up these document weights in the Weight-Block. A *Context Learner*, based on the notion of *context* described, could help overcome this type of problem. It should give a certain "perspective" on the user queries and thus any (relevant) contextual information can be used to help order/rank the document references presented to the user. This perspective is gained from a history of queries and is an insight into the user's queries through relevant documents chosen in previous sessions. This is a somewhat parallel problem to that identified by Hjorland (1992) who says that the larger the volumes of data, the more necessary it becomes to describe the subjects (based on user needs and not properties of documents) in them as accurately as possible (rather than prioritising the documents).

The learning system, here, is not used to modify the order of items in the whole retrieved set from the current query, but to break ties i.e. to rank items within Weight-Blocks. Reordering the whole retrieved set is a greater interference with the current ranking system. Another way of reordering the documents would be to, unbeknown to the user, modify the query using *context* data and retrieve a different set of documents. This is an even higher level of interference with the user's query. Neither of these two methods is seen as particularly desirable nor is it necessary as an initial step to improving the current ranking system. Thus, the context is used only to affect the order of documents which have the same scores.

2.1.5 Intelligent Information Retrieval

Work on intelligent information retrieval has been done in various areas such as user modelling, improving the user interface, document and term classificiation, thesaurus enhancing, using relevance feedback and mathematical techniques. The mathematical approaches are diverse but they generally focus on associations between word frequencies, document use and meaning.

Of the techniques used for intelligent IR, some constitute part of the retrieval system while others have been built in to intelligent front-ends. However, the work has tended to concentrate on storing and retrieving text, based on its lexical content rather than meaning. The following sections exemplify some of these techniques. The boundaries between the categories are not always definite and hybrid systems encompassing a variety of approaches have been developed.

2.2 Modelling the user and system

This involves having user and system models to determine the state of users' knowledge and their requirements. The subject of cognitive modelling and IR has been reviewed more extensively by others (eg: Daniels, 1986; Kobsa and Wahlster, 1989; Ingwersen, 1982). Here, the purpose is to illustrate how it can be used for intelligent IR.

In developing cognitive models for IR systems, we need to be clear about what we are modelling and for what purpose. For example, if we are aiming to enhance query formulation should we model the intermediary or the user? Additionally, certain aspects of users can influence modelling such as the roles they may have. Sparck Jones (1989) addresses these issues and that concerning what modelling information can be obtained. She also refers to a widespread assumption that the more user model the better. Sometimes having a simple but 'accurate' model may be better than a complex but 'fuzzy' one.

The classification of modelling used here is based on a distinction between *dynamic* and *static* models (Sparck Jones, 1984). Dynamic models represent changes of user state which depend on interaction with the system. Static models deal with the consistent user features which are independent of the system.

2.2.1 Dynamic User Models

THOMAS (Oddy, 1977), for example, was one of the earlier systems developed using user models. This prototype system was designed on the basis that users might learn more about the information that would be useful to them if they knew what was available. Particularly when requests may not necessarily be well-defined, browsing (manual/computer) through the available material can be important towards retrieval.

The underlying assumption of this dynamic model is that both the user and the intermediary have their own models of the information world and that each must construct a model of the other's view, as they understand it. The system, therefore, does not assist users with query formulation in the usual way but instead uses users' reactions to the shown references and document descriptions to create a model. It starts off with a model of the user's interest area based on the literature in the database and then modifies this continuously in relation to the

users' reactions to what is presented. The user did not deal directly with the information structure of the database, but interacted with it through a model of his perceptions or requirements.

The incremental process of adjusting the model of the user is akin to the model of *context* in this work. The approximation of the *user context* is based on *incremental learning*.

2.2.2 Static User Models

These models tend to build structures/frames to distinguish user requirements. They do not adapt in the same way to changing user requirements, in an on-line session.

GRUNDY (Rich, 1979; 1989) is such an interactive system which uses a user model in understanding the user's requests. It assists library users in choosing novels. Here, the user's goal is defined in the sense that the aim is to find a novel. The system takes a few descriptions given by the user, of him/herself, and uses them in comparing with the stereotypes encoded. The differences between these stereotypes and the user are resolved in order to recommend a novel. The system also provides a method for selecting which features of the recommended books would be of interest to the user.

I^3R (Croft and Thomson, 1985) is an Expert Assistant which is designed to provide the functions of the intermediary who may not necessarily be knowledgeable about the subject area but be well-informed about the system's capabilities. Hence, it contains knowledge about different methods of formulating queries, retrieval strategies and types of users. The system has three components. Namely, an interface manager, system experts (a browsing expert, an explainer, a search controller, a thesaurus expert, etc.) and the knowledgebase.

The user models in this system, differ from others in that instead of representing particular queries they represent certain characteristics of the user. The categorisations of the different user types are basic and range from novice to experts.

A similar approach of recognising characteristics, used in I^3R can also be found in WIZARD (Finn, 1983). This system was designed to recognise user "plans" in order to provide on-line help. It detects that a certain sequence of commands input by the user constitute a certain type of plan. Information is kept on which commands the user has been using and what advice has been given.

Belkin, Daniels, Brooks (Brooks et al., 1985; Daniels, 1986) worked on developing a distributed expert system for an automated intelligent system interface. The aim was to simulate the on-line retrieval done by good intermediaries. These individual parts (User Model, Problem Description, Retrieval Strategy) were designed to interact with one other so that information learned by one module could be the input for another. Some of these parts aim to form a user model through determining the user's status, goals, state of knowledge about the subject area, familiarity with information retrieval systems and their background. Others are aimed at identifying and specifying the problem. Many interactions between users and intermediaries were analysed in

order to form as realistic a model as possible. However, this study did not proceed beyond the design stage.

CODER (COmposite Document Expert/extended/effective Retrieval System) (Fox, 1989) was designed to acquire knowledge about users who may be of diverse groups. Both this system and I^3R were influenced by the distributed expert system architecture in the HEARSAY II (Erman et al., 1980) speech recognition system. HEARSAY contains several 'experts' who cooperate to interpret a particular stream of speech. The system consists of an analysis subsystem and a retrieval subsystem. The first deals with the entering, processing and representation of new documents, the second deals with the users' retrieval of part or whole documents. The two subsystems share some resources ie. the document database.

The following example is a summary of possible user menu choices during a session.

Eg:

1. ASSISTANCE: Would you like
   - a- An explanation of the current situation
   - b- Help regarding what you might do next
   - c- A tutorial about some phase of the system's operations
2. COLLECTING: Can you provide more information about
   - a- Your background
   - b- The context or problem that prompted you to begin this session
   - c- Your evaluation of the system's performance
3. QUERY: Can you
   - a- Enter a description of your information need
   - b- Revise the already existing description
4. BROWSING: Would you like to examine
   - a- Facts from the "Handbook of Artificial Intelligence"
   - b- Entries in the "Collins Dictionary of the English Language"
   - c- Retrieved or other documents
   - d- Information recorded about you in the User Model database
5. RESULTS: Do you need to
   - a- Print some items
   - b- Save some items in a file
6. EXIT

Below is an illustration of the type of information held in the system about the users:

```
USER BACKGROUND
    Userid is foxe
    User Identification: foxe
    Slot *info* is a frame:
    Education: doctoral
    Field of education: cs
    English as native language: y
    Gender (m=male, f=female): m
    Slot *knowledge* is a frame:
    Ever used a computer: y
    Used other IS&R systems: y
    Number of times used other IS&R systems: 30
    Taken Information Retrieval courses: y
```

Know Boolean logic: y
User classification: average
Frequency of system use: 2

2.3 Classification, Categorisation and Clustering

Classification, categorisation and clustering can all be performed on documents, terms and users[13]. "Classification", however, as Lewis (1992) points out, is an ambiguous term in IR. Thus, to help clarify this, Lewis's structured definitions relating to the above headed three terms are used.

Text classification involves the assigning of documents or parts of documents to one or more of a number of groups. *Text categorisation* is the "classification of documents with respect to a set of one or more pre-existing categories" (e.g. indexing documents for text retrieval). In *term categorisation*, like text categorisation, pieces of text are assigned to predefined categories (e.g. thesaurus index terms). However, the size of the pieces of text are different (words or small fragments of text as opposed to whole documents or document representations).

Rich (1989) classifies users according to stereotypes (naive, expert etc.) - also called *user profiles*. Salton and McGill (1983), on the other hand, address document and term classification. According to them, classification is used for two main purposes, in IR: to classify the set of index terms (key words) and to classify the documents into subject classes.

Examples of text categorisation systems include a rule-based system, CONSTRUE (Hayes et. al., 1988), and the work of Bhandarkar et. al. (1989) based on a statistical model. Both use news stories for the categorisation task. A more detailed analysis and description of text categorisation systems can be found in Lewis (1992). Some classification techniques such as decision trees, also used in the field of AI, apply to text categorisation as well. Bhatia and Deogun (1991) uses the ID3 algorithm[14], however, to determine user profiles to help cope with different users' viewpoints of terms. It is used to reformulate the user-specified keywords into a well-defined query that performs retrieval as per user perception.

Document and *term clustering*, on the other hand, "involve not only the assignment of portions of text to categories, but the creation of those categories from a corpus of text". In *document clustering*, categories of documents are automatically generated, usually based on some similarity measure between documents and a definition of what characteristics groups of documents should have.[15] *Term clustering* is similar to document clustering, "except that individual words or small fragments consisting of closely connected words are formed into groups."

---

[13]It is not usual, to have clustering algorithms that operate on users.

[14]The ID3 algorithm, advanced by Quinlan (1983), identifies most relevant properties in a large amount of data that determine the classification of an object (within the training set). See also Section 3.1.3.

[15]Document clustering can be used to improve the effectivenes of text retrieval or speed up the physical access to stored documents.

According to the *clustering hypothesis* (Salton and McGill, 1983), closely associated documents tend to be relevant to the same queries. Thus, it is hoped that using clustered document files may lead to high precision and recall values. The process consists of taking an item (document/query) and comparing it with an existing cluster to identify the degree to which that item may belong to that cluster.

Salton and McGill view clustering methods to fall into two categories, depending on whether an initial set of classes already exist or all new class must be constructed from scratch. Generally, a prior classification is not available therefore it is necessary to construct new classifications for given set of items. Salton refers to another criterion here which distinguishes between hierarchical grouping methods and iterative methods where an initial rough classification is improved upon iteratively.[16] The earlier work of Gotlieb and Kumar (1968) involved defining an association measure between index terms. They found association measure to be heavily dependent on the document collection. Lesk (1969) found that, in small collections, associations are not useful for determining word meanings or relations, as they tend to depend purely on local meanings of words. It would appear that (Sparck Jones, 1991) associative information itself is most likely to have some utility when refined using relevance facts.

Whatever the object (terms/documents) being classified, the classification should be stable so that any additions, deletions or changes do not disrupt or significantly change the structure formed initially. Classifications should also be well defined. From a theoretical viewpoint, as this seems to imply that classification could be done objectively and independently of the context in which it is to be used, it would somewhat tend to objective idealism, described earlier in section 2.1.

Context and Term Clustering

Term clustering has been used to provide alternative text representations to help improve and support text retrieval (Lewis, 1992). It can also be used to investigate word usage in the context of NLP.

In a sense, the work here is not just *term clustering* but *query clustering* (related to individual users) as well - at the abstract level. Query clustering does not mean that all terms in a query automatically group to form a part or whole of a cluster but that they are used to derive further approximations to *context*. It could be argued that trying to find all the terms to completely represent the *context* (defined earlier in Section 2.1) might reduce performance. It is the clustering of queries at the conceptual level that is important and the way in which they may be incorporated is a matter of choice of algorithm.

The quality of term sets (as in term clustering) has been looked into by Sparck Jones in order to understand why term clustering led to improved effectiveness in some collections and not in others. Some sets are better at separating relevant documents from non-relevant documents. If the terms in the set tend to be in both the relevant and non-relevant documents then they are not as effective. Van Rijsbergen and Sparck Jones, thus, developed a Clustering Hypothesis Test to determine whether an initial set of terms can be improved by term clustering. Precisely which

[16] In principle, hiearchical groupings could also be improved iteratively.

form of 'clustering' might be more appropriate for users with queries which tend to fall into a few *contexts* is addressed later in Chapter 5 and their evaluation can be found in Chapter 7.

Text Routing

Also known as *selective dissemination of information (SDI)*, *text routing* combines aspects of text categorisation and retrieval. It is an information alerting method designed to keep individuals (typically researchers in a particular field) informed of new developments in their particular fields of interest (Mondschein, 1990; Barker et. al., 1972). It is a personalised current awareness service directed at the individual user or a group of users. The categorisation task involves assigning documents to zero or more of a set of classes. In the retrieval task, each class is typically associated with the information needs of one or a small group of users.

In this approach, the aim is to develop an *SDI/search profile* to represent users' information needs into a set of terms, linked in a logical way, which can then be matched to the text/database. The profile can be modified as the user's (or user groups') needs change.

It is hoped that the use of an SDI would relate to productivity (i.e. research productivity). Relevance feedback can also be used in text routing and "has the potential for being more effective than in text retrieval, since the information need persists over a longer period of time". This is precisely the argument for the *context learner*.

In reference to SDI profiles, Robertson and Sparck Jones (1976) point out that relevance weights give better performance than simple term matching and in some situations performance may be expected to improve with cumulative information. They also indicate that the best use of "all available information" should be made (i.e. relevance feedback data, user's prior expectaions to using the system and frequencies of terms the user wants).

2.4 Expanding the Role of the Thesaurus

Shoval's Knowledge Base Thesaurus (Shoval, 1985) was a significant contribution to expanding the functions and contents of the thesaurus. The system consisted of a thesaurus which included information about the terms in it and their associations. It then searched other terms of potential relevance from this enhanced thesaurus. The nature of the relation between terms in thesauri, the assignment of thesaurus terms and their frequency of usage have been studied by others (Willetts, 1975; Henry and Diodato, 1991), although not incorporated into a knowledge base.

TOME SEARCHER (TOME, 1988) can be used as a front-end to available on-line services or it can be installed by a database host as an additional help to users. The software assists users in their queries before going on-line, thereby, reducing connection time and costs.

The system can consist, depending on the installation's requirements, of a base classical thesaurus (IEEE terms etc.), a specialist classical thesaurus to enrich its domain-specific knowledge (computer/information technology terms), client terminology (related companies, product names, project names etc.) and an individual user dictionary (unrecognised terms used

by users). It was considered that a thesaurus was also potential raw material for the knowledge-base of an expert system. Hence, the information is stored as a semantic network.

Following is an example of a query and how it is enhanced by TOME SEARCHER. There are two processes involved: the expansion of the query via a thesaurus and its interpretation in Boolean form.

Eg:

Query: I would like to know about optical discs used with PCs excluding Apples.

This is then converted into its equivalent Boolean statement.

Initial Boolean strategy:

(OPTICAL DISC$ or OPTICAL DISK$ or OPTICAL STORAGE) and
(PC$ or PERSONAL COMPUTER$) and not APPLE$
where "$" signifies truncation

The database or its model is consulted to determine whether the strategy needs broadening or refining, depending on the number of records. Should the above strategy need broadening, it could be converted to the following Boolean strategy:

Broadened Boolean strategy:

(OPTICAL DISC$ or OPTICAL DISK$ or OPTICAL STORAGE or CD$ or
WORM$ or DIGITAL STORAGE or COMPACT DISC$ or COMPACT DISK$) and
(PC$ or PERSONAL COMPUTER$ or MICRO$)

There is also another facility offered by TOME (TOME SELECTOR) which directs the user to the appropriate databases, depending on the query.

Another system that helps in query formulation is CANSEARCH (Pollitt, 1986). It contains knowledge specifically relating to cancer and queries on the subject. CANSEARCH was designed to assist users in forming valid search strategies for users to the CANCERLIT database on subjects relating to cancer therapy. The database uses MeSH headings which can be difficult for first time or infrequent users to adapt to. For example, below is an input statement with the corresponding MeSH query output.

Eg: FU in the treatment of breast cancer

"SUBS APPLY DT
1: BREAST NEOPLASMS
"SUBS CANCEL
"SUBS APPLY TU
2: FLUOROUACIL
"SUBS CANCEL
1 AND 2
3 AND HUMAN

The example shows the type of form of query that would be expected of "non-expert" users of CANCERLIT.

The system contains knowledge about how cancer related queries are structured, such as the fact that they deal with types, sites and forms of therapies. These are also implicit in the frames and rules encoded. A simple rule, reflected on the touch-screen, for forming a MeSH query might be as follows:

Eg: IF 'ear' is selected
THEN deselect 'ear'
write 'ear neoplasms' on primary site board

Additionally, the system uses a touch-screen interface to help reduce spelling errors and input time and costs.

Rada and Barlow (1991) refer to a different approach to using thesauri. According to this, a thesaurus can be viewed as a graph and an IRS can exploit this graph when both the documents and query terms are represented by thesaurus terms.[17] This strategy involves measuring the distance between query and documents, through path lengths in the graph. Work with similar strategies indicated that although the hierarchical relations in a thesaurus were useful, non-hierarchical relations were not. With the strategy they propose, Rada and Barlow suggest that a retrieval algorithm can benefit from such non-hierarchical relations. The model behind their approach is one which likens the thesaurus to a mental, semantic network, and the retrieval algorithm to a mental, spreading activation (where various relations are traversed).

Raghavan and Jung (1989) suggest that there might be gains in constructing a pseudo-thesaurus from user relevance feedback with respect to queries. A pseudo-thesaurus contains term-term relationships based on user feedback.[18] Their method involves a learning approach for constructing a pseudo-thesaurus given instances of positive and negative classes. For this, they use a single-layered perceptron (neural network) approach.

TEGEN (Güntzer et al., 1989) is a system with a slightly different motivation. It is based on the argument that users use many relationships between concepts a long time before they appear in the literature or classification schemes. Therefore, this system draws conclusions from search behaviour about possible thesaurus entries. This is done through an iterative knowledge acquisition process in which the intermediate results are acquired through various production rules.

---

[17]Rada and Bicknell (1989) also worked on ranking documents with a thesaurus by estimating the conceptual closeness between sets of terms from the thesaurus.

[18]This is different to pseudo-classification where term frequency information is replaced by term co-occurence distribution by user relevance judgements.

2.5 Organising the Concepts in Documents

RESEARCHER (Lebowitz, 1987) is a prototype intelligent information system which accepts natural language input relating, particularly, to device patent abstracts. It is based on an approach that aims to capture meaning in documents. The purpose being to resolve ambiguities which arise during text processing. This system is based on the model of information systems which view question answering as their purpose. To this end, various phrases in documents are analysed and *concept* (usually object) hierarchies formed from them. Below are examples of the ambiguities and questions in documents relating to patent abstracts.

Egs:

"A metal drive cover..." - Does the modifier (metal) apply to drive or cover?

"A disk drive including a disk with a metal plate and..." - Is a metal plate part of a disk drive or disk?

The operation of the system has three main processes. The first relates to parsing the new patent abstract and forming a concept hierarchy from this information. The second involves placing the hierarchy in the appropriate place in memory. The third relates to the resolving of ambiguities that may arise, for example, in further abstracts.

The system processes patent abstracts by using basic syntactic rules to identify objects in the eventual representation. The text processing algorithm involves identifying object descriptions (usually noun groups) and connecting them with various relational words (usually prepositions as patent abstracts do not tend to contain many verbs). These relational words indicate the various physical, functional and assembly/component relations. Places of ambiguity are identified and the memory is queried for resolution. Memory is asked which of several possible constructions is more likely or what relation is likely to occur between two objects. These questions are answered by looking for examples where the possible configurations already exist in memory. Any remaining unresolved ambiguities are marked until resolved later.

These objects or constructions are then stored in memory. The main point is that memory is automatically generated and no previous input from users or experts is used. The basic approach to memory is to store objects in terms of a hierarchy of automatically generalised prototypes created by noticing similarities among representations. Information in the generalisation nodes/hierarchies is inherited by the lower level generalisations and examples. Following is a simple hypothetical example of a generalisation hierarchy that may be automatically generated and stored in the memory:

Eg:

disk-drive
- floppy-disk-drive
  - single-sided-floppy-disk-drive
  - double-sided-floppy-disk-drive
- hard-disk-drive

However, hierarchies may not always so simple: they may not always be of binary form; the same object can be stored in more than one place and have other generalisation hierarchies. For

example, a hard disk drive can also be referred to as a high density disk drive. It could also have another generalisation hierarchy relating to its components such as a read/write assembly, spindle, motor. Additionally, complexities may arise in memory due to the order of the examples provided - which may not always be in their optimum form.

The design of the system involves three components; generalisation-based memory, memory-based text processing and response to users. The first two relate to the above described processes. The last component relates to how the system might tailor its responses to the needs of inidividual users. The emphasis is on the user's level of expertise. To this end, the attempt is to determine some basic answering strategies which are appropriate for 'expert' and 'naive' users of the system.[19]
In order to establish the different strategies used by users, text aimed at the various types of users was analysed. The type of difference is found in encyclopaedias. For example, adult encyclopaedias may be aimed at relative experts while junior encyclopaedias may only give general descriptions of processes.

The following two descriptions of telephones show the difference between the terminology and level of detail expressed in an adult encyclopaedia and a junior one.

Eg:

> Adult: The hand-sets introduced in 1947 consist of a receiver and a transmitter in a single housing available in black or coloured plastic. The transmitter diaphram is clamped rigidly at its edges to improve the high frequency response. The diaphragm is coupled to a doubly resonant system -a cavity chamber- which broadens the response.
>
> (Collier's Encyclopaedia, 1962)
>
> Junior: When one speaks into the transmitter of a modern telephone, these sound waves strike against an aluminium disk or diaphragm and cause it to vibrate back and forth in just the same way the molecules of air are .....
>
> (Britannica Junior, 1963)

The main conclusion from these type of examples is that those texts aimed at relative experts tend to describe the part structure of objects. Texts aimed at more naive or junior users, on the other hand, focus on the processes that take place in the device.

Another system employing similar strategies to RESEARCHER is UNIMEM, also developed by Lebowitz (1987). This system forms concepts of information aimed at those finishing USA high-schools and wishing to continue with further education. It does this by recognising regularities in the data. Hence, the aim is to assist with students' decision-making relating to their careers.

---

[19]This reference should also refer to expert and naive users in the field of device patents and their terminology, not just in the RESEARCHER system.

2.6 Query Formulation, Expansion and Relevance Feedback

Query formulation and expansion can vary according to the retrieval technique used in an IRS. Retrieval techniques compare queries with document representations. Belkin and Croft (1987) have developed a classification of retrieval techniques which is illustrated below in Fig. 2.7. However, as they do not relate strongly to this thesis, not all the categories will be discussed. The classification is used as a framework for the techniques that will be referred to here. As with most classifications, there will always be some techniques which may not fall solely into one category.

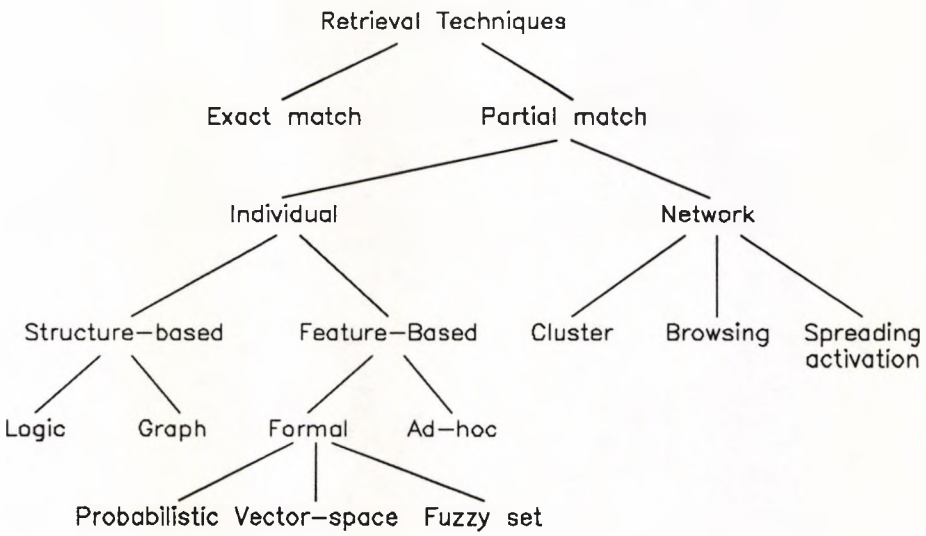


Fig.2.5: A classification of retrieval techniques

Briefly, the first broad distinction is whether the set of retrieved documents contain those whose representations are an exact match with the query (e.g. Boolean searching systems) or a partial match with the query[20]. The set of retrieved documents in a partial match include those which are an exact match with the query and are also ranked in some way. A subsequent classification of the partial match retrieval techniques is based on the connection between the document representatives and the query. Either an individual document representative or a representation emphasising the connections of the document to others in a network is used. The individual feature-based techniques[21] can be formal or adhoc (i.e. similarity measures). It is mainly the formal techniques and the probabilistic approaches, in particular, that is focused on here.

[20]Croft (1986) proposes a method of integrating Boolean queries with probabilistic retrieval models but generally IR techniques fall into either one of these categories.

[21]In feature-based representation, documents and queries are represented as sets of features i.e. index terms.

In order to help overcome some of the limitations of the Boolean systems[22] IR researchers have developed various partial (or best match) techniques. Of these the probabilistic and vector space models have tended to be more successful (Efthimiadis, 1992). Both have similar basic aims in that they aim to retrieve documents in order of their probability of relevance to the query. Both can be expanded to make use of user relevance feedback. However, the vector space model does not provide a formal justification for the particular form of weighting and ranking function that is used, whereas the probabilistic model does.

Probabilistic Models

An essential part of the probabilistic theory of retrieval is the Probability Ranking Principle (Robertson, 1977). This principle states that performance in terms of effectiveness measures will be optimal if documents are ranked according to their probability of relevance (to the query or underlying information need), according to the information available to the system. This principle holds providing that certain assumptions are made about the statistical properties of the variables involved. Robertson, Maron and Cooper (1982) point to different interpretations of "the probability of relevance" depending on whether the emphasis is on indexing or searching.

Work on probabilistic techniques in IR began as early as the sixties. Maron and Kuhns (1960) in their theory of *probabilistic indexing* introduced the notion of relevance. They refer to a relevance number which is a measure of the probable relevance of a document for a user (after a particular query). This number provides a means of ranking documents according to their probable relevance. However, they also point out that the IR problem involves 'properly' selecting the documents which are to be ranked. Thus, they define various measures of closeness between documents and between queries. Their technique of probabilistic indexing is based on giving weights to index terms so that "the information content of a document can be characterised more precisely". These weights are then used to compute the relevance numbers. Others who have also worked on probabilistic indexing include Bookstein and Swanson (1974), Harter (1975), Fuhr and Buckley (1993).

Probabilistic indexing, however, is different to probabilistic searching[23] for which there are differing assumptions. In the more recent work on probabilistic indexing, Fuhr and Buckley (1989) introduce the concept of a *relevance description* as a form of abstraction from specific term-document relationships. Instead of estimating probabilities for term-document pairs they do this for the relevance descriptions. Thus, through this abstraction, the estimation of the index

---

[22]Weaknesses of Boolean systems (as summarised by Efthimiadis (1992)) include that:
- there is a lack of control over size of output from a particular query.
- retrieved records are not ranked
- relevant records whose representations only partially match the query may be missed
- they require complicated query logic formation, something most users are not able to do well
- they can not be adjusted so that the relative importance of certain terms can be taken into consideration.

[23]For example, in the Maron and Kuhns (1960) model, binary subject indexing is replaced by weighting indexing. In the Robertson and Sparck Jones (1976) probabilistic searching model binary query terms are replaced by weighted query terms. Additionally, the first model effectively groups users together in order to compute a probability of relevance for a given document whilst the latter groups documents together in order to compute a probability of relevance for a given user.

term weights becomes document-independent. The idea of abstracting from relevance data is a form of learning and is also used in the approach described in this thesis - although, the Fuhr and Buckley learning method is based on mathematical principles and does not distinguish between users (as will be shown later in Chapter 5).

A probabilistic model of searching, based on the term occurences in document representations, is the Robertson and Sparck Jones (1976) model. Here, a query term has a value (weight) assigned to it. This weight is derived by

$$w=\log\frac{p(1-q)}{q(1-p)}$$

where

w: weight to be assigned to a term *t* (relevance weight)
p: probability that a document *d* will take *t* as index term given that d is relevant to the query
q: probability that a document *d* will take *t* as index term given that *d* is non-relevant to the query

Although the obvious use of this model is in relevance feedback, in the absence of this, term frequency can be used to estimate *p* and *q*.

The Sparck Jones (1972) collection frequency weighting (inverse document frequency) method assigns a weight by the function:

$$w=\log\left(\frac{N}{n}\right)$$

where

N: is the number of documents in the collection
n: is the number of documents indexed by term *t*

Croft and Harper (1979), on the other hand, propose

$$w=\log\left(\frac{N-n}{n}\right)$$

The above formulae also form the basis of the Okapi probabilistic retrieval system used as part of the work of this thesis. Further explanations of these formulae, their derivations and use relating to Okapi can be found in Section 6.1.3. Efthimiadis (1992) contains a more detailed survey of probabilistic models. Robertson (1981) and Van Rijsbergen (Van Rijsbergen et. al., 1980) also provide detailed derivations and comparisons between various probabilistic models.

One way of ranking documents in probabilistic retrieval systems is to sum the weights of the individual terms that index a document (e.g. Okapi). Probabilistic models, like most partial match systems, tend to be based on single terms. Most Boolean systems, on the other hand, offer proximity searching (i.e. to search for adjacent words or those which fall in the same paragraph or sentence). In his work on partial match systems, Keen (1991, 1992) proposes the use of term proximity to incorporate the ideas of sentence matching, proximate terms, term order specification and term distance computations. Although his methods are not based on probabilistic approaches they do fit into the same framework. The hypothesis, here, is that term position will act as a precision device. Like the above probabilistic models, a weighted score is used for ranking the document records. This score reflects the number of terms (in a document) that match the query and their proximity to one another in the fields and sentences of the records. The closer the terms are together the higher the match. Most of the algorithms Keen proposed based on this method of ranking, when compared with the simplest output ranking method of counting the number of matching terms, performed better.

Referring back to probabilistic models, Turtle (1991) proposes a probabilistic inference network to represent documents and information needs. This is a probability-based method that follows the probability ranking principle. The argument is that, IR is an evidential reasoning or inference process in which "we estimate the probability that a user's information need, expressed as one or more queries, is met given a document as evidence." This is based on the perspective that significant improvements in retrieval performance can be gained through 'understanding' the contents of documents and queries. An inference network, combines the idea of an inference model and a network model.

The network consists of four basic types of nodes: document, concept representation (concepts that describe the contents of a document e.g. document index terms), query and information need nodes. Document nodes are the roots of the network. Probabilities for each document node are propagated through the network (including the concept representation and query nodes) to eventually derive a probability for the information need node.

The model has also been extended by Haines and Croft (1993) to include relevance feedback. Turtle suggests that this model can also be used to simulate other methods such as Boolean and cluster-based ones and that results produced by these can be combined to form an overall assessment of relevance.

Relevance Feedback

Relevance feedback can be understood in the context of the following stages of the retrieval process:

1) The user states the query
2) The system provides certain documents (representations / descriptions) for evaluation.
3) The user then chooses the relevant ones.
4) This relevance feedback is subsequently used to modify the query.

Thus, relevance feedback can also provide a form of query expansion though it is not necessarily the only means. Relevance feedback can also be used to modify document representations,

although this will not be discussed further. Relevance feedback, its meaning, implications and usage is also discussed in sections 6.1.3, 7.2.2.

A system allowing such weighted searching (term weighting, document ranking and relevance feedback) is the CIRT front-end system (Robertson et al., 1986; Robertson and Thompson, 1989). CIRT uses MEDLINE and INSPEC from the Data-Star host. Other systems using probabilistic theories for retrieval include INSTRUCT and SIBRIS. INSTRUCT[24] (Wade and Willett, 1988) uses an inverted file approach on the LISA database to test for a variety of techniques such as relevance feedback, query expansion, cluster-based searching and browsing. SIBRIS[25] (Wade et. al., 1989) weights the query terms according to their position in the document to be retrieved.

The following systems do not use probabilistic models but they do use relevance feedback. CITE[26] (Doszkocs and Rapp, 1979) enables searching from MEDLINE and IIDA (Individualised Instruction for Data Access). The system performs syntactic analysis, synonym control and stemming on the words input in free form language and then selects the MeSH (Medical Subject heading) terms and dictionary terms. The titles are subsequently ranked and shown to the user for relevance feedback. The query may then be reformulated based on the references approved by the user.

EXPERT (Yip, 1981), which also uses relevance feedback techniques, is a rule-based system designed to automate query formulation in online searching. This system gives suggestions for splitting the topic given into concepts and suggests terms for these concepts. These topics are then transformed into Boolean strategies. The knowledge structure is syntactic and relates mainly to the structures of Boolean queries. Relevance feedback techniques are also used to improve system performance.

CONIT[27] (Marcus, 1986), which is based on the same principles as EXPERT, assists users in accessing MEDLINE, ORBIT and DIALOG databases. Through interacting with a basic self-instruction language the system helps identify appropriate databases and then automatically connects the users to them. EASYNET is a commercial package based on the ideas of EXPERT.

Vector Space Models

In this approach, as shown in the SMART system (Salton, 1983), documents and queries are points in a t-dimensional space for t indexing terms. Each document is represented by a particular vector (of terms). Each term is assumed to be unrelated to others and all terms are considered equally important. Queries and document terms have associated weights (calculated using frequencies of terms in documents and frequencies of terms in collection) that are calculated and stored prior to any search.

---

[24] INteractive System for Teaching Retrieval Using Computational Techniques

[25] Sandwich Interactive Browsing and Ranking Information System

[26] Current Information Transfer in English

[27] COnnector for Networked Information Transfer

For each query, documents to be retrieved are identified by performing similarity calculations betweeen stored items and incoming queries, and by ranking the retrieved items. As a similarity measure, the cosine correlation is used with the weighted query and document vectors. In some experiments, related documents are clustered/collected into common subject classes making it possible to start with specific items in a particular subject area and to find related items in neighbouring subject areas. Other experiments involve the system using relevance feedback to iteratively change the original query by adding terms from relevant documents and subtracting terms from non-relevant documents (or adjusting their weights).

2.7 Natural Language Processing

Natural Language Processing (NLP) involves handling unrestricted written or spoken language with purely "mechanistic" procedures. The techniques vary from those employed in text editors, word processors, automatic indexing in IR to those which aim to understand and express "meaning" for question answering or expert systems (Doszkocs, 1986).

NLP can be done at several different levels such as phonological, morphological, lexical, syntactic, semantic and pragmatic. Briefly, the first deals with speech recognition and generation. The morphological level consists of generating word stems through recognising and removing word suffixes and prefixes. The lexical level, on the other hand, operates on full words. This can involve identifying nouns, adjectives and other lexical features. However, more often in IR it consists of processing words found in dictionaries, replacing words with those in a thesaurus or deleting common words. The semantic level adds contextual knowledge to the syntactic process so that the text can be separated into units which represent its meaning. Lastly, the pragmatic level makes use of additional information regarding the environment in which the document exists. These can relate to the social environment or other available facts which would help text interpretation (Salton and McGill, 1983).

The application of NLP to IR has not been as easy as initially hoped (Smeaton, 1990; Salton and McGill, 1983). There is a problem of dealing with the content of natural-language texts in the absence of a unified theory of language and meaning. The various ways in which the same subject matter can be expressed and interpreted pose the main problem (Doszkocs, 1986; Sparck Jones, 1973). However, perhaps a greater problem is in the differing viewpoints regarding whether it is necessary, in order to retrieve items "about" a certain topic, to have facts pertaining to the topic or the meaning in the documents that may refer to it. Thus, we are confronted with the two differing viewpoints relating to question answering and document retrieval systems referred to earlier in this chapter. In retrieval, the aim is to "render a document retrievable rather than to convey the exact meaning of the text" (Salton and McGill, 1983). Hence, differing viewpoints in items covering the same subject matter do not result in the items being treated differently i.e. some being retrieved and not others. In a question answering system, however, these items would be treated differently.

There is more evidence for the usefulness of statistical, probabilistic or vector space techniques than the linguistic based ones in IR (Salton, 1983; Dumais, 1988). Nevertheless, there has been

some varied and potentially useful work in NLP and so a selected few are mentioned in this section.

Text processing in a question answering system means determining key facts within that text. Thus, a large obstacle to this type of textual analysis is lexical inadequacy. NLP methods require lexical entries for all words encountered in the text. This results in the lexical bottleneck problem. Machine Readable Dictionaries (MRD's) and word learning approaches have been developed to help overcome this. However, as Coates (1992) points out, MRD's do not seem to be the dominant tool. Only 1 of 15 text understanding systems in the MUC-3[28] (DARPA, 1991) experiments claims to have used an MRD.

One approach to help overcome the lexical bottleneck can be to identify and retrieve word collocations. Collocations, in NL, are recurrent (arbitrary) combinations of words that co-occur more often than expected by chance. The Xtract system (Smadja, 1993) is a lexicographic tool which uses statistical methods to retrieve and identify word collocations. These are not only based on the relevance of the word associations (within specific domains) but also contain some functional information to help identify collocations.

Automatic word learning is used in the SCISOR[29] system. It selects and analyses stories about corporate finance, mergers and acquisitions from the Dow Jones online financial service. Thus, it performs text analysis and question answering on this constrained domain.

SCISOR has levels of language analysis (from rough skimming to conceptual interpretation) and also uses a handcoded knowledge base. The Filter (topic analyser) component selects stories and performs the lexical analysis of names, dates, numbers etc. The NL components identify key attributes such as target, suitor, price and company products. Thus, a single representation of each story is then added to the central knowledge base. Future questions are then represented in the same way and matched with the representations of these stories.

Another example of word learning can be found in the FUNES system (Coates, 1992) which learns proper names in samples of news text. Proper names are viewed as a subset of general unknown words and the approach is based on applying constraints on a word's syntactic and semantic class at various levels of linguistic analysis. The system comprises of four interlinking models: the lexicon and knowledge base, the pre-processor, the syntactic parser and the semantic analyser. Input to the system is first passed through the preprocessor and then the syntactic analyser. The lexicon is used to help with both these stages. This is followed by semantic analysis which also uses the knowledge base. The knowledge base and lexicon can be updated with learned words and the meaning acquired for a word can be refined upon subsequent occurrences.

---

[28] 3rd Message Understanding Conference

[29] System for Conceptual Information Retrieval was developed by General Electric, which was the main centre for work on word learning, in the late 80s.

Diverting from the view of IR as question answering, we can mention systems such as IR-NLI[30] (Guida and Tasso, 1983). This system is an interface allowing users to state their request in Natural Language form. Here, incorporated knowledge of search heuristics is used for choosing between the search strategies. For example, the search requirements might be identified as having a high precision search objective or having an off-line preparation for the operation mode, for example. While this system assists with query handling it does also use user frames/models in the process. This is another example of the fuzzy boundaries between the categories of work in IR referred to in this chapter.

Other approaches include those of automatic generation of back-of-the-book indexes (Salton et. al., 1990), automatic construction of document abstracts from text (Paice, 1990), dealing with nominal compounds[31] (Gay and Croft, 1990) and anaphoric references[32] (Liddy, 1990).

2.8 Improving the User Interface

Work on improving the user interface for organising and searching information in IR involves several fields such as graphical design and cognitive psychology. The methods for designing interfaces are varied and include the application of hypertext, direct manipulation of objects and various cognitive tools. However, the benefits of these approaches are not too clear mainly as they usually have not been subject to systematic evaluation (Dumais, 1988). The purpose here is to highlight some of these methods with the view of applying them in IR. More detailed work on the theory for guiding the IR design process has been focused on by others (Sonnenwald, 1992; Marchionini, 1992).

Most of the new developments in IR have been aimed at better representing the relations among "objects" or entities, often to supplement a hierarchical organisation with cross-links between nodes. A node often refers to some unit of information (not meaning units but text, graph picture etc.) Hypertext or non-linear text refer to systems which aim to encourage such more flexible exploration, search and manipulation of information and ideas (Conklin, 1987; Agosti, 1990; Frisse and Cousins, 1992). It presents an alternative to a linear order and thus it is hoped that users can have more choice on the decision making, ordering, depth and coverage, cross-reference to different ideas, different levels of detail and different media (such as text, picture, graphs and voice).

Information is accessed by navigating through the network by following the particular links or browsing. Link-based navigation can be supplemented with content-based mechanisms. i.e. Users can also go directly to notecards by specifying keywords without navigating through the system (Halasz, 1987)[33]. These systems also enable to represent and manage subsets of nodes and links as entities themselves.

---

[30]Information Retrieval - Natural Language Interface

[31]A nominal compound is a sequence of two or more nouns that together form a structure which itself acts like a noun e.g. "information science".

[32]Anaphora are abbreviated subsequent references such as pronouns.

[33]Such systems are driven by similar concerns as menu-based systems enhanced by keywords.

As users often have difficulties in navigating through complex structures, maps can help show where they are. Some systems provide browsing and modification facilities for sections of a network with graphical display (Halasz et. al., 1987). Other graphical possibilities are three-dimensional views of the network (Fairchild et. al., 1987), zooming and fisheye views of large structures (Furnas, 1986). A problem with graphical interfaces is having meaningful abbreviations and icons which help users navigate throught the system.

Hypertext systems can also include dictionaries, encyclopaedias, theses, textbooks, document databases. In this respect, they can be used in the storage and management of information in different forms and medium.

Other improvements with interaction styles can be in menu selection systems (Pollitt, 1989), better screen design, semantic organisation of menus, improvement in command languages (e.g. at the moment there is no standard Boolean interface for host databases) and natural language processing (Shneiderman, 1987) (see also Section 2.7). Interaction devices such as touch screen, pointing devices, function keys also play a role in user interfaces. The most recent of these is the various devices that form the notion of 'virtual reality'.

The use of graphical interfaces (i.e. windows) in IR systems should not simply be a 'beautified' version of a menu driven system. Just as it is the case for other computer systems, the nature of the human-computer interface for IR systems can greatly affect the way in which retrieval is done. However, it is important that in using these new tools new ways of communication and retrieval are also explored.

2.9 Other Methods

Other approaches include those borrowed from AI such as neural networks, connectionist methods and genetic algorithms. Neural networks and connectionist approaches tend to be suited for term association and clustering tasks e.g. Wong (1993). The next chapter on learning techniques also refers to these.

Other developments have been more to do with improving the physical storage and retrieval of text. Examples include, compression techniques for the increasing number of text files and databases (Linoff and Stanfill, 1993; Witten, 1993; Bookstein et. al., 1993) and the use of increasingly available hardware for parallel processing to increase computation speeds and 'data' access. Parallel processing is an example of how low-level techniques can make some approaches more feasible. For example, can help with the application of neural networks. It can also be applied to query expansion techniques.

One such example is the CONNECTION MACHINE[34] (Durham, 1989). Although this involves considerable computer power, it improves search speeds considerably. Hence, it is possible to "broaden" a search by finding other synonyms for the terms referenced in the documents

[34]The Connection Machine was developed by Thinking Machines Incorp.

retrieved from the original specification. The synonyms are found by searching through the initially referenced documents and identifying where two or more words have been used in the same way in the same document.[35]

In addition to the methods discussed in the previous sections, more work is also being done on hybrid systems, combining methods and techniques based on different approaches or theories.

[35]Eg1: If "OPTICAL DISK" and "CD" happens to be contained in the same document the search can be broadened. Eg2: Upon requesting information about share in a certain company, through extensive cross-referencing the system could trace names of companies relating to take-over bids. It would do this when it found a document that held information about the shares and also contained the name of the company(ies) involved in the take-overs.

Chapter 3

Machine Learning

This chapter discusses some of the machine learning techniques in AI with a view to analysing how they may or may not be relevant to IR.

The ability to learn is one of the most important components of intelligent behaviour. Thus, as a branch of AI, machine learning (ML) is the field of inquiry concerned with the processes that constitute learning behaviour. It is generally thought that an "improvement" of the system is one of the conditions for *machine learning* e.g. Minsky's definition:

> "Learning is making useful changes in our minds."
>
> (Minsky, 1985)

However, this definition (as Minsky also notes) is too general for describing learning; it requires more explanation as to what and when something is considered to be an "improvement". Simon (1983) gives the following, more precise, definition.

> "Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently next time."
>
> (Simon, 1983)

However, this definition omits situations where learning is applied to similar or new tasks. There are also views that the "improvement in performance" in subsequent repetitions of these tasks is the *consequence* of learning rather than the "process itself and that learning is the process of constructing a representation of reality" (Michalski et. al., 1986).

ML can also be viewed as a possible solution to the famous "knowledge engineering bottleneck" problem often encountered in building expert systems (Feigenbaum and McCorduck, 1983). This is the situation where the effort spent to encode experts' knowledge, which may need frequent updating, can prove to be very costly both in terms of time and money. Savings could be made with programs that learn from their own experience, planned experiments or high-level advice from humans (Luger and Stubblefield, 1989). Hence, the learning process involves *inducing* and *refining* the knowledge in the system.

By refocusing their efforts on learning, many researchers hope to discover more general principles of intelligence. Thus, general learning models might help in the automatic construction

of knowledge intensive systems (Langley et. al., 1986). This chapter presents the learning models in the context of both a human-oriented analysis and computation-oriented analysis.

3.1 A Human-oriented Analysis of Learning

Machine learning has tended to model itself on various types and aspects of human learning. These include the notion of learning by 'simply' repeating or copying, learning (by being told) from an authoritative source, learning by moving from specific to general (induction/abstraction) or vice-versa (deduction) and learning through analogies. There are various ways of categorising learning approaches but the one chosen here is that which was defined by Carbonell (Carbonell et. al., 1983) and follows the above mentioned general categorisations. The categorisations that follow are not meant to be a statement of how humans learn but are simply used as a metaphor. The categories are as follows:

- learning by rote
  learning by 'simply' repeating or copying
- learning by being told
  learning from an authoritative source
- learning by example
  learning by moving from specific to general (induction/abstraction)
- learning by analogy
- learning by observation and discovery
  learning by moving from general to specific (deduction)

The boundaries between these categories are not always clear and some forms of learning can fall into more than one. Other researchers have tended to make categories according to existing learning techniques. For example, Langley (1993) identifies five general types of techniques such as inductive learning, deductive learning, genetic algorithms, neural networks and case-based learning. Such specific techniques as genetic algorithms and neural networks, for example, can be part of various learning approaches (i.e. learning from examples or by analogy). Further alternative approaches to categorisation can be according to the representation of the knowledge acquired by the learner (decision trees, schemas, formal grammars etc.) or the application domain concerned (Carbonell et. al., 1983). It is not the point of this thesis to develop a fully comprehensive classification system for learning approaches and techniques. However, within the framework set out in fig.3.1 various techniques will be discussed briefly along with their applicability to IR. The aim is to highlight the extent and depth of the work done in machine learning which may be of use to the field of IR.

3.1.1 Learning by Rote

Rote learning is the simplest type of learning. Here, the system simply relies on the memorisation of solutions or cases which are stored for future retrieval should the same situation or problem arise. There is no inference procedure (Carbonell et. al., 1983).

As an example, rote learning in a "mouse in a labyrinth" type problem could be done through simply memorising all the paths taken and noting which ones lead to dead-ends. The 'mouse' makes a quicker exit from the labyrinth after each iteration, by simply not repeating the same

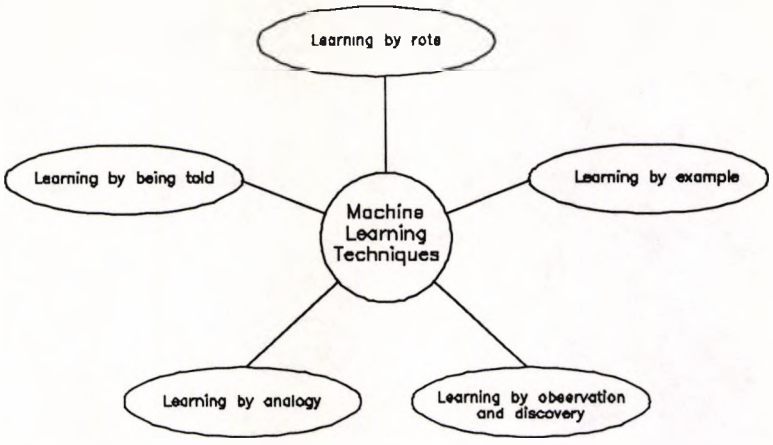


Fig.3.1: Machine learning strategies.

'mistakes' and using routes that were previously known to be available.

Learning by rote can also be done through (explicit) programming in which case the main effort is not made by the learner program but the person doing the encoding.

Despite the simplicity of this approach, if too many instances are stored, the cost of selecting a stored solution when confronted with new data can lead to a degradation of program efficiency/speed. The large number of cases that have to be stored make it necessary to have very efficient indexing methods. Alternatively, this could be overcome with the use of heuristics (Luger and Stubblefield, 1989).

Samuel (1959), used a form of rote learning for part of his famous draughts program. Not only could it play a game of draughts but it could also beat its oponents. In this system, the problem solving component involved searching the draughts game trees. Various values were then propogated back in order for the program to choose the next best move.[36] The rote component function ensured that, in later games when the same situations arose, previously computed and stored values were used (Samuel, 1959; Rich, 1983).

However, even with such a relatively simple strategy, there is some organised storage of information, generalisation and direction required. Firstly, organisation should ensure that retrieving a previously calculated value, for example, would be faster than recomputing it. Some form of generalisation is necessary to keep the number of objects or permutations stored to a

[36]Each tree consisted of a series of board positions (positions are reached with a certain number of moves). All levels in the trees were not searched as it would involve a lot of processing time and the number of levels depended on the situation. When confronted with a new move/situation the strategy was to search as low down the tree as possible (until the last node) and then apply a static evaluation function for that position. This was then used in continuing searching the game tree. After searching the game tree, these values were propagated back and a resulting value for the board position at the root of the tree was recorded. Thus, it could then choose the best move.

manageable size. Lastly, direction or decision making strategies are required to be able to choose intelligently from the (sometimes very large number of) possibilities (Rich, 1983).[37]

Rote Learning and IR

In IR, rote learning could be used to store user queries and a reference to the documents users may have seen. However, the extent to which this alone would be useful in helping with their subsequent queries is debatable. Reminding users of their past queries and whether they have previously seen a document may clutter what they are trying to achieve in the current query.

More importantly, there is no such thing in IR as revisiting the same (identical) 'place'. In rote learning, actions/moves for particular situations are recorded on the assumption that these situations will reoccur. In IR, even if the same query is repeated and the users were presented with an identical document list, one can not assume that the same relevant documents will be chosen. Even if the same user repeated the query and saw the same document list the information need can not be assumed to be exactly the same.[38]

3.1.2 Learning by Being Told

In *learning by being told* (or *learning by instruction*), the training and the high-level of information is provided by the teacher. The advice is transformed into machine understandable statements, knowledge structures and operations that relate to what the system has already learnt. The learning system should accept instruction or advice and then store and apply this learned knowledge effectively.

The source of the knowledge need not only be a teacher but could also be another organised source such as a textbook. Nevertheless, there still need to be some representational changes in order to transfer the knowledge from the input language to that which is used internally by the learner. Thus, this approach requires not only a correct understanding of the advice but a correct translation to the internal representation and an integration of this knowledge into the current knowledge base as well. The learner does have an inferencing mechanism but most of the effort still lies with the teacher (Luger and Stubblefield, 1989; Carbonell et al. 1983).

In this approach, there are some processes performed on the input but no generalisations or predictions are made. The input (applicable to a certain domain) is 'compiled' in some way to make it more efficient. This, in turn, could be integrated with what the system already knows.

Porteous (Porteous to be publised), uses this approach in planning domains.[39] In this case, the various objects and actions involved in planning are supplied by the user. The program makes some representational changes on these and performs compilation on them to make them more efficient.

---

[37]For example, in a games playing situation, this could be a simple criteria such as favouring positions that are reached with the least number of moves.

[38]The concept of information need was discussed earlier in Section 2.1.

[39]The planning domain was required as part of an air traffic control system.

Michalski and Chilausky (1980) in their system, compared expert-derived rules and those induced from case histories for the task of soybean disease classification. Plant pathologists were consulted in obtaining diagnosis rules for (15) soybean diseases. These expert-derived rules are the 'learning by being told' component of the system. Additionally, inductive diagnosis rules were derived from the training instances provided.[40] This part of the system is a form of learning by example (also discussed later in this chapter). Their results indicated that the inductively derived rules were superior even though they may not have been as expressive/clear. In terms of the classification of the diseases, there was not much difference between the two, except that the inductive rules were more decisive.

From this perspective of rule induction/encoding, one could argue that expert systems are a way of learning by being told. Although the rules and facts both making up the knowledge base in an expert system are subject to the inferencing mechanism of the programming language they are written in, the extent to which they are 'compiled' and transformed to improve system efficiency is variable.

Learning by being told, like most other methods, can also be applied to connectionist systems (Diederich, 1989). Connectionism is also referred to in section 3.2.3. For example, an instruction is initially expressed in a description language. It is then compiled into a script file which can be read by a connectionist simulator or a connectionist knowledge representation system. The result is an input to the connectionist network which changes in order to integrate new knowledge.

Learning by being told and IR

There is scope for this type of learning in IR, particularly in relation to using user models to help with users' retrieval requirements. Users' profiles/classifications, as seen appropriate by a teacher or model, can be defined and put through the learner to help with their subsequent information needs, for example. Text routing (SDI profiles) could also be regarded as a form of *learning by being told*, where profiles are constructed by a tutor/intermediary for a user to help with their current awareness searches.[41]

3.1.3 Learning by Example

In *learning from examples* the system proceeds from individual cases and develops classifying rules and general principles from them. The resulting general description should explain all positive and exclude all negative examples of the target concept. This method condenses a large knowledge-base into effectively a smaller one thereby enabling more efficient searches of the knowledge-base. This type of induction of general principles from the set of relevant examples is one of the oldest and most frequently used approaches in ML.

---

[40] 290 training instances were chosen from 630 diseased plant instances. Diseased plants were described in previously defined categories which contained specifications of the weather, time of year, the surrounding geographic conditions and their severity and details of the particular soybean plant leaf and root system conditions. This system had the capability of learning both mutually exclusive classes and (more usefully) overlapping classes hypothesis. An algorithm was devised for discriminating among many classes.

[41] At its simplest, text routing (routing queries) could also be seen as a form of *programmed rote learning*.

This type of learning can be either *incremental* where the system response is modified with each training instance, or *single trial (all-at-once)* where concepts are formed once in response to all the training data. It can also involve *instance-to-class* and *part-to-whole generalisation* (Michalski, 1986). In *instance-to-class* generalisation, "the system is given independent examples of some class of objects, and the goal is to induce a general description of the class". Whereas in *part-to-whole generalisation*, "the task is to hypothesise a description of a whole object, given selected parts of it".

This approach can also be further sub-divided (Carbonell et. al., 1983) according to the nature or source of the examples. In a sense, the last two of these are orthogonal to 1, 2 and 3:

1. The teacher
2. The learner itself
3. The external environment
4. The provision of only positive examples
5. The provision of both positive and negative examples.

1. The teacher knows the concept and generates ordered examples which are meant to be as helpful as possible in order to converge on the desired concept. The teacher can generate the examples or select them from existing data. Additionally, if the teacher knows or infers the knowledge state of the learner this can be used to help optimise convergence on the concept.

2. The learner generates the examples and gets an outsider (teacher or the environment) to classify them as positive or negative examples (examples/counter-examples). It knows its own knowledge state but not necessarily the concept to be acquired. So after cross-referencing the instances are re-examined to see which category they fit into.

3. The learner relies usually on relatively uncontrolled observations. To a certain extent this is the case with the *context learner* in the sense that the user provides a random set of positive examples (relevant documents). Here, however, these are filtered to include only the ones deliberately marked as positive. This situation is encountered when the learner can not know a priori when and where the concept will occur (even if it knows what the concept to learn is) i.e. not knowing when and where positive examples for a certain user's *context* will occur, but simply knowing that they will occur at some point.

4. The learner relies only on positive examples. However, positive examples do not provide information for preventing overgeneralisation of the inferred concept. This may be avoided by keeping to only the minimal generalisations or by using a priori domain knowledge to constrain the concept to be inferred.

5. The learner uses both positive and negative examples. The positives make the learner tend towards generalisation whilst the negatives prevent overgeneralisation, bearing in mind that the resulting concept should not be so general so as to include any of the negative examples. Research in ML has tended to concentrate on this method.

The early work of Winston (1975) consisted of using a series of positive and negative examples to learn the concept of an "arch" in a blocks world. The training instances consisted of blocks world structures that fit in the category, along with "near misses" (structures that almost fit in the category but failed on one or two aspects).

Eg: After some initial examples, the concept could be
"An arch consists of two vertical blocks and one horizontal object". Later, the two "near misses" below are provided.

1) Two vertical blocks that touch each other and the horizontal object on top.
2) Two vertical blocks that do not touch each other and the horizontal object not on top of the two i.e. on the side.

This results in the following concept

"An arch consists of two vertical blocks that do not touch and a horizontal object that rests atop both blocks."

Thus, new positive instances not covered by the current hypothesis indicate that the concept being formulated is overly specific while new negative examples covered by the hypothesis indicate that it is overly general.

Another example, is the ID3 algorithm (Quinlan, 1986). Given a set of examples, ID3 induces an optimal decision tree for classifying instances. The decision tree (classification rule) is derived from examples relating various features of domain objects to their classification.[42] In order to build an optimal tree, ID3 first ranks all the features in terms of their effectiveness in partitioning the set of target classes. When all branches lead to single classifications, the algorithm halts. The tree can determine the class of any object from its features (or values of its attribures) and does this by checking the fewest features.

[42]Objects are described in terms of a collection of attributes. Each attribute measures some important feature of an object.

For example, in the domain of chess endgames, the concepts to be learned can be "lost in one move", "lost in two moves" etc. However, below is a simpler example that shows the decision tree derived given a set of examples.

Eg: The table below lists the relationships between species of animals and four features (diet, size, colour, and habitat).

diet	size	colour	habitat	species
meat	large	striped	jungle	tiger
meat	large	tawny	jungle	lion
meat	small	striped	house	tabby
meat	small	brown	jungle	weasel
grass	large	striped	plains	zebra
grass	small	grey	plains	rabbit
grass	large	tawny	plains	antelope

Table 3.1: Examples for classification, taken from Luger and Stubblefield (1989).

Not all of these features may be necessary for distinguishing classes. For example, 'size' is not needed. Similarly, once 'brown' or 'grey' branches of the tree are taken, the remaining features can be ignored. i.e. 'colour' alone is sufficient to distinguish rabbits and weasels from the other animals.

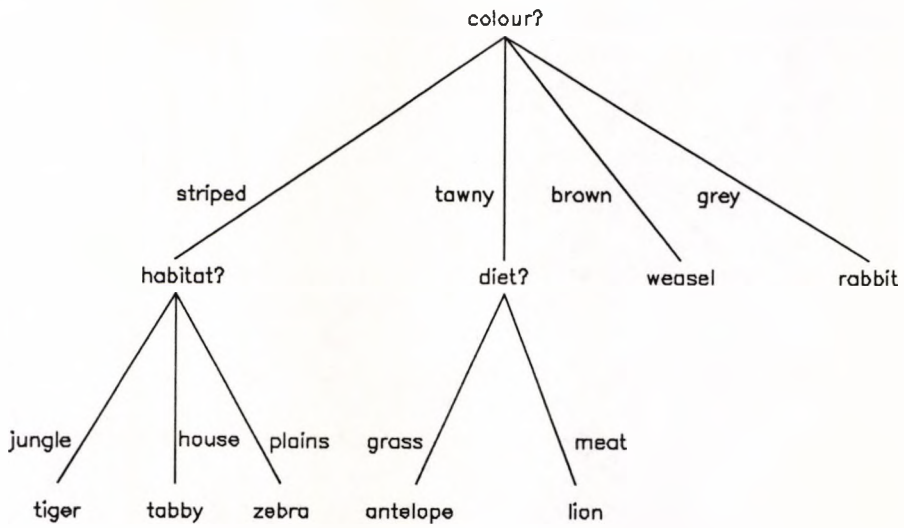


Fig.3.2: Optimal tree diagram produced by ID3.

However, ID3 is limited by the expressiveness of the decision tree representation. *Version space searches* (Mitchell, 1979) have been developed to help overcome these limitations. Given positive and negative examples of concepts, version space searches learn general concepts. In order to limit or focus the search, a partial ordering is made from the generalisation operations. Upper and lower bounds are set in the concept space to form the boundaries of the version space. The upper bound is set by identifying the set of most general descriptions that cover all

positive and none of the negative training examples. The lower bound is set by the set of most specific descriptions that cover only the positive instances.

In learning from examples, there are two techniques for guiding and constraining generalisation: the similarity and constraint-based techniques. The first explores inter-example relationships in that it examines the positive and negative examples of a concept in order to create a concept description. Constraint-based techniques, on the other hand, explore intra-example relationships, which constrain the explanatory concepts applied to the facts/examples (Michalski, 1986). Any generalisation of the examples must satisfy these constraints. Variants of this concept include the justification for a hypothesis and explanation-based learning/generalisation. Much of the work in learning from examples has tended to focus on similarity-based learning and explanation-based learning. The descriptions of these two approaches and their applications in IR follow. The two techniques can be combined, although they are referred to individually here. Both proceed within a hypothesis space and do not break out of this space.

Similarity-based learning

In similarity-based learning or generalisation (SBL/SBG), common features between examples and counter-examples in the same class are determined and explanations are derived as to why different examples belong to the same class.[43] It deals with the differences by generalising over them either by ignoring them or by encompassing these differences (Michalski et al., 1986). The technique relies on some form of inductive bias to guide the generalisation. Winston's program (1975), mentioned earlier, which learns structural concepts of shapes, is an example of SBL.

SBL and IR

A limitation of SBL is that it needs several "training instances" before it can converge on a useful concept. Although, there is more control over this in IR test collections, having a 'sufficient' number of examples can not be guaranteed in interactive IR. Also, it may not always be possible to clearly categorise positive and negative examples. (See also comments on 'learning by examples and IR'). Nevertheless, the idea of exploiting similarities in features of documents and queries is not new in IR and there is potential for applying the principles of SBL to IR.

Explanation-based learning

Explanation-based learning (EBL) (DeJong and Mooney, 1986; Kedar-Cabelli and McCarthy, 1987; McCluskey, 1990; Mitchell et. al., 1986), addresses the problem of trying to learn meaningful generalisations from a single training instance. *Explanation-based generalisation (EBG)* is representative of the family of EBL approaches that formulate generalisations by constructing explanations.[44] In this technique, the *domain knowledge/theory* provides the means of constraining the search. The theory is provided along with various instances or examples of this theory. The system must explain to itself why the training example is an example of the theory or concept. Although the resulting definition or rule from this form of learning is more general than the specific examples provided, it still is more specific than the theory.

---

[43]The term SBG was suggested by Lebowitz (1985).

[44]The term was suggested by DeJong (1981).

There are four parts to an explanation-based learner. Namely, a *goal concept*, a *domain theory,* the *training example(s)* and the *operationality criteria*. The *goal concept* is the concept that the EBL is to learn. The *domain theory*, is the domain-dependent knowledge expressed in "rules", for example. The *facts* or *training examples* are used in generating the concepts. As mentioned earlier, explanation-based learners can perform learning even from one training example. This is because the domain theory is assumed to contain sufficient information to prove that the example is an instance of a more general concept.

In EBG, the above mentioned four components are in Horn clauses. The task, as mentioned earlier, in EBG is to prove that the training example is a member of the target/goal concept. This is done by using the domain theory and then generalising the proof to form an *operational definition* of the target concept. In other words, it is the process of reformulating the goal concept in terms that satisfy the operationality criterion, with the domain theory providing the means for re-expressing the goal concept.

Appendix G (Explanation-based Learning - An Example) contains two examples of this type of learning. The examples are based on "real world" facts which have been trivialised. Despite this it is not obvious how it would form part of a useful system. The equivalent IR example would have to be about the user's subjective requirement for documents - either very trivial and obvious or not generalisable.

Mitchell et. al. (1986), in their discussion of building systems that automatically formulate their own generalisation tasks and inputs (goal concept, domain theory and operationality criteria) refer to the SOAR (Laird et. al., 1986) and LEX2 (Keller, 1985) systems.

To solve a problem, SOAR divides it into subgoals. It then uses "implicit generalisation" (a technique closely related to EBL) to infer subgoal preconditions. In other words, each time a subgoal is solved, a generalisation task is formulated. This task is to infer the general conditi ons under which the solution to the subgoal can be reused. The system is based on the chunking hypothesis[45] and forms 'chunks' from the results of goal-directed problem-solving behaviour using rule-based representations.

In LEX2, a subgoal (such as USEFUL-OP3) becomes the goal concept in the process of planning to improve performance at solving calculus problems. The subgoal is one of introdu cing a filter to reduce the search moves that it considers. The filter's purpose is to allow only 'useful' problem solving steps i.e. those that lead towards solutions. Keller refers to this as "contextual learning". However, this 'context' is more akin to a problem solving domain rather than the *user's context* referred to in this work. METALEX (Keller, 1987) takes the contextual learning further and evaluates the degree of operationality[46] of the concept definition in relation to the performance task and objective.

---

[45] The chunking hypothesis (Rosenbloom and Newell, 1986) states that "a human acquires and organises knowledge of the environment by forming and storing expressions, called *chunks*, that are structured collections of the chunks existing at the time of learning".

[46] This is calculated by measuring effectiveness (percentage of the problems solved) and efficiency levels (CPU time).

EBL and IR

In IR, this technique is rather difficult to apply as even though training examples can be found or provided, it relies on a definition of the domain theory and the target concept. Defining 'the domain theory' of IR is not possible (although various IR theories were discussed in Section 2.1) nor is it clear exactly what the 'target concept' should be for a particular search. Is it to find any references that mention the words/terms in the query? Is it to simply browse the database? Would this definition depend on what the user thinks his/her information need is at a particul ar point in time? Or should more objective criteria apply? Can the definition (no matter how it's arrived at) be taken to be true? The uncertainties surrounding these questions highlights the need for a strong theory in EBL.

Pazzani (1989) addressed the problem of EBL in weak-domains. The 'weaknesses' he refers to are when statements in the domain theory are not necessarily known to be true. However, there still is a domain theory which can be covered (in some way) by the examples provided. Thus, the task is one of proving or disproving the statement with the given facts/examples. There is the guarantee that what you are looking for is in the examples.

Assuming that you could represent retrieval tasks or information appropriately for EBL, in IR, it would still be unclear as to whether the asserted domain theory (however weak) is related or, more importantly, usefully related to the set of examples provided. Pazzani (1989) does also test his idea of applying EBL in weak domains further by applying it to situations where no domain theory is specified. Even so, there is still the underlying assumption that a useful or relevant domain theory can be derived from the provided examples. Although something may be derived, it is not clear what exactly it would mean in terms of the retrieval or usage of information. Also the ordering of these examples can determine how quickly you would converge on such a theory.

In IR, it is difficult to define a 'domain theory' in terms of specific rules. Like all *learning by example* techniques, the resulting inference depends on the what is inherent in the provided examples in the first place. In AI, the order of examples may affect the efficiency of the inference and the speed at which it is arrived at. However, in general it is not likely to alter the meaning. On the other hand, in interactive IR, users' tasks tend to fall into a chronological order. Altering this so as to perform 'learning' would necessitate additional assumptions.

Learning by examples and IR

Negative examples tend to have important roles in ML for acquiring or deriving various concepts. However, there several arguments for their unsuitability for learning in IR.

(a) There are statistical arguments. The number of negative examples are too small to be representative of the negative class;

(b) The negative examples seen (i.e. retrieved) cannot be typical of the totality of negative items - the retrieved and non-relevant set is peculiar in that not all non-relevant documents are so for the same reason. However, they still share some terms with the query which is why they were retrieved in the first place. If any 'learning' is to take place from the use of negative examples, a supply of "genuine rubbish" would be better;

(c) Van Rijsbergen (1986) did test it and found it not useful, but those working on the SMART system tend to prefer the so-called dec-hi method, which uses the first retrieved non-relevant document only (Salton, 1983; Rocchio, 1971; Ide, 1971).

Additionally, the role of the two document types is also unequal. When relevant documents (or items) are used to update a query, certain terms are added to the query and the weight of other terms is increased (so as to force the query to move in a direction where other relevant documents may be found). On the otherhand, when non-relevant documents are used, certain terms are deleted from the query and other term weights are decreased. This option reduces the 'closeness' of the query with some documents but does not provide alternative directions for the query.

Learning by example techniques, such as ID3, have been used in IR (Section 2.3) for classification tasks. The learning by example approach has considerable potential for application to IR and in many ways has already been done so, particularly with statistical techniques.

The Context Learner and Learning by Examples

The *context learner*, described in this thesis, is a form of learning by examples. However, of the popular variations discussed, it is not EBL and not quite SBL either. It is not EBL particularly as there is no domain theory defined. It is not SBL as these techniques tend to encourage the use of counter examples.

The *context learner* as described later (Section 5.6) has modules/options relating to the *static* or *dynamic* notion of *context*. The first is analogous to the *single trial* method, whilst the dynamic or changing notion of *context* relates more to the *incremental* approach, described previously in this section. With a static context, all relevance feedback data upto the current query is used *all-at-once*. With a dynamic context, only the relevance feedback from the last query is used in order to update and merge with the pre-existing 'context'. However, the *context learner* operates *incrementally* for both situations in the sense that the 'context' is updated/modified in some form after each query (with positive relevance feedback). It is what happens behind the scenes for the update that represents these two differing approaches to learning by examples.

3.1.4 Learning by Analogy

The principles of learning by analogy are based on the human ability to exploit past experience or to follow the solution of a solved example problem to quicken problem solving in new but closely related situations. Thus, this approach proceeds to hypothesise from analogous cases. It has more use if the types of problems solved earlier are indicative of new problems likely to be encountered. With this approach, past problem solutions are used directly to guide the construction of new problem solutions. The knowledge given may not necessarily be directly relevant. But this method applies existing knowledge to a new problem on the basis of similarities between them. Hence, modifications are made on the existing knowledge to fit the new cases.

The *process of learning by analogy* involves both induction and deduction. Finding the common substructure between the different domains involves inductive inference whilst applying these structures to specific cases through analogical mapping is a deductive process. This approach requires more inference than *learning by rote*, *learning by being told* or *learning by example*. Learning by examples proceeds within a hypothesis space (e.g. EBL and SBL). However, *learning by analogy* breaks out of that hypothesis space.

Anderson and Kline (Anderson and Kline, 1979; Anderson, 1983) investigated skill acquistion in geometry in which learning by analogy was useful. Following is an example of the type of problems they worked on.

Eg: The *"RONY"* problem below requires some inferences from learning by analogy. Here, points *R, O, N,* and *Y* are colinear and *RO=NY*. Given these facts, the task is to prove that *RN=OY*. The proof is based on the two line segments being equal in length and that adding the same number to both will not alter this equality (although the resulting line segments would be different in length).

The analogous problem is to do with congruent angles. Similarly, given that angles *BAC* and *DAE* are congruent, the task is to prove that *BAD* and *CAE* are also congruent.

The analogical transfer that is required is based on recognising the structural similarity, converting line segments to angles, and confirming that the addition of line segments can be replace by the addition of angles.

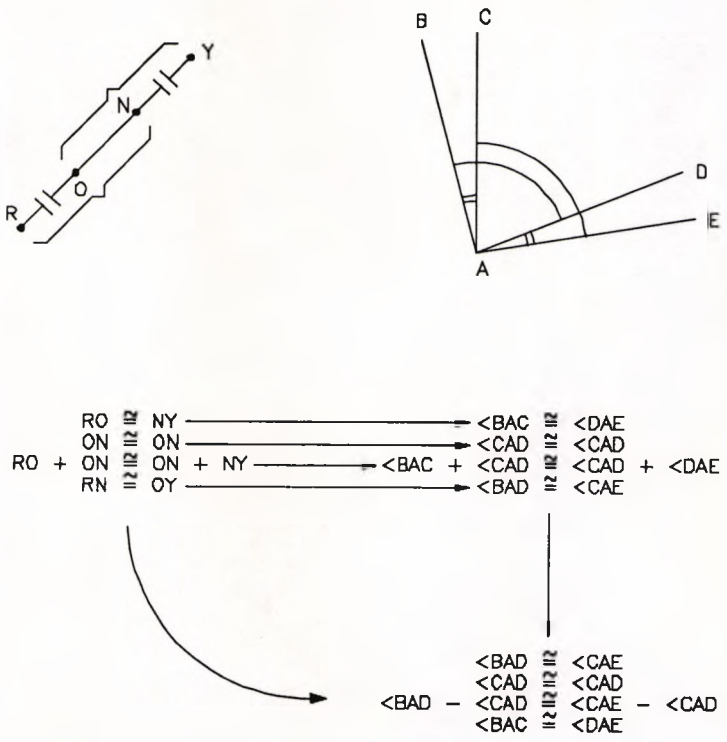


Fig.3.3: Analogical reasoning in geometry.

The *learning by analogy* approach is also important in case-based reasoning (CBR) (Kolodnor, 1987; Williams, 1988). CBR approaches have played an important role in expert systems in

medicine and law. The knowledge base for this type of reasoning is a set of relevant examples rather than general rules. These are then applied to new problems through an analogical reasoning process.

Learning by analogy and IR

In order to use learning by analogy in IR, one has to establish what new situations will be closely related to any existing ones. Establishing this automatically may be difficult and it is not very clear what type of modifications would be necessary in existing IR knowledge to make solutions apply to new IR tasks.

The problems are similar to those for rote learning where there are difficulties in saying that one IR problem/task or situation is the same as another (in learning by analogy the difficulty is in identifying 'analogous' tasks i.e. what identifies the degree to which the situations are 'analogous').

Whether the same solution method applies to a new situation would require some significant assumptions about the nature and similarity of IR tasks. One needs to be clear just how problems would be indicative of future ones and how they would be classified. i.e. How analogous can a solution for one user (IR task) be to another. With reference to the work here, extending the context learner to apply to user groups rather than individual users may involve learning by analogy.

3.1.5 Learning by Observation and Discovery

Learning by observation and discovery is based on the metaphor where humans discover regularities in their environment through observation and experimentation. In this approach examples are unclassified. Of the five methods, it is the only one based on unsupervised learning. This means that training instances/data are passed to the program in raw form or produced by the program itself through experimentation and not organised by a teacher, for example.

This form of learning encompasses discovery systems (particularly rediscovering scientific and mathematical laws), theory-formation tasks and creating classification criteria. When theories are being rediscovered, as Zytkow and Simon (1986) point out, this is not necessarily how it may have happened historically. The task of forming classifications is also performed in *learning by advice*, however, here it is done without the benefit of a teacher/tutor.

Carbonell et. al. (1983) further classify *learning by observation and discovery* according to the degree of interaction with the external environment. The extreme points in this classification being *passive observation* and *active experimentation.* In the first, the learner classifies/taxonomises various aspects and objects of the environment. In the latter, the learner is driven to explore its environment applying different observation and experimentation strategies (as the need arises).

Related to learning by observation and discovery is the idea of conceptual clustering. In conceptual clustering, sets of objects or observations have associated features. The aim is to divide this set into classes and subclasses. Cluster analysis methods, however, have been previously used by statisticians and biologists for producing taxonomies. Michalski and Stepp (1983) point out the various limitations of numerical taxonomy methods.[47] They argue that resulting clusters should be more conceptually coherent than those generated by the simpler traditional methods.

Although, there are similarities between this method and *learning by examples*, there are some differences, as highlighted by Langley and Carbonell (1984). 1) In conceptual clustering there is no tutor to place objects into classes, the learner must do this; 2) The resulting taxonomy involves disjunctive classes 3) These systems must form concepts at multiple levels of description and impose some hierarchical organisation on the concepts.

An example of this approach can be found in the RUMMAGE (Fisher and Langley, 1985) system. This system uses a model driven approach to construct its taxonomy. The algorithm has similarities with ID3 (mentioned earlier in section 3.1.3) but differs in its evaluation component. ID3's evaluation function requires instances to be grouped into positive and negative classes, whereas RUMMAGE generates descriptions of each class and evaluates these instead.

COBWEB (Fisher, 1987; Fisher and Langley, 1985) is an incremental learner which uses a probabilistic representation for concepts. It stores the probability that a given feature will occur for an instance of a concept. These probabilities direct the search through the space of conceptual hierarchies.

AM[48] (Lenat, 1983; Lenat, 1977) operates in the domain of number theory. It starts with some one-hundred initial concepts (such as set membership, set union etc.) and several hundred heuristics for deciding which concepts are 'interesting' (e.g. if one concept has only a few examples it could be 'interesting'). Given the basic objects and operations of number theory, AM rediscovered a number of familiar concepts such as integers, addition, multiplication, prime numbers and so forth. The inferences are based on the qualitative relations between the concepts.

Likewise, BACON (Langley et. al., 1986; Langley et. al., 1983) has rediscovered laws from the history of physics and chemistry (Snell's law of refraction, ideal gas law etc.) In doing so, the system also postulated a number of intrinsic properties such as mass, atomic weight, index of refraction etc. The method consists of gathering data systematically, varying one independent term at a time and examining the values of dependent variables. The system then looks for (monotonic) relations between terms and uses these to define new ones and recurses until it finds

---

[47] Firstly, numerical taxonomies generate definitions of categories based on the range of a term as measured by the number of objects to which it applies (extensional). They indicate a preference for more concise definitions based on concept comprehension (intensional) with some predictive capabilities. Secondly, numerical methods use only the objects themselves in evaluating alternative clusters as opposed to understanding and describing the nature of object clusters (e.g. preferring simpler to more complex descriptions and preferring greater predictive power to overly specific concept descriptions).

[48] Automated Mathematician.

terms with constant values. Hence, the approach is to rediscover quantitative laws, based on discovering quantitative regularities, that summarise numeric data.

Learning by observation and discovery and IR

As this is a form of unsupervised learning it is necessary for the learner to do more processing (take more 'risks') than the other approaches. This inductive approach fits more suitably to those domains (fields/subject areas) which can be more easily defined and formalised. This is not as straightforward in the case of IR, however. IR as a domain is not a 'closed' environment and its problems are not similar to discovering various physical or mathematical laws as approached by AI problem solvers. It is not possible to formally define 'laws or axioms of retrieval'. This would rely on the existence of a rule or theory of IR which is generally applicable accross all users, systems and databases which could then be 'discovered' by the learner - an unlikely situation. In fact, it is disputable whether such a 'definite' rule/theorem could be applicable to even one user on a particular system using a particular/specific database. Users' information needs, even if they could be defined by the users themselves, are not usually constant and easily identifiable. Problems of defining the rules aside, even in the case of scientific laws it is existing laws that are discovered and not new ones.

Nevertheless, this type of learning has been done in IR with statistical methods. Statistical clustering techniques in IR have been discussed in section 2.3. The difficulties referred to here relate particularly to conceptual clustering as IR does not lend well to formal representations and extensive literature on associated proofs would be required.

3.2 A computation-oriented Analysis of Learning

In the previous section, five general *learning* approaches analogous to human learning were described. As already mentioned, existing learners do not always fall exactly into only one of these categories and this is not the only dimension in which to observe the learning techniques. Other dimensions include the consideration of the following factors:

- whether they are data or model driven (bottom-up/top-down)
- whether they improve system speed, scope or quality
- whether they are based on symbolic or mathematical approaches

3.2.1 Data and Model Driven Learners

Learning strategies can also be executed in a variety of ways. They could be data-driven or model-driven, i.e. essentially bottom-up or top-down approaches - a similar situation to that when writing programs. The difference between the two approaches are particularly evident when learning from examples. Data-driven learners generalise by relying entirely on the data presented to them. They incorporate instances in the generation of new hypotheses. Model-driven ones proceed by generating fairly general hypotheses that are subsequently tested against the given examples or the user in an interactive session.

In some cases both approaches could be used together through an iterative process. For example, the system could initially be data-driven but as the training instances increase, the acquired

higher levels of generalisations (used to build a model) can then be applied to the other data sets. Examples of the data and model driven approaches follow.

Data-Driven Learner

The following example, based on (Kolokouris, 1986), expressed in Prolog-like syntax shows what kind of generalisation is possible even when given a small number of facts in the data/knowledge base and how it is arrived at.

Eg: Given the following facts, the generalisation further below can be extracted.

customer(X), profession(X,accountant), lives_in(X,london), buys(X,500).
(A customer who is an accountant and lives in London can afford something of value upto £500).

customer(X), profession(X,lawyer), lives_in(X,kent), buys(X,1000).
(A customer who is a lawyer and lives in Kent can afford something of value upto £1000).

customer(X), profession(X,accountant), lives_in(X,kent), buys(X,1500).
(A customer who is an accountant again but this time lives in Kent can afford something of value upto £1500).

The generalisation is as follows:

customer(X), lives_in(X,kent), buys(X,Z) and Z ≤ 1000.

So long as the customer lives in Kent, he/she can afford anything upto £1000 irrespective of occupation.

The learner uses a simple generalisation rule known as "dropping condition rule" and the fact that *buys(X,1500) => buys(X,1000)*. The rule states that in order to generalise a conjunction you may drop some of its conjunctive conditions. (Each of the above facts are made up through conjoining terms).

An analogous situation, of redundancy, in IR would be if every record containing "Thatcher" also contained "Mrs" then "Mrs" would be redundant.

Model-Driven Learner

The following is an example of MARVIN (Sammut and Banerji, 1986), a Model-driven Learner. The description language used is a subset of Horn clause logic. The system depends on hierarchically structured examples to be provided by the teacher.

Eg: Here we assume that the system has already been taught the existence of a concept called "letter" where examples of "letter" such as "A", "B", ... "Z" have already been typed in.

:

:

Remark: Having taught the system the existence of letters A..Z continue as follows:

:

Teach another concept? **Yes**

Concept name?	**letter-list**
Show me an example of a letter-list:	**[]**
X is [].	
Show me an example of a letter-list:	**[A]**
The head of X is "A".	
The tail of X is [].	
"A" is an instance of a letter.	
The description can be generalised to:	
The head of X is a letter.	
The tail of X is [].	
Is [B] also an instance of a letter-list?	**Yes**
[] is an instance of letter-list.	
The description can be generalised to:	
The head of X is a letter.	
The tail of X is a letter-list.	
Is [B A] an example of letter-list?	**Yes**
Show me an example of letter-list:	**No**
The description of letter-list is:	
X is a letter-list if:	
X is []	
or	
The head of X is a letter	
The tail of X is a letter-list.	

In summary, the concept of a letter-list including its recursive definition has been "learnt".

i.e. letter-list = [letter, letter-list]
or the empty list "[]".

By being given examples such as "[A]","[B]" as an example of "letter-list", the system deduces that the content of a "letter- list" is an occurrence of the concept "letter" which it had learnt previously.

In general terms, a model-driven learner would have the following algorithm (Kolokouris, 1986).

Eg:

```
begin:  develop a hypothesis

while   hypothesis does not satisfy target concept
        do
            try another hypothesis that is more general
            or more specific depending on how the trial
            concept relates to the target concept
        done
end while

end
```

In IR, finding an example of a 'concept' that matches the above algorithm may be difficult. However, teaching the system a user profile (what would constitute a naive/expert user) could be such an example, providing the purpose and assumptions for differentiating between users is clear.

3.2.2 Improving System Speed, Scope and Quality

Learning approaches in terms of their purpose and functionality can also be divided into the following crude categories depending on their effect on the performance component of the system. The categories are according to whether they improve system *speed* vs. *scope* vs. *quality*. The latter two can be more closely associated with some form of "intelligence" being built into the system.

In the first type of learning (improving system speed), the main purpose is for the learner to provide a short-cut through using various heuristics for an otherwise lengthy and cpu intensive process. Although throwing more hardware and computer processing powers can achieve the same end, the second category of techniques provide a more 'intelligent' solution to the problem.

In the second category, the aim is to improve system *quality* through more adaptable and intelligent techniques rather than solely concentrating on quantitative aspects. In IR terms, intelligence could be to 'impress' the user with a more adaptable system as is the intention with the *context learner* (however, this may not necessarily make a difference in system speed of displaying the retrieved documents, for example). The sphere of influence of the techniques (*scope*) is also important. Keeping it too wide can reduce system quality. In the field of ML, techniques tend to work better when contained in particular domains.

Learning applications, however, may fall into both categories. Sometimes making the system more intelligent can result in a direct improvement in system speed and efficiency, for example.

3.2.3 Sub-symbolic Learning

Alternative divisions of learning techniques have been according to the amount of a priori knowledge built into the system and in the way knowledge is represented and modified (Michalski et. al., 1986; Carbonell et. al., 1983). Below are these broad categories in decreasing order of prior knowledge encoded in the system.

- Knowledge-intensive domain specific models
- Symbolic concept acquisiton
- Mathematical models

The boundaries between the first two classifications are actually not very clear. The final category distinguishes itself as it is based on numerical approaches to learning (e.g. neural networks). These approaches are not necessarily detached from the five original approaches mentioned earlier (Section 3.1). For the purposes of the work in this thesis, the final category of numerical approaches have not been used. The major reason being that it is not possible to reason (symbolically) from any results achieved (positive or negative), as described more fully below.

Knowledge intensive - domain specific learning

In *knowledge intensive - domain specific* learning the system contains knowledge structures, pre-defined concepts, domain constraints and heuristics. Relevancy or non-relevancy of concepts are then proved and new concepts derived. Hence, the process is also referred to as "constructive induction". Examples include AM (Lenat, 1983), described previously, and expert systems such as Meta-Dendral (Buchanan and Feigenbaum, 1978).

Symbolic concept acquisition

Symbolic concept acquisition (SCA) is where previously given concepts are represented symbolically through analysing examples and counter-examples. Learning takes place with the form of representations such as logical expressions, trees, production rules and semantic networks. Example systems, of those described previously, include ARCH (Winston, 1975) and ID3 (Quinlan, 1986).

Often the first two approaches are combined in interchangeable modules. In this case, general-purpose learning techniques can be separated from domain-specific knowledge, so that the general-purpose learning techniques can be applied to other domains as well. Hence, the possibility of having exchangeable modules. Example systems include LEX (Mitchell et. al., 1983) and SOAR (Laird et. al., 1986).

Mathematical models

This broad category contains the approaches with the least amount of a priori knowledge built in to the learning system. It includes *neural networks (conntectionism), decision theoretic approaches* and *genetic algorithms*. These techniques cut across all the classifications made according to the human-oriented analysis of learning described at the beginning of this chapter. Following is a brief description of these approaches. However, as they are not based on symbolic learning (the acquisition of symbols) they are not the focus of this work.

Neural networks are based on models of the neurons in the human brain. The neurons are computing units which have thresholds enabling them to act as classifiers or feature detectors. A neural network consists, typically, of a large number (hundreds or thousands) of these simple computing units. Each unit possesses a single output, but may have a very large number of inputs from other units. Each unit performs a single type of computation. Learning takes place through the iterative process of adjusting and readjusting the weights/strengths of the connections between these units/neurons and hence, involves considerable numerical computing. In other words, the learning takes place with incremental changes in the probabilities that the units would transmit a 'signal'. The knowledge, in neural networks, therefore, is not explicitly encoded as such but is held within the connections between the neurons. It is stored in the network by adjusting the weights until the network gives the desired effect.

Neural networks are designed to assume complex interactions. The input of the units also depend on what happened to them before. In this sense, the network must be learning although it does not look like an explicit learning model. There are connections between probabilistic models and neural networks although their starting points are different. Neural networks' starting point is the

analogy with the brain and the computational networks are another way of casting a model. The starting point of probabilistic models, on the other hand, is probabilistic theory.

There are various network types and learning rules. Work on this began as early as the 1940s (McCulloch and Pitts, 1943). Considerable research has been done in this area including the work of (Hebb, 1949; Rosenblatt, 1958; Widrow and Hoff, 1960; Minsky and Papert, 1969[49]; Kohonen, 1984; Hopfield, 1982; Rumelhart et.al., 1986; Ackley et. al., 1985). The earlier work was mainly theoretical although some special purpose hardware was used e.g. Perceptron (Rosenblatt, 1958) and Adaline (Widrow and Hoff, 1960). Many commercial packages supporting these network types have since been developed. Currently, neural networks have been used with success particularly in pattern recognition, image processing and data trend analysis including currency movements, mortgage forecasts and even predicting ballistic paths.

Neural networks are also referred to as *self-organising systems* as they start with a random or partially random structure and 'organise' themselves with training. Their benefits include considerable robustness, in terms of immunity to noise in the input data or the failure of some nodes. They also implement parallelism.

Paradoxically, despite being numeric methods, they are not good at the kind of mathematical tasks easily performed by conventional computer programs. They can not model higher-level cognitive mechanisms and there is some question as to whether they are the right level of abstraction to describe higher-level processes and implement 'intelligence' on a machine (Luger and Stubblefield, 1989).

Additionally, in comparison to the human nervous system they need higher numerical accuracy. The nervous system relies on collective effects involving billions of neurons which are individually imprecise. The brain's precision comes from the ensemble of these neurons. Neural networks use only a fraction of what is used in the brain and therefore require much higher numerical accuracy (every neuron counts more) (Durham, 1989).

Research in neural networks also led to the *decision theoretic approach* used particularly in pattern recognition. In this approach, learning is equated with acquiring forms (linear or polynomial, for example) of discriminant functions from a given set of training examples. Also related is the work in *genetic algorithms*. *Genetic algorithms* involve the simulation of evolutionary processes. The idea being that through random mutation and "natural" selection one might create a system capable of some intelligent behaviour (Friedberg, 1958, 1959; Holland, 1980).

Genetic algorithms induce general descriptions from sets of examples through maintaining a number of competing description hypotheses. With the new incoming training instances, the system tests its hypotheses and rates them on their ability to correctly classify the new instances. At each iteration, the algorithm produces new hypotheses through a process resembling genetic

[49]Minsky and Papert's (1969) work highlighting the limitations and practical constraints of perceptrons had a significant effect on the subsequent lack of funding in the field.

reproduction (two hypotheses are combined to form a new description containing components of both). Akin to the model of evolution, successful hypotheses are subsequently allowed to carry over to next iterations (produce more offspring). Less successful hypotheses are allowed to reoccur, but with lower frequency thus enabling valuable components in descriptions to be kept. Thus, over a number of trials, successful hypotheses come to be further generated (breed) in the population (Holland et. al., 1986).

The categorisations above (Knowledge-intensive domain specific models, Symbolic concept acquisiton, Mathematical models) were generally based on the amount of built in a priori knowledge and the way in which it is represented in the system. Categorisations, however, have also been made according to the level and depth of knowledge encoded. Rada (1987) made such a distinction (knowledge-sparse vs. knowledge-rich learning) with particular reference to IR.

Knowledge-sparse learning emphasises the power of statistics and numbers attached to objects or terms. One kind of knowledge/information, in IR, can be to use frequency of words. Another possibility is to get user feedback on relevance which can give additional information on frequency of words.

Knowledge-sparse learning also includes the use of genetic algorithms and learning based on document words. The genetic algorithm would use descriptions of the documents and values based on performance of these documents. The descriptions are tested and the ones with better performance figures are then used. Learning based on document retrieval involves analysing the word frequencies and weightings. The systems using mathematical approaches such as neural networks and those using parallel processing are also examples of knowledge-sparse learning.

The **knowledge-rich learning** approach uses learning from the behaviour of querists for learning patterns in documents. The grammatical structures of queries are compared to help resolve ambiguities when learning from queries. Learning the syntax of the queries and the semantics does need some assessment and checking from the users.

3.3 Evaluation in Learning

Evaluation in ML concentrates on examining the degree to which the original learning objective is achieved. For example, if the aim was to provide a clustering mechanism then one might measure the extent to which new objects fit into these clusters or how quickly they stabilise i.e. the more quickly and accurately a new object is fitted into its classification the more 'successful' the learner is regarded to be. In the ML community (but also AI in general), the tendency is not so much to concentrate on the nature, correctness or usefulness of the knowledge that the learning programs 'know' but on whether the programs perform correctly (proving the "correctness" of programs). For example, if a natural language learning program is evaluated, the emphasis may typically be on what percentage of the words found were identified correctly - rather than what percentage of the whole collection was identified correctly (Coates, 1993). It is difficult, in IR, to have an appropriate measure of success and the obstacles in IR, the difficulties of evaluation and finding the appropriate measure of success are well known.

Additionally, ML applications usually concentrate on 'closed' domains in which the problems, rules and procedures can be fairly well defined. There is, however, a social science side to IR which makes it difficult to envisage and define all problems and information needs as well as their 'solutions'.

In ML, performance is also measured by any speed improvements in the system that may result through the application of learning. However, the main difference with IR is in the nature and size of data sets. A good size training data set, in ML research up to now, typically consists of hundreds of examples - an almost insignificant amount when dealing with IR databases. Even if a subset of the original database is used for test collection purposes the number of records is usually at least an order of 10,000 (more often 100,000). The ML community tend to bemoan the fact that their training sets are small and that it is difficult to get 'real life' data. In this sense, IR should provide a good potential application. However, there are some difficulties as well.

Data in IR can be captured from the database records/documents, user queries and relevance judgements (and possibly thesauri). In IR test collections, although the number of database records and index terms can be very large the number of user queries are much smaller. The difficulty of obtaining queries is exemplified with the TREC (Harman, 1993) experiments which now, after one-and-half year have some 150 queries. Additionally, one is typically looking at documents judged to be relevant for a query which means that the resulting number of documents contributing to the data set is considerably smaller.[50]

For IR system development and evaluation, a data set should consist of a training set and an evaluation set. The requirement, in terms of the nature of the sets, is that they both contain the same kind of data but not the same data. This may not always be easy to obtain or ensure.

In some cases, however, three sets have been used: a training set, a preliminary evaluation set to help decide which method to use (i.e. training the researcher), and a final evaluation set. We can view the work in this thesis as containing two training sets and an evaluation set. The training sets consisted of all queries and their (positive) relevance judgements to date for a group of users. The first set was used to decide which context learning method to use. The second, what to apply the method to. The evaluation set consisted of the new query (for that user). Thus, for the two evaluation processes undertaken (Evaluation 1, Evaluation 2, in Chapter 7), in each case training was performed on a training set and performance measured on a new (incoming) query.

The existence of an external (to ML) evaluation tradition and state of knowledge in IR is an advantage (most application of ML do not have that), as well as a disadvantage because the difficulties are well known.

[50]This does depend on the application. For example, if the task is learning to classify documents from those already classified the data set would be larger.

Chapter 4

Learning in Information Retrieval

The previous two chapters provided an overview of the developments in IR and ML and how the two fields may be combined. Alongside this, however, the importance of *context* and how it typically exists in an IR situation was discussed. Chapter 2 introduced the idea of *context* and discussed developments in intelligent IR. Chapter 3 provided an overview of the current ML techniques and how they may or may not apply to IR.

The purpose of this chapter is to combine some of these aspects with a view to summarising and justifying the kind of path taken in applying ML for IR. The algorithm, application and results are discussed in the next chapters. What follows in this chapter is the design considerations in developing such systems, a summary of ML applications suited for IR, the sources of learning in IR and the hypotheses for applying ML to IR in this work.

4.1 Designing Intelligent Information Retrieval Systems

The kinds of decisions to be made in designing intelligent IRSs are listed below. The emphasis is on the way in which they have been interpreted in this work. They cover machine learning application related aspects, what there is to work with in IR and evaluation. They are as follows:

- Whether a model is to be used and if so which type
- Point of view of the model. There are various ways to classify models but one prime distinction is *user* vs *intermediary*[51]
- Degree of expert knowledge (if any) to be encoded initially. For example, whether the system ought to have an initial set of rules as a starting point or not.
- Machine learning strategy or strategies to use
- Type of system to be developed. In the context of this general decision, one consideration is whether to develop a front-end or modify the system itself.
- Methods for the initial analysis of user search information and application program development.
- Method of helping users e.g. helping with/via their queries, reconstructing or extracting 'knowledge' from the existing databases

[51] *User* and *intermediary* have been used as keywords, here, in order to distinguish between *end-users* of an IRS and the intermediaries who perform searches on their behalf.

- Type and level of referral to be offered by the IRS
- Evaluation method for the machine learning strategies in the IRS

Deciding on the nature and function of the system to be developed is largely dependent on both the available hardware, software and the way these interact with online databases. As regards to using a model, a typical consideration is whether to use a user or system model. It is important to identify the approaches to be used. If a model is to be used then its point of view needs to be specified. It could, for example be based on the user or it could be based on the intermediary (as explained in footnote). i.e. Should the system aim to simulate some of the functions of the intermediary?

The amount of initial knowledge to be encoded needs also to be decided. Is the system to learn the basics of the requirements and proceed from there? Or should some domain knowledge be encoded initially with further techniques and solutions being learnt later. This decision would also affect the level and depth of knowledge learnt.

The method of helping users should be established. Whether it is directly through query formulation or perhaps a more indirect way such as organising the knowledge in the databases (forming semantic networks from the terms in thesauri) or providing/improving document ranking. So far, in this discussion, user requirements have been considered to be online requests for documents. However, they may have other requests involving referral to other sources, organisations, societies etc. Hence, it is important to distinguish the extent of referral offered by the system.

The appropriate machine learning strategy needs to be decided. It may be more appropriate to have a combination of strategies. Although each application should be considered separately, "learning from examples" lends itself well to IR (for reasons discussed earlier Section 3.1.3).

In addition, the methodology to be used (in system development) needs to be agreed upon. This should include an overall approach of analysing the information the system would use, developing the system itself and a more specific approach of deciding a programming methodology. Establishing an evaluation method for comparing the performances of the various machine learning strategies would help decide which ones should be used. Evaluation in ML and IR along with their differences have been discussed in Sections 3.3.

The above are a few example issues that will need to be resolved when applying machine learning to information retrieval. However, due to the often "fuzzy" nature of information retrieval where compromise can be more common than certainty, solutions might often not be as "clear-cut" as has been the case in some other domains where machine learning has been applied.

It is also worth remembering the quantity of information that is likely to be handled in any information retrieval system. Such large quantities have not traditionally been used in machine learning systems.

Despite the growing realisation of the importance of using machine learning and other artificial intelligence techniques in information retrieval, there is still a lack of available products reflecting this in the market. Database hosts/suppliers often claim "intelligence" in their products when what is really used is perhaps an optimisation technique of space, quicker indexing methods or files which simply contain linear lists of information about users (typically access security details). Hence, often it is the nature of the information rather than the way in which it is handled that can make a system appear "intelligent".

However, there is potential for the use of artificial intelligence techniques and in particular, as argued in this thesis, machine learning to achieve more "intelligent" tasks in information retrieval. These tasks may include the generation of rules for expert systems instead of having to encode them (a more remote possibility), forming semantic networks (or some form of links) and other knowledge structures from the databases or providing services to users without expecting proficiency on the subject or the system.

4.2 Why some Machine Learning Applications are more suitable to Information Retrieval

A variety of learning techniques and their applicability to IR were discussed in Chapter 3. This section brings together some of the more important points in the previous chapter with a view to explaining the approach taken in this work.

Rote learning has its main disadvantage in IR in the fact that it is not possible for the same IR situation to reoccur in exactly the same way with the same 'meaning' and for the same information need. Therefore, repeating a memorised set of actions for a certain situation (which will never happen) is pointless. *Learning by being told*, in IR, exists to some degree in text routing and has the potential of being applied to classification and user profile construction. The work here is related to approximating user *contexts* (see also Section 4.4) and this approach is certainly a possibility. However, without more substantial evidence on user *contexts*, how they are formed and how they may be defined, this somewhat didactic approach is inappropriate at this stage (as it is necessary to be able to tell the system what to learn explicitly).

Learning by analogy has some similar problems to rote learning in its applicability to IR. Establishing which situations are similar to each other so that they could generate the same system reponse may not be so easy. Fuhr and Robertson (1993) also state that "the behaviour of a specific term in respect of one query is unlikely to tell us much about its behaviour in relation to a different query".[52] Broadly, this learning approach, however, has been applied in statistical form (clustering) by calculating degrees of similarity. The main problem with *learning by observation and discovery*, on the other hand, is that it is more suited to domains where the rules can be more easily defined. Deriving rules, in IR, to encompass a large number of users

[52]The behaviour of a specific term with respect to one query may, however, indicate something about the behaviour of other terms sharing certain characteristics with the specific term.

or queries or an even larger number of documents is no simple matter. Any rules or heuristics derived, such as those obtained by intermediaries, are not necessarily absolute either.

The *learning by example* approach remains for most cases the more suitable approach for IR. It implies abstracting in some way from various IR situations or IR related units (such as documents, terms, queries). It has already been used for classification tasks.

Two popular variations in the *learning by example* approach were discussed in the previous chapter. Neither of these are used here. *Explanation-based learning* is not used as there is no domain theory of/in IR that can be defined (explicitly). *Similarity-based learning* is not used either as this tends to rely on negative examples (the disadvantages of which where discussed in Section 3.1.3). The *context learner* developed uses positive examples in the form of positive relevance feedback. Both a *single trial* and *incremental* approach have been investigated and tested.

The purpose of the *context learner* is to approximate a user's context (introduced in Section 2.1) as accurately as possible. In application, the *context* consists of terms with various attributes such as the number of times each has occured in relevant documents (judged by that user). Although some of the term selection criteria may be quantitatively based (e.g. frequencies, postings) the concept of a context is not. It is not the number of terms but rather their quality that is more important in forming a useful context.

The *context learner* uses first-order predicate logic for approximating the context. Like other ML approaches it focuses on symbolic descriptions (of concepts) and not the learning of coefficients for real-valued functions. *Connectionist* or *neural network* approaches have not been chosen. The main reason being that any learning performed can not be traced or understood easily nor can the reasons for any errors be identified explicitly (known neural network disadvantages, see also Section 3.2.3). In ML, the learning process tends to viewed as an explicit heuristic search through a space of concept descriptions whereas in approaches, such as connectionism, favouring statistical techniques the idea of search is less explicit (Lewis, 1990).

In looking at the application of subsymbolic connectionist approaches in IR, Belew (1991) states that some aspects of the IR problem may well make use of symbolic representations and thus describes some benefits of hybrid systems. Fuhr and Robertson (1993) point out the need for different levels of abstraction, which should depend on the actual circumstances. For text, the levels of abstraction are either the term itself or its statistical parameters (e.g. inverse document frequency). Here, essentially the 'term' level of abstraction is used, although statistical parameters have some role in deciding which terms should be considered.

4.3 Sources for Learning in Information Retrieval

Deciding the learning technique is one issue but decisions also have to made regarding which sources and what data/information can or should be used in the IRS. This section discusses the possible sources of information for the learning process. At one level the possibilities include:

-1- Features in documents or the documents themselves.
-2- Structure of subject area as represented by classification scheme or thesaurus, for example.
-3- User searches and user search behaviour.

1. In addition to the users' queries, only the first of the above is seriously used for the scope and purposes of this work. The representation of the document in the system is used as a basis. The representation would typically include the author, title, abstract and (inspec) subject headings. A combination of these are used to index the document (see Sections 5.5, 6.1) for details). It is these index terms for a document that are the major source of input to the learning algorithms here[53].

Documents themselves (or rather their full representations on-line, not the actual piece of paper or book) no doubt are a wealth of information, not only for retrieval but for natural language understanding and processing as well. However, the full-text document is not one of the input sources for the algorithms described in the next chapter.

There are various sets of documents based on whether they are retrieved or not and whether they are marked as relevant or non-relevant by a user. For the purposes of learning, the emphasis here is on the retrieved and relevant documents.

2. In the second source, a knowledge base of subject hierarchy, could be particularly useful in classification tasks. Such a knowledge base was not used in this work nor is it created. No generally applicable absolute subject hierarchy is attempted, for the reasons given in Section 2.1.4. However, it is implemented only in the 'subjective' sense. Users' contexts' can include some 'subject' information. If they should happen to contain any hierarchical information/structure this is purely incidental and is not part of the original intentions.

The use of thesauri was considered originally (section 6.5), but abondoned later as an option for technical reasons (such as the unavailability of an Inspec thesaurus at the time of this study).

4. As mentioned above, user searches have been used and users' IR sessions analysed mainly through logs. Not only their current sessions but also their historical ones have been analysed. Although this highlighted some interesting aspects and problems of IR, the focus of the work has not been in developing a formal model of user search behaviour.

Above were described some general sources of information. However, when looked at more closely some core elements can be identified. These are documents, queries, terms, and users. The way in which they are connected is exemplified in figure 4.1.

A user can have several queries and the same query can be made by more than one user (this does not mean they have the same information need). When parsed, queries essentially are

[53]From a retrieval perspective, query terms are useful so long as they retrieve documents. For this, they need to be indexed in documents.

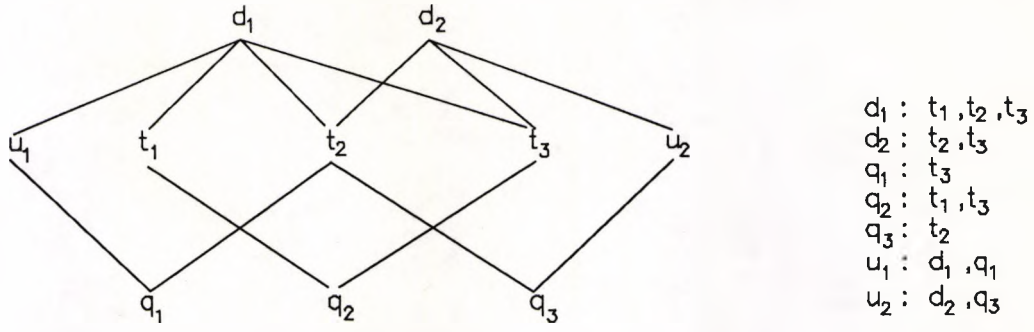


Fig.4.1: An example connection between documents, terms, queries and users.

reduced to a set of terms. In the same way, when documents are prepared for indexing they can be represented by an array of terms. Thus, most of the above elements can be reduced to *terms*. In this sense, the work here can be referred to as *term learning* (for context).

A good time for obtaining information from these various sources is during *relevance feedback*. As pointed out by Fuhr and Robertson (1993), relevance feedback is a major source of information. If there is a candidate set of features, relevance feedback can contribute to their selection. Identifying the candidate features, however, may not be easy.

Fuhr and Buckley (1993) (see Section 2.6) also use relevance information to do some statistical learning. They refer, in particular, to two IR approaches; routing queries and ad-hoc queries. For the first, the relevance feedback data is for some documents with respect to a specific query. The system's task is to rank further documents for the same query. For the ad-hoc queries, relevance information relating to query-document pairs is exploited in order to rank new documents with respect to new queries.

For routing queries, the abstraction is from specific documents based on the presence or absence of terms. For ad-hoc queries, features of specific queries and terms are used instead of the queries and terms themselves (description-oriented indexing). Firstly, a relevance description for term-document pairs is derived. Secondly, a probability relating to this relevance description is estimated.[54]

Having discussed these various sources of information and how they can be used in applying ML in IR, the following section focuses on the hypotheses and system design aspects of this work.

4.4 The Basis for the Current Investigation

Prior to discussing the details of how the above sources for learning in IR are used in this application, it is important to state the basic hypotheses of this work.

[54]More specifically, the probability that "an arbitrary term-document pair having a specific relevance description will be involved in a relevant query-document relationship" is estimated (Fuhr and Buckley, 1993).

Hypothesis 1: IR has something to learn from ML or
ML has something to offer IR (to be diplomatic).

This (*H1*) hypothesis is the starting point of this work. However, in looking at its application we come across the idea of *context*. This is a question in its own right and there is another high-level hypothesis (*H2*) that "context exists". There is some empirical evidence (Chapter 6), particularly in an academic IR environment, for this.

Hypothesis 2: Each user has a particular *context* (which may or may not change) that is predominantly undercurrent in their queries.

The two hypotheses are independent but have been investigated in parallel and merged later in application. The next question is whether *context* may be of use in solving IR problems. Section 6.6.4 describes such a case referred to as the *Weight-Block problem*. Briefly, in a ranked document list, this is when there is a large number of documents with the same score. To help overcome this problem, *context* is used to improve the ranking scheme (the way in which document ranking may be affected is discussed in Section 5.2).

However, this does not mean that the *context learner* is the only way to do ML or that it is the only solution to the *weight-block* problem. Neither is the claim that a ML approach is the only way to make use of *context*. It is simply that, through observing ML and investigating it in IR one was looking at something which people focusing on IR had not come across. An important point is that the historical searches/queries of a user may be related to his/her current search. Having identified the connection between a user's queries and how they point to a common context (for that user) one could develop statistical approaches as well.

The system developed here is one which has a learning component that can be linked into the existing interactive retrieval system. It is not a front-end. The system is not based on modelling intermediaries in particular, but data is taken from any user of the system. No prior expert knowledge or rules are encoded. However, the *contexts* learned or acquired for each user are kept. The focus is on the users' online bibliographic retrieval requirements, no other pointers are given to where users may find more information offline, for example. The learning strategy developed *(context learner)* is based on *learning by example*, as mentioned previously. Its evaluation is based on the rank position of documents. To this end, precision values for specific cut-off points (rank positions) are derived.

Chapter 5

The Context Learner

This chapter describes the *context learner* (CL) and its use. The idea of *context* was discussed more widely earlier (Sections 2.1.4 and 4.4), thus its description is more brief here. The chapter includes a description of the modules for the *context learner* and provides examples. Its use is particularly focused on improving document ranking. The purpose, of the chapter, is to show how *context learning* can be performed and the principles outlined are essentially independent of implementation. Application of the CL to an IRS is detailed in the next chapter.

5.1 Description of Context

As previously described, although each user search must be regarded as representing a different *information need*, they can all be assumed to have a *common context*. The main argument is that *context* is undercurrent in user queries. *Context* is the particular perspective that the user has when forming a query, i.e. the background, the subject area the user has in mind.

More specifically, a possible hypothesis is that there is a relationship between two queries *Q1* and *Q2* (from a particular user), in that they belong to the same context *C*. An iterative learning process can be applied to user relevance feedback in order to help estimate the *user's context* at a particular point in time. Context *C1*, for *Q1*, and *C2* for *Q2* are, in the least, related as indicated by the strong and weak hypotheses described in the text box below.

Contexts *C1* and *C2* need to be approximated. There are questions, however, of the adequacy of the representation when referring to the context as it exists after a query and the approximations to it. Nevertheless, the following is an attempt to describe the process.

In general terms, we can refer to the current query as Q_i and to the previous one as Q_{i-1}. The signs ' and '' refer to representations of the context at different times. A ' refers to the context before a particular query and '' refers to the context after that query. (Any representation of a context is an approximation to it, though the notation does not have any particular implication about the closeness of the approximation). For example, the context derived *after* Q_{i-1} is C_{i-1}''. The approximated context *before* Q_i is C_i'. Although we can assume identity between C_{i-1}'' and C_i' we might consider other information coming available in the interim (e.g. further details about the user or the subjects of interest). However, in the discussion that follows, it is assumed that no further information is available.

In summary,

- *Q1* is related to a context *C1*,
- *Q2* is related to a context *C2*,

then,

Strong hypothesis: *C1* = *C2* = *C*
Weak hypothesis: *C1* and *C2* are related (there may a gradual shift in the context)

Under the strong hypothesis, we should clearly take C2' to be C1''. Presumably, the more iterations of learning the more closer the estimated context should be to the 'true' context (which in reality can never be 'completely' defined).[55] Under the weak hypothesis, we still take C2' to be C1'', because before Q2, the only information we have about Q2's context is C1''. However, we must assume that the approximation lags behind the true context.

At present we do not know much about *user contexts* and how they are formed or behave. However, to take two extreme cases, we can presume that: Either context is *stable* or it is *changing*. If it is stable, the purpose of the user's online sessions would be to provide more evidence towards this fixed entity. If it is not stable, then we can not assume that the current true context can be derived from previous ones. It would not be possible to 'catch up' with such a context. Therefore, it is necessary to assume some continuity i.e. the current ideal/true context is not too different from the previous ones. Thus, if the context is changing slowly/gradually, then the significance of the latter sessions should increase in order for it to be up-to-date. These points are investigated in the modules described in Section 5.5.

A further hypothesis is that,

Context can be used to improve document ranking.

The *strong* hypothesis is a stronger argument for this, the *weak* hypothesis is a weaker argument.

More precisely, the argument is that context can be useful in breaking ties between documents with the same scores. This is based on extracting terms from documents as described in Section 2.1.4. The following section describes the process in further detail.

[55]The claim is not that the true context can be identified and represented accurately but rather that there can be some reasonable approximations to it.

5.2 Use of Context

Potentially, *context* can be used for a variety of IR problems. These include phrase identification and homonyms (see also Section 6.6). In particular, it can be useful in improving document ranking. Ranked documents can cluster themselves within *weight-blocks*. The concept of a *weight-block* was mentioned earlier in Section 2.1 and its existence in the Okapi IRS is described in Section 6.6.4. It consists of documents with the same score. A *document score*, in Okapi, is obtained by summing all the weights of the terms that index it, in order to rank the document.

If the size of a *weight-block*, within a retrieved document set, is large the effect of *document ranking* may diminish - particularly when a user may not be able to notice any ranking due to the large number of documents in a weight-block.

In a ranked retrieval system, context can be used for document ordering/ranking in the following ways (see also Fig.5.1):

1) Break some ties within a Weight-Block.
2) Reorder the whole retrieved set (of documents) for a query.
3) Generate a new query producing a new document list.

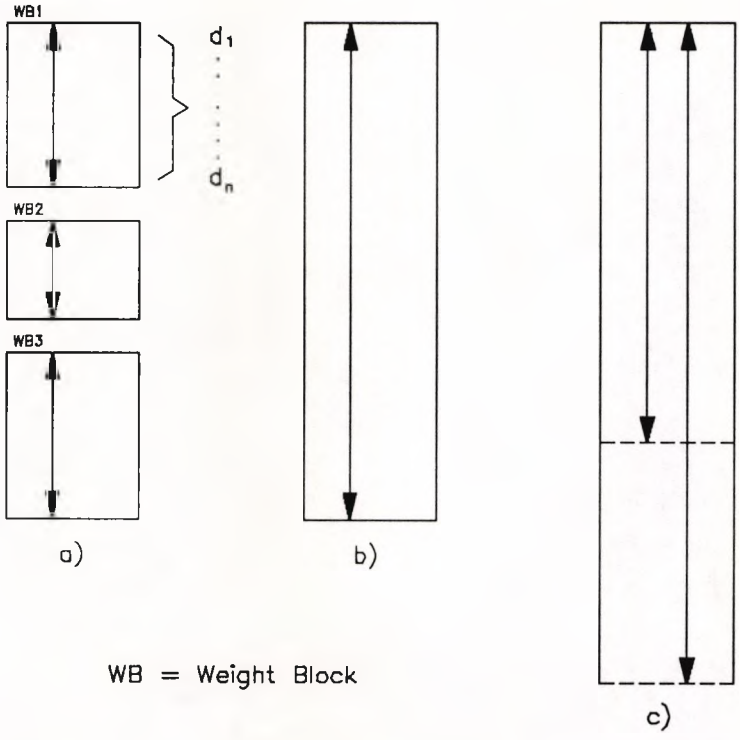


Fig.5.1: The retrived document lists showing the alternative ways of changing document ordering.

1) The first way of reordering involves breaking some ties in the weights within documents in the same weight-block. As Fig.5.1a indicates, the main principle is for documents to be

reorganised within their original weight-block. i.e. taking into account their original scores. Therefore, documents d_1 to d_n in *WB1* will be reordered but should not crossover to *WB2*. This method consists of reordering within 'chunks' in a retrieved set of documents.

2) Alternatively, the whole of the original retrieved set (of documents) for a query can be reordered. This involves modifying the retrieved set in a stronger way than in the first method.

3) Thirdly, a new query can be generated producing a new document list. This is based on modifying the original query. This is the strongest way of affecting a query and changing the originally retrieved document set.

In deciding between these three methods, the approach with the least amount of interference with the user's orginal query has been preferred initially. The results of this approach, in incorporating the *context learner* for making the system more *adaptable*, should indicate whether an alternative ought to be used.

Following is a description of how the ranking would change, (document and relevance feedback are inputs):

> *rank_function (document + relevance feedback) =*
> is a function matching document to the *context*
> in order to estimate its relative value.

More specifically, referring to the first option above, we need to "match a document to the *context* in order to estimate its relative value within the same *weight-block*." On each occasion after the initial query, the *context data* is used to break ties within weight-blocks. The documents are reordered according to the *context*. As the documents within a weight-block are essentially in random order[56], on average one should not get worse results than the random initial distribution.

The above procedures, in practical terms, would not be changed greatly to adapt them to the second method of reordering the retrieved document set - should this be necessary. In both methods 1) and 2), the main point is reordering an existing set or subset of documents but not altering the elements in the set. The third method involves creating a new set altogether. The simplest way of doing this would be to add or remove terms from the original query and resubmit the result as a completely new query. The weights of any newly added terms can be calculated in the same way as the original query terms in which case the new terms are treated equally with the original terms. Alternatively, the weighting formula could retain a bias towards old query terms.[57] This could be integrated automatically without the user's knowledge or

---

[56]It may be that within in a *weight-block*, documents will be in descending date order and author alphabetic within that (Section 6.1). However, in probabilistic ranking terms this can be considered to be random.

[57]Infact, in the absence of document rank position information, an extreme bias towards original query terms can ensure within weight-block ordering. For example, giving added terms (such as those indicated by the *context*) very low weights. This form of weighting is used in this work. See also Section 6.6.4, 7.3

manually by presenting it to the user as an alternative (although it could be argued that this may frustrate and confuse the user).

5.3 Inputs to the Learner

Each user online session consists of one or more queries. Not every query results in relevance feedback from the user. Learning only takes place when there is such feedback. Relevance feedback results in the highest weighted search terms being collected from the relevant document records. Hence each *learning session* (except the first) begins with input to the learning component, made up from:

- A set of document records D which have been picked as relevant to a previous query by the user.
- A set of term structures M containing those terms (from D) with the highest weight, using the probabilistic model described in section 2.6 and 6.1.3.

A document record (d) contains a unique identifier ($d.id$) together with a set of stemmed terms (index terms) which make up the record ($d.st$). Each stemmed search term is stored in a structure (m) containing the structure's identifier ($m.t$, the stemmed search term), the number of documents in the database containing the term ($m.c$), the weight of the term ($m.wt$), the number of relevant documents that contain the term ($m.dc$). Thus, a term $m.t$ covers a document d if, at least, (bearing in mind that $d.st$ has several elements/terms)

$$m.t \in d.st$$

Within this framework it is assumed that a document marked as relevant in one session, while not necessarily still regarded as relevant in a later session, nevertheless remains representative of the *context* for the later search. Using relevance feedback from each of the searches (where it exists), the system will then create and evolve a *context* C, that can be used in future searches to influence the ordering of documents displayed.

5.4 Description Language

This section describes the *instance* and *context description languages*. These are notations which enable us to represent the document and relevance judgements (and terms) in a form which allows them to be formally manipulated.

5.4.1 The Instance Language

After relevance feedback, term information is processed into a suitable form for the context learner, i.e. the *instance language*. The purpose of this language is to represent a *case* or *instance* in order to generalise and learn from it. The language must be expressive enough to

represent all the relevant features[58] of the example session. It is used to generate terms in a *Context Description Language*, within which the concept evolves incrementally.

An element of the instance language can be defined to be a 'term structure' of the form *m(t, c, dc, wt, o)* where each named component is used as a selector function:

- *m.t* is the name of the stemmed term.
- *m.c* is the number of documents in the database in which the term appears.
- *m.dc* is the number of relevant documents in which the term appears.
- *m.wt* denotes the weight of the stemmed term
- *m.o* identifies where the term originates and in which learning sessions it has been used.[59] For example, *q(1)* means that the term was used in query 1 and *qe(2)* means that it was a potential merge (for query expansion) term in query 2.

An example term structure generated from relevance feedback and contributing to the context formed in section 5.5 is *m(supercomputer, 627, 3, 72, qe(3))*. The posting for the term was 627, the number of relevant document it appeared in for query 3 was 3 and its weight was 72.

5.4.2 The Context Description Language

An element of the Context Description Language is defined as a *context term structure* of the form *c (t, s, n, a)*.

Again, each named component is used as a selector function:

- *c.t* is the name of the stemmed term.
- *c.s* is an integer derived from the original term weight, denoting the *strength* of the stemmed term.[60]
- *c.n* is the number of relevant documents in which the term appears.
- *c.a* is a list of all the sources from which the term originates.

An example how a context is formed is in the following section and Section 5.6 details the various modules and the variety of ways in which *context learning* can be done.

5.5 Examples of Context Learning

Knowledge of a user's needs from their earlier sessions should help improve the document ordering (within the *weight-blocks*) and is the kind of improvement an adapting IRS should be

---

[58]Relevant features can, for example, be designated by a knowledge engineer. However, in this application they refer to features that can be extracted from user queries and relevant documents.

[59]This type of information would not typically be available in an online log session for a query. It can be produced through a *learner* or a filter program (prior to *learning*) which analyses the sessions in order to identify the appropriate values.

[60]This function could be used in the future but it has not been significant in the modules described in Section 5.6.

able to make. To illustrate the workings of the *learner*, here are two examples, each including several queries and the subsequent *context* 'learnt'. This includes terms to be *Used* and those on *Hold*.

Eg 1:

The *contexts* derived after each iteration in this example are based on a simple algorithm, as per test T2, the modules for which are described in Section 5.6. The table shows the *context* terms consisting of *Used* terms and *Hold* terms. The *Used* term weights/strength[61], the number of documents they cover (frequency) and where the term previously occured (query or query expansion) are given. Due to the size of the set of terms on *Hold*, only their numbers are shown in the table. Also given is a table showing the relevant documents and their index terms for query *q(1)*. The documents and terms for the remaining two queries can be found in Appendix Q (Example of Relevant Documents for Queries).

The queries were as follows:

q(1) = *''graph grammars programming''*

q(2) = *''graphical programming concurrent''*

q(3) = *''software tools parallel''*

[61] In otherwords, the weights for iterations where a term has been a potential context term.

Query	Used Terms				Hold Terms (No.)
	Term	Weight	Freq.	Occur.	
q(1)	directed	86	4	qe(1)	18
	edit	69	2	qe(1)	
	graf	86	5	q(1),qe(1)	
	grammar	105	5	q(1),qe(1)	
	rewrit	78	2	qe(1)	
	syntax	72	2	qe(1)	
q(2)	directed	86	4	qe(1)	36
	edit	69	2	qe(1)	
	graf	86	5	q(1),qe(1)	
	grammar	105	5	q(1),qe(1)	
	rewrit	78	2	qe(1)	
	syntax	72	2	qe(1)	
	concurrent	107	9	q(2),qe(2)	
	grafic	84	9	q(2),qe(2)	
	interprocess	83	2	qe(2)	
	notation	72	3	qe(2)	
	program	63	9	q(2),qe(2)	
	visualisation	61	2	qe(2)	
q(3)	concurrent	107	9	q(2),qe(2)	57
	directed	86	4	q(1)	
	edit	69	2	q(1)	
	graf	86	5	q(1),qe(1)	
	grafic	84	9	q(2),qe(2)	
	grammar	105	5	q(1),qe(1)	
	interprocess	83	2	qe(2)	
	notation	72	3	qe(2)	
	program	63	9	q(2),qe(2)	
	rewrit	78	2	qe(1)	
	syntax	72	2	qe(1)	
	visualisation	61	2	qe(2)	
	parallel	73	6	q(3),qe(3)	
	supercomputer	72	3	q(3)	
	tool	71	6	q(3),qe(3)	

Table 5.1: Table showing the query and context (Used and Hold) terms for e.g.1.

Query	Doc-id	Index terms
q(1)	3601331	support, grafic, languag, structur, specification, design, model, representation, basic, pictur, element, line, segment, rectangl, geg, object, oriented, generator, technologi, chang, dependent, group, need, computer, directed, graf, grammar, high, level, program, softwar, tool, edit
	3512638	program, interconnection, structur, aggregat, rewrit, graf, grammar, parallel, 0105, logical, abstraction, specification, script, derivation, sequenc, support, element, level
	3557048	graf, grammar, paradigm, implement, visual, languag, specification, rewrit, data, structur, diagram, syntax, directed, attribut, arrow, box, color, representation, formal, description, programmed, attributed, computer, grafic, edit, user, interfac
	3513448	graf, sup, ed, interactiv, grammar, undirected, multipl, edg, arbitrari, label, computer, grafic, design, diagram, representation, visual, support, directed
	3624845	1989, ieee, workshop, visual, languag, cat, 0012, rome, 0169, 4-6, oct, reason, 0163, larg, 0091, intelligent, grafic, interfac, directed, graf, program, iconic, queri, pictorial, data, gloto, syntax, qbd, unix, live, micro, tool, 0105, form, manipulation, type, browser, symbolic, expression, spatial, algebra, xviqu, expert_system, signor, computer, 0092, grammar, user, model, formal, specification

Table 5.2: The relevant documents and their (stemmed) index terms for query q(1) in e.g.1.

The following example consisting of two *learning* iterations, shows the difference between several *context learning* algorithms (tests T2, T12, T13, T15 as per Section 5.6.4). The first test (T2), like the previous example, consists of a simpler algorithm in which the main criteria for a term to be in the *used context* is its document count and whether it is above a certain threshold. The second test (T12), investigates the idea of a *minimal coverage* (the minimum number of terms necessary to represent a group of relevant documents) detailed in Section 5.6.2. The only difference between this test and the next one (T13) is the role previously acquired terms in the *context* (particularly those with a more passive role) have in forming the current *context*. The fourth test (T15), investigates an algorithm which ensures that when a maximum number of context terms is reached, a certain proportion are replaced with those from the new learning iteration.[62]

The two learning iterations consist of the queries below and the relevant documents in Table 5.3. The second query appears to be part of an author name. The *used context* terms (see Section 5.6 for further explanation) after each iteration for the four tests are given in Tables 5.4 - 5.7.

Eg 2: This example shows *contexts* based on slightly more complex (T12, T13, T15) algorithms, in addition to T2. The queries are as follows:

q(1) = "active vision",
q(2) = "sandini",

[62]More specifically, as detailed in Section 5.6.2, when a maximum number of used *context* terms is reached, a certain proportion of terms from the new learning iteration are placed in the (used) *context*. Terms from the new iteration are either already in the this *context* or older terms are removed so that they can be included.

Query	Doc-id	Index terms
q(1)	3941163	egomotion, perception, visual, track, activ, vision, biological, organism, locomotion, motion, stimuli, decod, parameter, binocular, robot, eye, human, ocular, mechanist, navigation, computer, mobil
	4106586	computationalli, inexpensiv, egomotion, determination, mobil, robot, activ, camera, motion, control, navigation, computer, vision, animat, gaze, fixation, point, track, parameter, position
	4097441	purpos, qualitativ, activ, vision, collision, avoidenc, prei, catch, visual, perception, recoveri, recognition, medusa, machin, navigational, pattern, computer, computerised, navigation
q(2)	4076995	sandini, stereo, vision, real, time, obstacl, avoidenc, unknown, @0105_>environment, computer, computerised, navigation, pattern, recognition, mobil, robot, brook, subsumption, architectur, ground, floor, dispariti, grei, level, imag
	3809688	sandini, activ, vision, based, space, varient, sens, computer, overlap, shape, nonoverlap, ccd, visual, sensor, resolution, field, width, data, reduction, coordinated, visuomotor, strategi, retina, preprocess, unit, 2d, track, move, target, level, recognition, imag
	3507773	sandini, uncertainti, analysis, visual, motion, depth, estimation, activ, egomotion, observer, strategi, point, track, imag, pair, match, flow, field, varienc, center, fixation, optic, velociti, zero, cross, contor, partial, independent, parameter, error, propagation, successiv, precision, accuraci, key, space, real, computer, vision
	3648649	sandini, cooperation, stereo, motion, 3d, data, acquisition, representation, esprit, extraction, reliabl, rang, sensor, modaliti, independent, estimat, imag, uncertainti, algorithm, coars, fine, control, strategi, dispariti, depth, activ, camera, description, direction, gaze, egomotion, partialli, occluded, object, equation, real, scene, move, point, space, vision, @0105_>environment, computerised, pictur

Table 5.3: The relevant documents and their (stemmed) index terms for queries q(1), q(2) in e.g.2.

In the following tables, the constituing parts (weights and frequencies) of terms that reoccur in new relevant documents and are picked for the *context* can also be found. For example, the term "activ" (Tables 5.6, 5.7) was in three of the documents for q(1) and three others for the documents of q(2) - a total of six of all the relevant documents - hence, the breakdown of '3+3' for its frequency. Likewise, its weight for the two queries are shown.

As detailed in Section 5.6, not all terms from all relevant documents are entered into the *context*. Firstly, a potential list of *query expansion / context terms* is generated. For this, all index terms from the relevant documents are ranked and the top *n* chosen. Terms such as "navigation" (Tables 5.5, 5.6) were not in this top *n* for the second query q(2), it only occured in one of the four relevant documents in that iteration. So although it was in three of the relevant documents for q(1) and one of those in q(2), its 'frequency' value in the table is not '3+1'. The reason why "navigation" is still in the (used) *context* is because of its performance in the first iteration (which is considered recent) and not because of its performance in the second.

Query	Used Terms			
	Term	Weight	Freq.	Occur.
q(1)	activ	91	3	q(1),qe(1)
	computer	74	3	qe(1)
	egomotion	137	2	qe(1)
	mobil	72	2	qe(1)
	motion	62	2	qe(1)
	navigation	98	3	qe(1)
	parameter	44	2	qe(1)
	perception	81	2	qe(1)
	robot	53	2	qe(1)
	track	65	2	qe(1)
	vision	80	3	q(1),qe(1)
	visual	66	2	qe(1)
q(2)	mobil	72	2	qe(1)
	navigation	98	3	qe(1)
	parameter	44	2	qe(1)
	perception	81	2	qe(1)
	robot	53	2	qe(1)
	activ	91+76	3+3	q(1),qe(1),qe(2)
	computer	74+59	3+3	qe(1),qe(2)
	depth	72	2	qe(2)
	dispariti	100	2	qe(2)
	egomotion	137+130	2+2	qe(1),qe(2)
	imag	68	4	qe(2)
	independent	52	2	qe(2)
	motion	62+54	2+2	qe(1),qe(2)
	move	56	2	qe(2)
	real	51	3	qe(2)
	sandini	167	4	q(2),qe(2)
	sensor	54	2	qe(2)
	space	53	3	qe(2)
	stereo	82	2	qe(2)
	strategi	56	3	qe(2)
	track	65+58	2+2	qe(1),qe(2)
	uncertainti	64	2	qe(2)
	vision	80+84	3+4	q(1),qe(1),qe(2)
	visual	66+58	2+2	qe(1),qe(2)

Table 5.4: Table showing the used context terms for e.g.2, when using the context learner of test T2.

Query	Used Terms			
	Term	Weight	Freq.	Occur.
q(1)	navigation	98	3	qe(1)
q(2)	navigation	98	3	qe(1)
	sandini	167	4	q(2),qe(2)

Table 5.5: Table showing the used context terms for e.g.2, when using the context learner of test T12.

Table 5.5 shows that for a *minimal coverage* (test T12) of the relevant documents the term "navigation" is sufficient after the first query. As can be seen in Table 5.3, it occurs in all three relevant documents for q(1). The term "sandini" represents the four new documents in q(2). There are other terms such as "vision" which also represent these documents. However, the reasons for the choices made and the particular algorithm employed can be found in Section 5.6.2.

Query	Used Terms			
	Term	Weight	Freq.	Occur.
q(1)	navigation	98	3	qe(1)
q(2)	navigation	98	3	qe(1)
	sandini	167	4	q(2),qe(2)
	activ	76+91	3+3	q(1),qe(1),qe(2)
	computer	59+74	3+3	qe(1),qe(2)
	egomotion	130+137	2+2	qe(1),qe(2)
	fixation	104+108	1+1	qe(1),qe(2)
	gaze	108+114	1+1	qe(1),qe(2)
	motion	54+62	2+2	qe(1),qe(2)
	track	58+65	2+2	qe(1),qe(2)
	vision	84+80	4+3	q(1),qe(1),qe(2)
	visual	58+66	2+2	qe(1),qe(2)

Table 5.6: Table showing the used context terms for e.g.2, when using the context learner of test T13.

Query	Used Terms			
	Term	Weight	Freq.	Occur.
q(1)	motion	62	2	qe(1)
	track	65	2	qe(1)
	visual	66	2	qe(1)
	mobil	72	2	qe(1)
	perception	81	2	qe(1)
	egomotion	137	2	qe(1)
	computer	74	3	qe(1)
	vision	80	3	q(1),qe(1)
	activ	91	3	q(1),qe(1)
	navigation	98	3	qe(1)
q(2)	dispariti	100	2	qe(2)
	egomotion	137+130	2+2	qe(1),qe(2)
	real	51	3	qe(2)
	space	53	3	qe(2)
	strategi	56	3	qe(2)
	computer	74+59	3+3	qe(1),qe(2)
	activ	91+76	3+3	q(1),qe(1),qe(2)
	imag	68	4	qe(2)
	vision	80+84	3+4	q(1),qe(1),qe(2)
	sandini	167	4	q(2),qe(2)
	perception	81	2	qe(1)
	navigation	98	3	qe(1)
	mobil	72	2	qe(1)
	motion	62+54	2+2	qe(1),qe(2)
	track	65+58	2+2	qe(1),qe(2)
	visual	66+58	2+2	qe(1),qe(2)

Table 5.7: Table showing the used context terms for e.g.2, when using the context learner of test T15.

The next section describes the modules and tests, including those mentioned in the examples above, for the *context learner.*

5.6 Modules and Tests

This section outlines the modules that form the *Context Learner*. The sources of data for the *learner* were discussed earlier (Section 5.3). The modules form a template for the overall algorithm. However, within each module there are possible variations. Some of these have been eliminated or excluded for theoretical reasons whilst others have been tested. Thus, also included in this section are the various tests (consisting of a combination of the subcomponents of the modules) that are necessary to identify the effect of components. No doubt some of the tests

would vary depending on the applied IRS.[63] However, the type of tests discussed in this section would remain predominantly the same.

The main principles of estimating or evolving a context involve the following three components:

- *R* - the set of Relevant documents
- *M* - the set of potential Merge terms (term structures)
- *L* - the set of *old context* terms (after a learning iteration, these also affect the subsequent *new context*).

The first of these is the set of relevant documents *R* as identified by the user. This is the main raw material for *learning*. Potentially, all terms extracted from *R* can be used to form the *context*. However, this is likely to create a lot of 'noise' and is not likely to be useful in discriminating between documents. From the set *R* and the query terms[64] a set of terms *M* is derived which provide us with the potential elements for the *context*. These terms somehow need to be filtered in order to determine which ones would best approximate the *context*.

The process of 'filtering' involves dividing the set *M* into *Active* and *Passive*. The *Active* set *A* includes the terms which meet certain criteria (discussed later in this section) or are above a previously determined threshold. Terms not satisfying these criteria are not discarded completely, instead they are temporarily put into a *Passive* set *P* (on reserve). For now, this has been the preferred approach as without further evidence it is not possible to know the likelihood of previously eliminated terms being important again in later approximations to *context*. Terms in this *passive* set may (in some versions of this algorithm) later be activated depending on subsequent queries and relevance judgements.

After the first iteration of learning, an approximation to *context C* is formed. On the next iteration *(n)* this becomes the *old context* and its contents are referred to as the set *L* (Last) or more specifically C_{n-1}. A *context*, in the wider sense, consists of those terms (term structures) actually being *used* to improve an IRS *(U)* and those put on *hold (H)*. In practice, it is the *U* set that affects the performance of an IRS at any given time. Although the *H* set may influence subsequent contexts, it has no role in the user's current query. Again, criteria are needed to determine how these old terms *(L)* should be merged with the new ones *(M)*, as addressed in the following subsections.

Thus, A and P relate to information about the present query, U and H relate to previous information and what was derived from the present query. The stages of using the three components *(R, M, L)* are shown in fig. 5.3.

The modules for the *context learner (CL)* are referred to as *A, B, C and D*. Each are outlined below and detailed in the following subsections.

---

[63]Depending on the IRS, the results of certain tests may eliminate the necessity of certain other ones.

[64]A relevant document will have at least one of the query terms in it, otherwise it would not have been retrieved. However, although it is quite likely that the pool of terms from the set R will include all the query terms, this is not guaranteed to be the case.

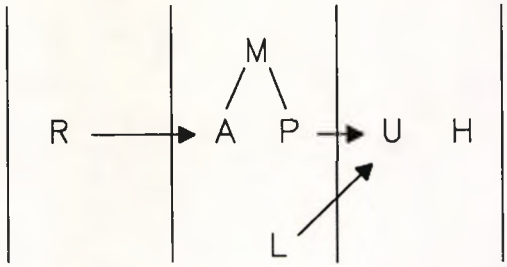


Fig.5.2: The stages of using the relevant documents including their index terms (R), potential merge terms (M) and old/last context terms (L) for approximating the context.

- **Module *A***: Create set *R* of relevant documents.
- **Module *B***: Create *Active* and *Passive* sets of terms.
- **Module *C***: Merge these newly acquired terms with the *old context.*
- **Module *D***: Use the past performance of a term to determine its role in the current *context.*

Modules *C* and *D* are very closely related in that the past performance of a term (e.g. its reoccurence in relevant documents) will determine how it is to be merged with the latest acquired terms. However, there are also other merging criteria unrelated to terms' past performances (e.g. limits on the number of terms). For this reason, the historical criterion, represented in module D, has been distinguished from the others.

From these modules we form tests (*T1, T2,* etc.) to identify which combination of the module variations gives better results. The tests include those for identifying whether *context* is *stable* or *changing* (Section 5.1). Not all permutations of the modules are tested. Sometimes there are theoretical reasons for this and sometimes it is because the results of one test can eliminate the necessity for others.

5.6.1 The Set of Relevant Documents

The purpose of this module *A* is to

- Create/obtain a set *R* of relevant documents.

In order to achieve this the following decisions have to be made regarding how *R* should be formed during each *learning* iteration.

Decision 1: What should constitute *R* - the relevant documents from the last query only or all queries.

Decision 2: How duplicate document records should be treated (whether they are removed or not).

Therefore, the resulting modules are:

A^1: Include only the relevant documents from the last query (R_{latest}) and do not remove any duplicates.

A^{II}: Include only the relevant documents from the last query (R_{latest}) and remove any duplicates.[65]

A^{III}: Include all relevant documents, starting from the first query upto and including the last (R_{total}). Keep any duplicates.

A^{IV}: Include all relevant documents, starting from the first query upto and including the last (R_{total}) but remove any duplicates.

Two models based on different premises of the nature of *context* were described; *context* as a stable entity and that in which it is dynamic/changing (Sections 2.1.4, 4.4, 5.1). The two ways of forming the set R (R_{total}, R_{latest}) reflect these differing premises.

R_{total} corresponds to the idea of true *context* being stable. Viewed from this point, the purpose of the iterations is simply to provide more 'evidence' for this *context*. Thus, all the relevance feedback data (relevant documents) are merged after every iteration to form the basis of the learning algorithms. At any point n (after n iterations - where there is feedback) the set of relevant documents in the latest search (R_{latest}) is merged with all the previously acquired relevant documents.

The second option is to use R_{latest} only. In this case, it is the renewal of terms that is important and R_{latest} reflects the idea that *context* does change. Thus, under the strong hypothesis (Section 5.1) using R_{total} would strengthen the argument. Whilst R_{latest} is more appropriate under the weak hypothesis.

Each of these sets, in turn, can be made to exclude or include duplicate references to documents. Figure 5.4 shows a tree diagram of the possibilities described, for set R.

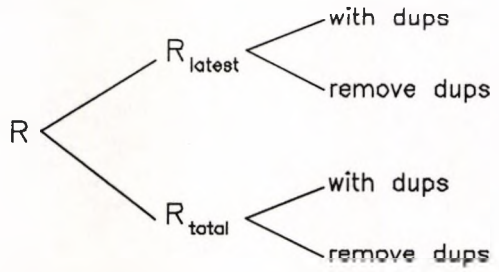


Fig.5.3: The possible ways of forming the set of relevant documents (R).

If a document is judged to be relevant in more than one query then one option is to keep only *one* (the latest) reference to it. Alternatively, *all* references to a document can be kept. The

[65]Keep the new reference to the document and remove the old so that if there are any changes in the document index terms the most up to date is kept.

likelihood of duplicates within the same query, however, is very small. Hence, we are unlikely to find them in R_{latest}.[66] On the other hand, it is quite likely to happen in the case of R_{total}.

The argument for including duplicates can not be that it reflects "more relevance" of the document concerned. A document can be chosen more than once for a variety of reasons (the simplest being that the user forgot it was already chosen). Indirectly, it may be that the terms within the document are more likely to be in the *context* but that is more to do with the abstraction process. Rather the argument should be that each query is to do with a different information need and that, therefore, any apparent "repetition" should not be interfered with.

If each query represents a separate information need then we can not automatically assume that relevance for one query is relevance for another, irrespective of the hypotheses (strong or weak) described in the previous section. However, relevance judgements provide information about context which is transferable outside a query (Section 5.1).

5.6.2 Filtering the Terms

The purpose of module *B* is to

- Create *Active* and *Passive* term sets (*A*, *P*).

Dividing a set of terms into *A* and *P* is a binary representation of the terms before entering the *context*. Likewise, once in the *context*, a term is either in the *Used (U)* set or on *Hold (H)*. Whether *A* and *P* terms are put in *U* or *H* is addressed in the latter modules.

Another way to represent terms is to weight them as is done in a probabilistic model. However, it is not obvious how historical information about a term can be incorporated into this model. A probabilistic model can weight terms for a particular query and relevance judgements. 'Merging' these with previous queries would involve a weighting mechanism for terms which reoccur (either in queries or relevant documents). There is the problem of carrying weights across queries.

A probabilistic model, however, could be incorporated in the *context learner*. A cut-off point can be applied to the weighted terms to form the *A* and *P* sets. Those above it can be put into *A* and those below into *P*. The cut-off point can be a particular weight or the top *n* terms from a weighted list. Subsequently, they can be merged with previous occurences based on the criteria described in modules *C* and *D*.

In representing terms, it is also possible to have a three way split. For example, there could be an *active* set, a *probable* set and a *definetely passive* set. The definetely passive terms can be removed from the *context* and the probable ones can be put on *hold*. In order to establish whether a term should be discarded, evidence from previous *contexts* will need to be introduced at this stage. A three way split could also be done when merging with the *old context* terms. For example, terms which have been on hold for a long time can be removed. Introducing more sets

[66] Additionally, in most IRSs, it would not be usual to be able to select a document to be relevant more than once. In application terms, this is also not possible with the Okapi IRS used in this research.

for the terms in this way makes the scheme closer to a weighting one. A version of model *B* does sort some terms and applies a maximum number for those that could be entered into the *active* set. Module *C* also has a similar version for applying a limit to the number of terms to be in the *used* set. That is the extent to which a sorted list of terms is investigated in this work. As mentioned earlier, without further evidence for the usefulness of the idea of context it was considered inappropriate at this stage to investigate further splits in the sets.

Hence, the purpose of this module is confined to creating an *active (A)* and *passive* set *(P)* of terms. For this, the set *R* and the query terms are used to derive a potential list of terms *M* useful for the *context*. *M* is then divided into *A* and *P*, where those meeting certain criteria are placed in A and those which do not in *P*. The reasons for this are discussed later. However, in forming the *A, P* sets the following decisions first have to be made:

Decision 3: This relates to two points. One is about using statistical information regarding terms and their relevance (3a), the other (3b) about document coverage. In 3a, decisions are made on a term by term basis. In 3b, decision relate to the set of terms (as a whole) covering the set of relevant documents.

3a: How to treat the number of relevant documents a term has been found in. For example, if a term is in two relevant documents then the document count *m.dc* = 2. One option, crude as it may be, is to have a predetermined threshold for deciding whether a term should be *active* or not. For the moment, this value will be referred to as *threshold1*.

If the threshold value is too high it is possible some documents would go unrepresented. If it is too low it may not be selective enough.

3b: How to represent (cover) the complete relevant document set. A possibility is to *minimally cover* the set R of relevant documents (discussed later in this sub-section).

Decision 4: Whether to consider the weight of a term (according to a probabilistic model) for that query, in forming a *context*.

Decision 5: Whether to have limits on the total number of elements in the *Active* set (*A*). For example, the total number in *A* could be limited to a certain proportion of those in *U*. If we refer to the maximum number of elements in *U* as *threshold2*, then the total number allowed for *A*, at any iteration, could be *threshold2 / 2* or *threshold2 / 3*, for example. It is this *proportion* (i.e. ½, ⅓) that needs to be decided here. The value of *threshold2* is addressed in module *C*.

Before describing the resulting modules, the idea of *minimal coverage* referred to above is explained.

The assumption behind *minimal coverage* is that users' relevant documents when covered minimally will have the least amount of 'noise' or redundancies in representing the *context*. This does not mean that it is the best and most accurate way to represent *context*. Indeed, there is a question as to whether it will be sufficient to represent it.

The aim is to derive a *complete minimally sized cover* of relevant documents. This is done by using two bias criteria: terms of *high document counts* and terms with *high weight*. All documents indicated relevant by the user have to be represented.

Thus, the main purpose is to find the combination which contains the smallest number of terms to represent all the relevant documents. A strategy is to find the terms which cover the most amount of documents and in a tie situation choose the one with the highest weight. In a non-probabilistic IRS, at this point one would probably have to make a choice from an alphabetic ordering. However, as term weights are available in a probabilistic IRS, it seems reasonable to use them instead.

Minimal coverage is a theoretical aim but may be difficult to ensure technically. There are algorithmic reasons for this as well as efficiency ones. Figure 5.5 shows some of the problems which can be encountered in implementing such a theory. It shows various documents with the terms that cover them. Below is a description of four situations which highlight some of the difficulties:

1) In figures 5.5 a) and b) terms t_1, t_2, t_3 *and* t_4 cover relevant documents d_1, d_2, d_3, d_4. A choice has to be made between choosing terms t_1, t_2 to represent the relevant documents or t_3, t_4. As both options cover two documents, the term weights have to be taken into consideration. So the term with the highest weight should be chosen and the remaining documents covered by the appropriate term. If the weights are the same then an alphabetic order is as good as any random order.

2) In figures 5.5 c) and d) terms t_5, t_6, t_7, t_8, t_9 cover the five documents d_5, d_6, d_7, d_8, d_9. If $weight(t_5) < weight(t_7)$, then the term combination represented in diagram d) is chosen which involves a larger number of terms than in option c). Also, in option d), each remaining term (t_8, t_9) has a lower document coverage.

3) Figures 5.5 e), f), g) and h) all show combinations of two terms covering the documents d_{10}, d_{11}, d_{12}, d_{13}. The question is whether it is better to have

- 2 terms, each covering 3 documents or
- 1 term covering 4 documents, the other covering 3, if this is preferred then the weight of terms have to be considered in deciding between f) and g) or
- 1 term covering 4 documents, the other covering the remaining 2.

4) Similarly, in the case of representing documents d_{16}, d_{17}, d_{18}, d_{19}, d_{20}, d_{21} there is the question of which coverage would be better:

- 1 term covering 3 documents, a second term covering 2 documents and a third term covering one document or

- 1 term covering 4 documents and two other terms covering 1 document each. With the strategy outlined below, this option would be chosen.

Following is a description of an algorithm/strategy for implementing *minimal coverage*.

Strategy:

1. Sort the available terms (from the relevant documents) in their document count order and weight within that.

2. Choose the first term in the sorted list.

3. Identify documents represented by this term.

4. If any documents remain not covered then re-sort the remaining terms and go to step 2.

As mentioned earlier, the idea of *minimal coverage* has its technical problems. For example, with above strategy will end up with j) instead of i). A heuristic solution may not satisfy all criteria. Ensuring this, for a *genuine minimal coverage*, requires all possibilities to be investigated and then a comparison to be drawn between them to choose the best one.

The requirements for *minimal coverage* include covering all relevant documents. Not all terms from the documents are chosen. The number of documents a term covers is important. However, in a genuine minimal coverage, we would also need to consider which documents are covered with a term.

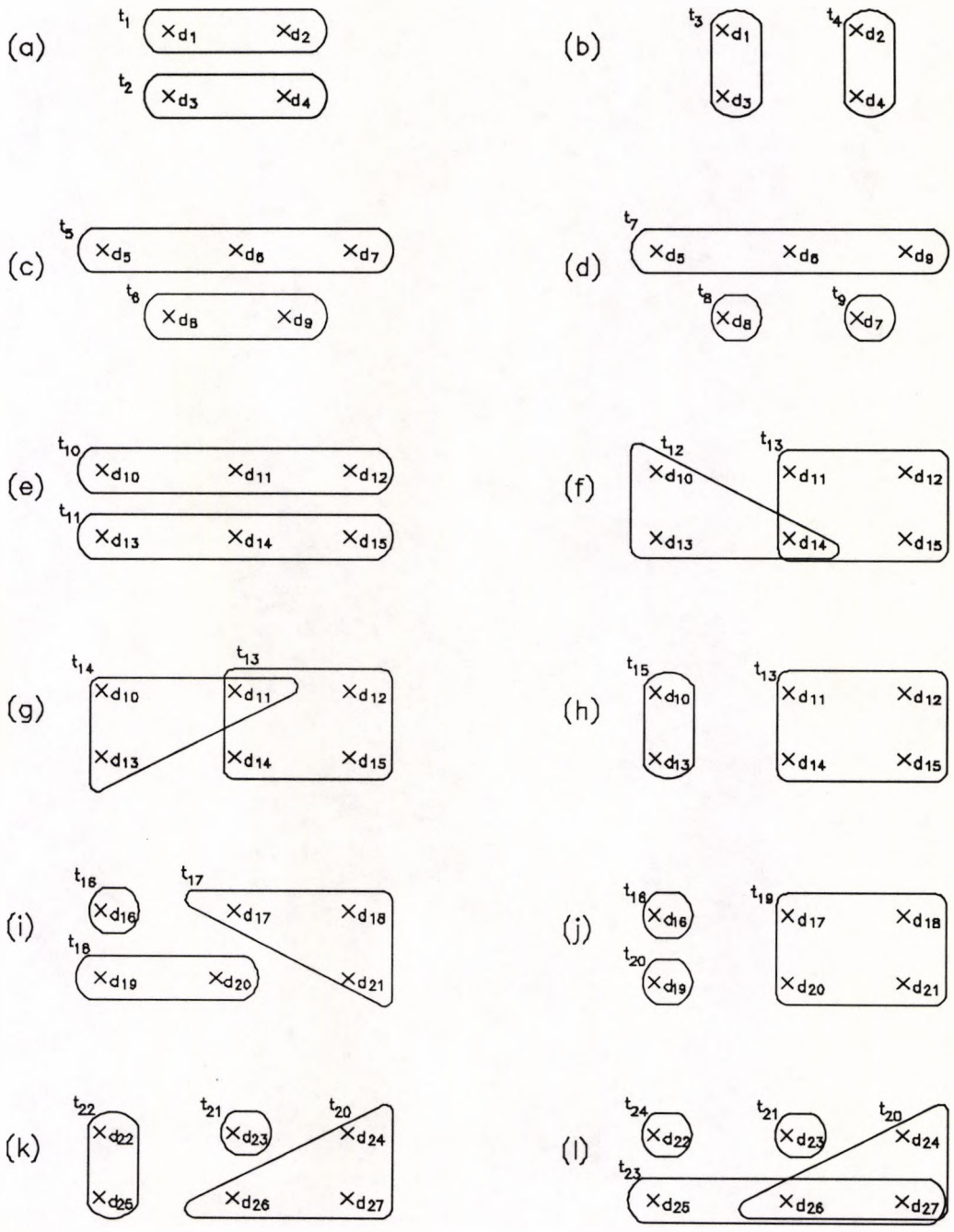


Fig.5.4: Examples of documents and the terms that cover them.

Having discussed the various decisions necessary in this module and their associated concepts (such as *minimal coverage*), following are the resulting *B* modules:

> B^I: Include all terms whose *document count* (no. of relevant documents in which it appears) is greater than a predetermined *threshold1*, *m.dc* > *threshold1*, (the value for this is discussed later in the section). Documents can be chosen with this simple criteria. If all documents are still not covered then more terms can be selected (by their document counts or weights in the case of ties).
>
> B^{II}: As per above, include all terms whose *document count* (no. of relevant documents in which it appears) is greater than a predetermined *threshold1*, *m.dc* > *threshold1*, (the value for this is discussed later in the section). If all documents are still not covered and there are ties in the *document counts* then terms can be selected according to their *weights*. However, in addition to the above module there is a limit set on the number of elements/terms in *A (ntA)*. Thus, the *proportion* referred to earlier (Decision 5) is used.
>
> B^{III}: Sort terms according to their document counts and sort for weight within that. Then, limit (using the value for *proportion* × *threshold2*) the number of terms that can be in in set *A*. The aim is to ensure that at least a certain proportion of terms relating to the current query are *Active* and that, therefore, there is a certain percentage of changeover of the *used context* terms. The number of terms to be included from the current query are a proportion of the maximum number of *used context* terms.
>
> B^{IV}: Include the term with the *highest document count* (for *minimal coverage*, as explained earlier). If all documents are still not covered pick the term with the next largest document count and so on. Where there is a tie consider the term weight. Here, there are no limits on the numbers in *A*.

The above versions of module *B* contained some variables (*threshold1*, *proportion*). The values for these can vary but the ones chosen for the tests described in 5.6.4 are explained. *Threshold1* is used to identify terms with a 'low' document count. For the tests, a value of 1 was chosen for *threshold1* i.e. those with a count of 2 or more. In a system where relevance judgments are not abundant[67] this seems reasonable.

The limits on the number of *Active* terms *(ntA)* is expressed as a certain *proportion* of the total number of terms that will be *Used* (*ntU*). In other words, *ntA* = *ntU* × *proportion*, the maximum for *ntU* being *threshold2* (see also module *C*). Using a *proportion* ensures that at least some the *old context* terms are kept. For example, initially 1/2 was considered reasonable, meaning that upto one-half of the total number of *Used context* terms (*ntU*) could be changed after a query with relevance judgements. Later, a value of 1/3 was also tested.

[67] This was the case for the Okapi IRS at the time of the experiments in this work.

These limits and proportions are particularly important if we assume that latter queries are more significant in ensuring an up-to-date representation of *context* (if it is changing).

5.6.3 Merging with Old Contexts

This section describes the last two modules *C* and *D*. Module *C* deals with the merging of the newly acquired *A, P* terms with the *old context*. However, this may depend on some historical information about a term which is addressed more specifically in module *D*. In other words, the various destinations for *A* and *P* terms are listed in module *C* while module *D* addresses the reasons.

Module C

The purpose of module *C* is to

- Merge the new terms (*A, P*) with the *old context*.

All *M* terms can potentially be in the *context*. Within a *context* there are also distinctions between terms which should be used (*U*) in improving IRS performance (i.e through improving document ranking), and those which are temporarily put on hold (*H*) but may at a later date be transfered to *U*. *U* and *H* terms are revised after each *learning* iteration.

The following decisions relate to the possibilities of merging, more detailed reasons for these can be found in module *D*.

> **Decision 6**: Whether an *Active* term should be put in *U* so as to be used, or put on hold in *H*. However, if we assume that later queries will give a closer approximation to *context*, then putting an *Active* term on hold would not be reasonable.
>
> **Decision 7**: Whether a *Passive* term should automatically be put on hold or in some circumstances be 'activated' and put in use (*U*). This typically would depend on the historical information about the term. i.e. how long ago it was in the *context* and what happened to it thereafter (see also module *D*).
>
> **Decision 8**: As implied in module B, in some cases a threshold needs to be set for the number of terms to be Used (*ntU*). This limit has been referred to as *threshold2*. In order to see whether a limit does make a difference, a few values should be tested. In the least, two 'extremes' such as a *low limit* and a *high limit* should be investigated.

The resulting modules are as follows:

> C^{I}: Put *Active* terms into *Use*. *Passive* terms can be put in Use or on *Hold* (depends on module *D*). There is no threshold value for the total number of terms in *U (Used)*.
>
> C^{II}: Put *Active* terms into *Use*. *Passive* terms can be put in Use or on *Hold* (depends on module *D*), as per above. However, ensure that the total number of

terms in *Use (ntU)* is below a threshold *(threshold2)*. In otherwords, $ntU \leq$ *Limit (Low or High)*. This module relates to the quality of terms (as per C^l) but also the size of the set of terms. In order to ensure that the U set size is below the set limit, in principle, it is possible for some *Active* terms to be secluded from U. However, in practice this does not occur often.

For the purposes of this work, after having done some trial runs to observe typical range term numbers, 6 and 30 have been chosen as *Low* and *High Limits*, respectively.

Module D

The purpose of module D is to specify the ways in which what happened to a term previously can affect its position in the *current context.* Hence, the aim to

- Use historical information.

Terms in a *context* are divided into U and H, as previously explained. At iteration n, we can refer to these as U_n and H_n. Thus, the previous U and H terms are referred to as U_{n-1} and H_{n-1}. It is the role of a term in U_{n-1} or H_{n-1} in affecting a current A or P term that is addressed here.

The decisions typically involve,

Decision 9: Whether a term previously in Use ($t \in U_{n-1}$) should remain so. If it has reappeared as an *active* term there is strong reason, as explained earlier, for it to remain in U. However, if in the current query it has reappeared as a *passive* term, it could be put in U or H.

Decision 10: Whether a term previously on Hold ($t \in H_{n-1}$) should be 'rejuvenated' or not. For example, if it is currently *active (A)* then there may be cause to put it in use *(U)*. On the otherhand, if it was previously on hold and is also currently *passive* again there is little reason to put it in use *(U)* - unless a certain quota of terms have to be met and there are not enough gathered through the other means.

Decisions 9, 10 are related to decisions 6 and 7 respectively and any limits that may apply. There are two parts to module D; D_α and D_β. D_α deals with the possibilities for the terms that were also in the previous *Used* set (U_{n-1}). D_β addresses the options for the terms that were in the previous *Hold* set (H_{n-1}). The possibilities for this module are illustrated in fig.5.6. and subsequently detailed. As is indicated in the diagram there are some options which have not been encompassed in the tests described later in this chapter. For example, if a term is currently *Active*, there seems little reason in putting in on *hold*. This is particularly so if one assumes that later queries are more representative of the *context*. Earlier (Sections 2.1.4, 5.6.1), two models of *context* - *stable* or *changing* - were discussed. If *context* is changing then the later queries and relevance judgements ought to provide a closer approximation to it.

For brevity, particularly in diagrams and tables, the notation U', H' instead of U_{n-1}, H_{n-1} respectively has been chosen.

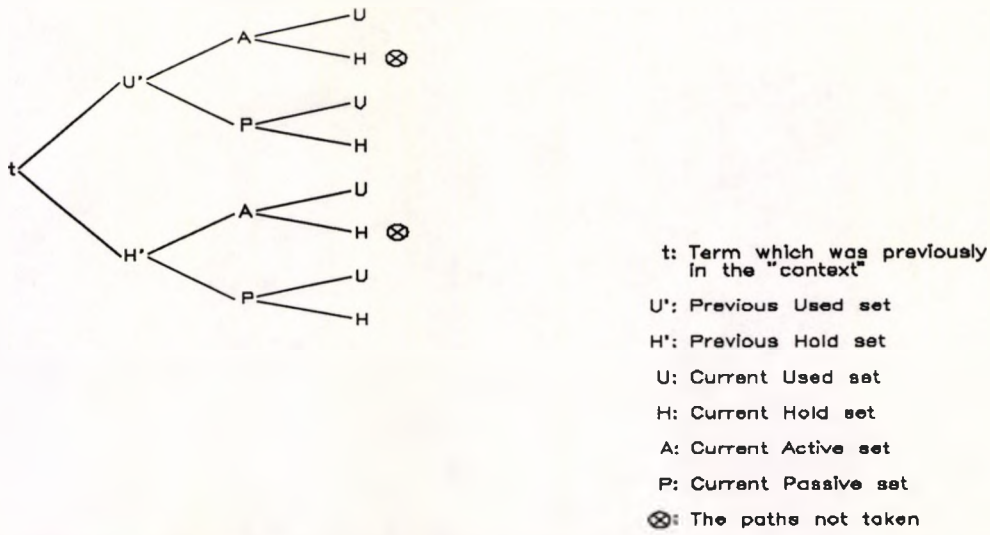


Fig.5.5: The possibilities for a term which was in the previous 'context' and has now reoccured.

For the above figure, the following stages apply:

1) If a term is in the old context then it has to be in *U'* or *H'*.

2) If the term has reoccured in the current learning iteration then it can either be *A* or *P*.

3) The result is that the term has to be placed in *U* or *H*.

Thus, following are the resulting options form module *D*.

***D*α^I**: Term *t* was previously in *U* and now is in *A* then it is kept in *U*. If, it is in *P* then it is also put in *U*. This means that a term which is currently *passive* can be put into *use* if it was previously in the *Used* set.

$[(t \in U' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in U' \wedge t \in P) \rightarrow t \in U]$

***D*α^{II}**: Term *t* was previously in *U* and now is in *A* then, like above, it is put in *U*. However, if it is in *P* then it is put in *H*. This means that it if a term was put on *hold* previously and is currently *passive* then it remains on *hold*.

$[(t \in U' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in U' \wedge t \in P) \rightarrow t \in H]$

***D*β^I**: Term *t* was previously in *H* and now is in *A* then it is put in *U*. If it is currently in *P* then it is also put into *U*. There may be cases where it may be useful for a term that was in the *context* (whether as *U* or *H*) before to be put into use, particularly when there may not be that many *U* terms.

$[(t \in H' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in H' \wedge t \in P) \rightarrow t \in U]$

***D*β^{II}**: Term *t* was previously in *H* and now is in *A* then, like above, it is put into *U*. However, if it is currently passive then it remains on *hold* in *H*.

$[(t \in H' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in H' \wedge t \in P) \rightarrow t \in H]$

In the cases where module *D* is used in conjunction with any *limits* on the number of terms in *A* or *U*, some prioritisation of the terms is necessary. This is done by sorting them in order of *time* and *weight* within that. The more recent ones have higher priority and the term weights are only compared for the same query.

5.6.4 The Tests

Prior to performing the tests described in this section, some informal analysis was made on the variations of the *context learning* algorithm. A group of consecutive queries for particular users were run on *learning* programs to ensure that what was being produced, at least initially, appeared to be reasonable and to identify whether it might be worth developing certain strands of the *learner* further prior to more formal evaluations (Chapters 6, 7). In some cases, those more expert in the field (of the particular queries) were consulted while in others the *context* terms were shown to users for their opinions and judgements.

Previously, the modules for the context learner, their variations and reasons were discussed. Testing for all permutations of these is not feasible in an implemented experimental IRS with real users and queries. The constraints are various. There are usually limitations on the number of obtainable users and queries, within reasonable timescales. Also there are hardware related difficulties that stem from the increasing size of data and number of programs.

Tests have to be done in light of evidence. Thus, they have to be structured as they are performed - new tests performed with the evidence from previous ones. The tests here are considered to be an example of how decisions can be applied. They have been ordered bearing in mind some control variables. As the numbering of the tests indicate, some were abandoned at the development stage and some after comparing the first few results. All those chosen (Table 5.8) have been formally evaluated - see Chapter 7.

The tests are as follows (Table 5.8) and their order and purpose are described after Table 5.9:

Modules		Tests													
		T1	T2	T3	T4	T5	T6	T12	T13	T14	T15	T16	T17	T18	T19
A	A^{I}		✓	✓	✓	✓	✓	✓	✓	✓	✓				
	A^{II}	✓													
	A^{III}														
	A^{IV}											✓	✓	✓	✓
B	B^{I}	✓	✓									✓			
	B^{II}												✓	✓	
	B^{III}			✓	✓	✓	✓				✓				
	B^{IV}							✓	✓	✓					✓
	T_1=1	✓	✓									✓	✓	✓	
	P=½			✓	✓	✓	✓								
	P=⅓										✓				
C	C^{I}	✓	✓					✓	✓			✓			✓
	C^{II}			✓	✓	✓	✓			✓	✓		✓	✓	
	T_2=6					✓	✓							✓	
	T_2=30			✓	✓					✓	✓		✓		
D	$D_\alpha{}^{I}$	✓	✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓
	$D_\alpha{}^{II}$				✓		✓								
	$D_\beta{}^{I}$								✓						
	$D_\beta{}^{II}$	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓

Table 5.8: Table showing the tests performed with their corresponding modules.

The dashed line in the table differentiates between the various options of the modules (those relating to different principles) and the different values for the relevant variables within these options.

As an example of the interpretation for one of the tests, test T14 uses modules A^{I}, B^{IV}, C^{II} (with a value of 30 for max. *ntU*), and $D_\alpha{}^{I}$ in conjunction with $D_\beta{}^{II}$.

To help interpret this table of results a summary table (Table 5.3) of the modules and their options are also provided.

Modules		Description
A	A^{I}	R_{latest} with duplicates
	A^{II}	R_{latest} remove duplicates
	A^{III}	R_{total} with duplicates
	A^{IV}	R_{total} remove duplicates
B	B^{I}	A term's doc count has to be greater than *threshold1* ($m.dc>T_1$), cover all docs*
	B^{II}	A term's doc count has to be greater than *threshold1* and limit *ntA* ($P \times T_2$)
	B^{III}	Sort terms according to doc count and weight then limit *ntA* ($P \times T_2$)
	B^{IV}	Aim for *minimal coverage* **
	$T_1=1$	*Threshold1* = 1, applies to no. of relevant documents a term occurs in *(m.dc)*
	$P=½$	*Proportion* = ½, *proportion* × *threshold2* sets a limit for *ntA*
	$P=⅓$	*Proportion* = ⅓, *proportion* × *threshold2* sets a limit for *ntA*
C	C^{I}	$(t \in A \rightarrow t \in U) \wedge ((t \in P \rightarrow t \in U) \vee (t \in P \rightarrow t \in H))$
	C^{II}	$(t \in A \rightarrow t \in U) \wedge ((t \in P \rightarrow t \in U) \vee (t \in P \rightarrow t \in H))$ with *threshold2*
	$T_2=6$	*Threshold2* = 6, the max. no. of terms in the *Used* set *(ntU)*
	$T_2=30$	*Threshold2* = 30, the max. no. of terms in the *Used* set *(ntU)*
D	D_{α}^{I}	$[(t \in U' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in U' \wedge t \in P) \rightarrow t \in U]$
	D_{α}^{II}	$[(t \in U' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in U' \wedge t \in P) \rightarrow t \in H]$
	D_{β}^{I}	$[(t \in H' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in H' \wedge t \in P) \rightarrow t \in U]$
	D_{β}^{II}	$[(t \in H' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in H' \wedge t \in P) \rightarrow t \in H]$

Table 5.9: Summary table of the modules and variables for the Context Learner.

* To cover all documents, if first criteria does not already, select from remaining terms with highest document counts and weights (in the case of ties).

** The phrase *minimal coverage* has been used to briefly describe the strategy addressed in module *B* in more detail.

There follows a list of abbreviations for the above table.

Abbreviations:

R_{latest}: Set of relevant documents from last query
R_{total}: Set of relevant documents from all queries
A: *Active* set (for current iteration *n*)
P: *Passive* set (for current iteration *n*)
P: Proportion of threshold2 which sets the maximum value for *ntA* - for modules B^{II}, B^{III}
T_1: *Threshold1*, applies to no. of relevant documents a term is found in - for modules B^{I}, B^{II}
T_2: *Threshold2*, max. no. of terms in the *Used* set - for module C^{II}
ntA: No. of terms in *Active* set.
ntU: No. of terms in *Used* set.
U: Current *Used* set (also referred to as U_n for current iteration *n*)
U': Previous *Used* set (also referred to as U_{n-1})
H: Current *Hold* set (also referred to as H_n for current iteration *n*)
H': Previous *Hold* set (also referred to as H_{n-1})
α: U' or U_{n-1}
β: H' or H_{n-1}
D_α: *D* modules concerned with α set
D_β: *D* modules concerned with β set

Note: For brevity, the sets U, H, A, and P at iteration n (current iteration) have not been referred to as U_n, H_n, A_n and P_n.

Note: For T17 and T18 there is no *P* value as there is only one iteration *ntA* = *ntU*.

The purpose of the Tests

A *context learner* or *learning program* consists of a combination of the modules previously described. A test involves running this program and comparing it with the results of others so as to identify the effect of particular variables.

The ordering below is to explain the rationale of the tests.

1) Test for the effect of duplicates in R_{latest}. Perform **T1**, **T2** and compare their resulting *terms*.

2) Introduce *sorting* criteria, as per module B^{III}, with limits on term numbers where necessary. This is an alternative to the earlier more simple module B criteria applied (m.dc > 1). Hence, perform **T3**.

3) Change the role of a currently *passive* term *(P)* previously in the *used context (U)*. Try putting it on *hold* (in *H*). Perform **T4** and compare it with the resulting *terms* of T3.

4) Try T3 with lower limit (6) on *used context* terms. Perform **T5** to compare particularly with T3.

5) Try T4 with lower limit (6) on *used context* terms. Perform **T6** with a view to comparing especially with T4.

6) Introduce *minimal coverage*, keeping D modules as per tests T1, T2, T3, T5. Perform **T12**.

7) Try *minimal coverage* with a different role for a currently *passive* term *(P)* previously on *hold* (in *H*). The D module is changed so that the *P* term is kept on *hold*. Perform **T13**.

8) Try *minimal coverage* with limit on the number of *used context* terms, keeping module D as per tests T1, T2, T3, T5, T12. Perform **T14**.

9) Go back to T3 but try a different proportion for the *active* terms in a current learning iteration. Perform **T15**.

10) Introduce R_{total} as an alternative way to forming the set of relevant documents (see module A). Keep remaining modules as per the first test T1. Perform **T16**.

11) Try T16 with a limit on the number of terms in the *used context* due to the size of the resulting *contexts* of T16. Perform **T17**.

12) Try T17 with much lower limit (6). Perform **T18**.

13) Try *minimal coverage* for R_{total}. Perform **T19**. As the resulting number of *used context* terms are quite small there is no need to try for a lower limit such as 6.

Having described the various *modules* of the *context learner* and their associated *tests*, the next chapter describes the IRS which the tests have been applied to, its weaknesses and the scope for a *context learner* within this IRS.

Chapter 6

The Role of a Learner in Okapi

The previous chapter described the *context learner* and its modules. This chapter focuses on the experimental *probabilistic* IRS, Okapi, to which the *context learner* has been applied.

The chapter describes the Okapi system, its queries, patterns of usage and the probabilistic model which it is based on. The possible sources for learning, the various means of recording information and the weaknesses of the system are also discussed along with ways in which they may be be improved upon with learning, in particular *context learning*.

6.1 Description of Okapi

The Okapi system is a fully implemented, interactive experimental information retrieval system. The Okapi systems are a class of systems providing a skeleton for evaluating different techniques. The Okapi projects have been spread over a variety of locations such as the Polytechnic of Central London, University of Bath and City University, London. The system used as part of the work for this thesis is a version implemented at the City University. Appendix A (Okapi System Description) contains a detailed description of the system along with screen layouts presented to the users.

The work at the University's Centre for Interactive Systems Research (Department of Information Science) involved setting up the Okapi IRS as a generalised facility in order to do some experiments and evaluation on an interactive IRS (Walker and Hancock-Beaulieu, 1991).

Users were able to search the University's Library Catalogue, Inspec (The Institute of Electrical Engineers) and LISA (Library and Information Science Abstracts)[68]. Details of the databases are as follows:

-The City University Library catalogue contains some 155,000 records.

-The Inspec database, at the time of this experiment, initially consisted of 95,000 and later some 224,000 records.

-The LISA database is the complete set of records for 1976-1989 and some later material amounting to some 100K.

[68]This database is no longer available with the latest version of Okapi at the City University.

Work on the Okapi projects has been in progress since 1982. It has concentrated on research and development in the field of bibliographic retrieval for direct use by end-users. The earlier projects were involved with producing an OPAC on a LAN. In these projects, search decision trees were used to perform Boolean searching automatically (Mitev and Walker, 1985).

Later versions of Okapi included automatic stemming, truncation and functions to improve subject retrieval. With stemming and truncation it was hoped that through systematic abbreviation of morphologically related words any semantic relationships between them could also be exploited (Walker and Jones, 1987). Various ways of dealing with misspellings or miskeyings were looked at. The current system, however, uses the simplest of these methods in which search terms not found are reported to the user so that they can correct them if they have made a mistake.

Evaluation on previous projects showed that subject retrieval was more common than specific item searches. Hence, more work was done to improve the performance of subject searches. In later systems (search) terms were weighted in accordance with their relative frequency in the indexes (Mitev and Walker, 1985) where rare terms were given higher weights than the more commonly occuring ones. For displaying the search result to the user, the records are ordered by *weight (document scores)*. These scores are the sums of the weights of the query terms that result in the document being retrieved. Hence, the result of the search is a list of records ordered by document weight/score. The implementation concerned uses the probabilistic model, as described in Section 6.1.3, to calculate the term weights.

6.1.1 Installation and Access

The Okapi system described is installed on a Sun SPARCstation 330 with 16 MB memory and 900 MB disc storage[69] (Walker and Hancock-Beaulieu, 1991). Users can access the system over the campus wide network (ethernet) or from a dedicated terminal at the library. This terminal accesses the University's library catalogue only.

For monitoring and research, it is important to be able to identify which sessions belong to which users. Network users are asked to fill in a registration form and are then given user-ids. Library users are asked to type in their library card number. Should this not be possible they are requested to type in a rather more difficult set of character/number combinations only after which they are given access to the system. This is done to encourage them to use a constant identification when using the system so that any evaluation on the searches can be as correct and consistent as possible.

All users, network and library, are informed that they are using an experimental system and that some data from their sessions may be used for evaluation purposes.

There is an extensive automatic user logging facility. An example log of a session can be found in Appendix B (Okapi Transaction Logs). A compromise had to be made between the log being understandable by humans and being interpreted/processed by the computer programs.

---

[69]The hardware configuration has been upgraded since.

6.1.2 System Use

The use of the Okapi system in the library has been steady though not very heavy. There were some 1800 user sessions during the first six months of installation. About half were by identifiable users - identifiable through the process described in the previous section. During this period there were about 80 network users registered of which most used the system at least once, a total of 682 network user sessions. Of these, 47% used the Inspec database, 42% used the City University Library Catalogue and 11% used the LISA database.

6.1.3 The Retrieval Process

The information retrieval process in Okapi consists of the following stages:

- User input, preprocessing, parsing and stemming,
- Searching and the probabilistic model,
- User feedback and relevance judgements,
- Query expansion.

User Input, Preprocessing, Parsing and Stemming

After some general system information and introductory screens, the user is presented with an input screen for the subject search. There is no controlled language that they must adhere to in order to express their query nor need this produce Boolean expressions. They can type in a single word or any combination of words or indeed a "natural language" phrase (Appendix A: Okapi System Description).

The system deals with this input by first putting it through an input preprocessor to enable parsing and lookup of the appropriate terms in the index. For example,

E.g. After input pre-processing,

"Input pre-processing & information retrieval in the U.K." becomes

"input preprocessing and information retrieval in the uk" - upper case letters are converted to lower case, signs (&) are converted and any "." removed.

The constituents of the preprocessed search statement are then looked up in a database called the gsl (go/see list). As Walker and De Vere (1990) state, this database contains some "rather meagre linguistic knowledge". The entries include: Stop words (e.g. "the"), common prefixes (e.g. "anti"), synonymous words (e.g. "united kingdom", "uk" and "great britain") and phrases which behave like single words (e.g. "information retrieval").

If a query term is in the gsl then it is processed/stemmed according to its category. The resulting terms are looked up in the indexes and the user is given some feedback as to how many references were found under each term. Thus, in the above example, four terms "input preprocess ir uk" would be looked up after parsing and stemming.

Searching and the Probabilistic Model

The probabilistic model used in this version of Okapi aims to reflect the probability that a document is relevant to the query by attaching numerical measures/weights to it using a "probability ranking principle" (see also Section 2.6).

In order to estimate probabilities, simplifying assumptions have to be made. For example, almost all practical implementations assume that document descriptor terms occur independently of each other (among both relevant and non-relevant documents in the collections). With this independence assumption we can estimate the probability of each descriptor term occuring in relevant and non-relevant documents independently. By cumulating the weights of the query terms that index/describe a document we arrive at a numerical measure reflecting a probability of relevance for that document. These measures can then be used to place the documents in descending weight order so that the relevant ones are likely to be nearer the top of the list.

Hence, term weights are assigned with the formula below:

$$w=\log\frac{p(1-q)}{q(1-p)} \qquad \text{F 5}$$

where

p: the probability that the term will occur in relevant documents

q: the probability that the term will occur in non-relevant documents

With relevance feedback, p and q in the above equation can be estimated to

$$p=\frac{r}{R} \qquad q=\frac{n-r}{N-R} \qquad \text{F 6, F 3}$$

where

n: Number of postings for the term (number of documents containing the term)

N: Number of documents in the collection

R: Number of records chosen as relevant

r: Number of chosen records containing the term

Thus, equation F1 becomes

$$w_i=\log\left(\frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)}\right) \qquad \text{F 4}$$

where 0.5 is added to each of the components in order to avoid infinite or indeterminate values and increase accuracy when there is little relevance information. The derivation of this, is given by Robertson and Sparck Jones (1976) and is also explained in Efthimiadis (1992).

In the absence of relevance information, as is the case with the initial search, p and q in equation F1 are estimated differently. Croft and Harper (1979) proposed that identical relevance-probabilities (p) should be assigned for all the terms in the query. As for the non-relevance

probability (q), since most documents in a collection will be non-relevant to most queries, this can reasonably be estimated as the proportion of documents in the whole collection which contain the term (n/N)).

Hence, p in F1 is the same for all the terms and $log(p/(1-p))$ can be represented by a constant c and the term weight becomes

$$w_t = c + \log\left(\frac{(N-n)}{n}\right) \qquad \text{F 5}$$

However, various values could be assigned to p in obtaining the constant c. When p approaches 1, c becomes large in comparison with the second component of the above equation (also known as coordination level matching.)

Should p be 0.5 then c becomes 0 and the term weight becomes

$$w_t = \log\left(\frac{(N-n)}{n}\right) \qquad \text{F 6}$$

This is almost the same as the inverse frequency weighting (Sparck Jones, 1972)

$$w_t = \log\left(\frac{N}{n}\right) \qquad \text{F 7}$$

Experiments have been done on such values and Croft and Harper suggest those between 0.6 and 0.9 for p. Okapi systems, however, use 0.5 as a value for p, thereby taking no direct account of the number of terms common to the query and retrieved documents. N is a constant which must be larger than the number of postings for the most highly posted term in the search (Walker and De Vere, 1990).

Thus, the weight of a descriptor term initially depends on its frequency throughout the database. Terms occuring more rarely (eg: "filter") have higher values and those occuring more commonly (eg: "analysis") are given lower ones.

After deriving the term weights, document details are presented in descending document score/weight order. The user is also given an indication of how well the resulting list of documents match the query such as "one book matches your search well". A document score (or matching function) is arrived at by summing the weights of the query terms that index it. Should several documents have the same score then they are ordered chronologically (with the most recent at the top) and within that in alphabetical author order.

There are two threshold scores which the system considers in displaying the document list to the user. One is a score above which a record will match "quite well", the other below which the record will not be retrieved at all (not acceptable) (Walker and De Vere, 1990). Examples of these thresholds can be found in Appendix A (Okapi System Description).

User Feedback and Relevance

After typing in a query, the user is presented with a list containing the brief description of records (part of title, author and date). At this point the user can choose to see more details on a specific record by selecting its number (see Appendix A: Okapi System Description for a view of what is presented to the user). Once they have looked at a record in more detail they are asked to make a relevance judgement on that document/reference with the question "Is this the sort of thing/book you are looking for?". They must answer this question with "yes" or "no" before they can resume (Walker and De Vere, 1990). This is how the relevance information necessary for the F4 formula above is obtained but it is used only after a More option is specified.

With this type of relevance information it then becomes possible to do query expansion. Query expansion is initiated by the More option which the user initiates by choosing "to see More like the ones (you have) chosen".

Query Expansion

Query modification can be done through relevance feedback. With the information from this feedback, it is possible to expand the query in order to help improve the result of the original search (see also Section 2.6). It is hoped that with automatic query expansion (AQE), other descriptors of records the user has judged to be relevant are also likely to be useful in retrieving additional relevant records. This pool of terms from the relevant records can be assembled in many ways from the different component fields of the record. Exactly which fields should be used can be set by a parameter. In this implementation the *title, subject headings, classification code* were used for the Library Catalogue and *title, feature headings and descriptors* were used for the Inspec and LISA databases (Walker and Hancock Beaulieu, 1991).

Weights are calculated for these terms with the Robertson and Sparck Jones F4 formula (referenced above), given previously, and the terms sorted in descending weight order. The ones above a certain cut-off point are those used for the query expansion. The maximum number of terms is a parameter that can be set (32 in this experiment). The terms chosen are looked up in the indexes and a merge performed just as in the original search (Walker and De Vere, 1990). Any records already chosen or rejected by the user are removed from the resulting document set. The user invokes the AQE facility described when they choose "to see More like the ones they have chosen".

6.2 Data and Information Sources

There are two types of data and information available. The first can be referred to as the "raw" material itself, the second being any further statistical or other analysis derived from this. The following sections describe these two types of information.

The "raw" material or "source" of data/information for analysis consists of the following:

a) The databases and the index/term structure and data.
b) The gsl (go/see list) for each database.

c) User information.
d) System information.

a) The database

The databases themselves contain information about a document. The database records have various descriptor fields such as *title, author, subject headings and publication.* Here, there is information about the documents and about terms, e.g. term frequencies are generated in preparing the indexes.

b) The gsl

The Go-See-List is adapted to each database. It is able to hold and cross-reference various classes of words. It can perform some of the functions of a thesaurus (identifying synonymous relationships between words/phrases) and eliminate stop words which will not carry much meaning in IR terms (see also Section 6.1.3).

c) User information

There is some background information mainly pertaining to network users. This consists of the details on their system registration form such as their name, department and email address. A secondary form of background information is collected in the process of interviews. Some frequent users of the system were interviewed as part of the Okapi experiments (Walker and Hancock-Beaulieu, 1991) (see also Section 6.3.2).

There are also the user transaction logs which contain a substantial amount of information about user searches. There is one log per user online session: chronological records of what has happened during this session. They include the queries, the modifications, any listings of references that the user has seen, those references found to be relevant by the user, the terms added by the system to expand on the query, some timing information, user keystrokes and some summaries on the total number of functions used in that session. Please refer to Appendix B (Okapi Transaction Logs) for an example.

As is quite often the case where a lot of information is recorded about the "events" during an online session, the data is not always in a format that is easy to process. Often a compromise has to be made between storing the data in a form that makes it easier for humans to analyse and storing it so as to ease further processing in order to obtain statistics, summary information etc. In practice, more machine-readable versions of the logs were used with the information summarised in the form of tables to ease processing. Users' online sessions can thus be analysed to identify any inherent patterns in their searches, sessions or behaviour, not just for individual users themselves but also for specific groups of users.

d) System information

This includes details of system usage and access by users. There is also some other system information available, for comparison purposes, from earlier Okapi experiments. Details include number of search statements and average number of words per statement (Section 6.3.5).

All the above material can be used to identify particular requirements or possible areas of improvement in the system. It should also form a strong basis for any system adaptation. The adaptation can be automatic during a session or can be done off-line. The latter would use system and user performance analysis for subsequent generations of improved versions of the system.

6.3 Analysis of User Data

The analysis of user data was a preliminary investigation done before the *context learner* (described in Chapter 5) was fully developed and implemented. It precedes the two stages of evaluation described in Chapter 7. The aim is to identify specific problems in an implemented IRS and establish whether *context* exists as per the *context hypothesis* (Chapter 4). In addition, the purpose was to establish whether a *context learner* may help with any of the problems. Hence, user data relating to their online queries was analysed (subjectively and more formally) towards this end.

The analysis that follows concentrates mainly on the data about the users and the system (*c)* and *d)* of Section 6.2). Following is the type of information that was gathered:

1) Information about the patterns and context of frequent users,
2) Background information about the users,
3) Information about the terms (stemmed and parsed queries) used by individuals,
4) Information about the terms (stemmed and parsed queries), their variety and frequency and distribution amongst all users,
5) Information about user queries and their amendments.

The above are expanded on in the subsections below.

6.3.1 Pattern and Context of Frequent Users

In order to obtain information about the patterns and context of the user searches and their behaviour within the system, frequent users' logs were looked at in depth. There were some 20 users (mostly network) who, over a period of three months, were identified as having used the system frequently. "Frequent usage", for this stage of analysis, means the user has done at least 5 sessions on at least 4 days. The library users[70] are referenced as LIB-A, LIB-B etc. and the

[70]The library users are those using Okapi with a library card number via the machine available in the library and so able to access the City University Library Database only.

network users are NET-A, NET-B and so on. Appendix C (Assessment of Frequent Users' Logs) contains an assessment of the individual frequent users' queries and sessions.

The logs of each frequent user were analysed and notes in the form of three categories were made. These were the general subject areas queried by the user, some overall impressions about the sessions, and more specific details and examples.

Diagrams to help summarise the processes and stages in particular user logs were also drawn. Please see Appendix D (Frequent User Session Summary Diagram - An Example) for an example of a frequent user's session.

6.3.2 Interviews to Expand Background Information

A limited amount of background information is available about those users who register over the network as mentioned in Section 6.2. For those who use the system only from the library this information is not available as there is no registration process. However, interviews were conducted with the frequent network users in order to check, for consistency, the analysis made solely from their logs and to further clarify any other issues or impressions the users have of the system.

The interviews were, on average, about 20 minutes long. The questions came under four headings: context; okapi-general; search-details; conclusion (Walker and Hancock-Beaulieu, 1991).

After asking/checking the users' name and departments, the first section aimed to ask more information about what their field of work was perceived to be, what stage they were at in their work and whether they had connections with others with whom they regularly exchange/share information on these topics.

The second section related to their perception of the Okapi system in general. For example, if they found it easy to use, whether they used all the databases, which of those they found the most useful and whether they felt they had to adapt/change their search strategy over time.

More questions about the way they performed their searches were asked in the third section. They were asked how they went on to choose which records they wanted to look at in Full from the Brief records' list, how they decided that a reference was relevant, if they felt they had to look through a lot of references to find useful ones, whether they noticed any points indicating different levels of relevance (i.e. *"The rest of the searches may match your search less well"*), and whether they were using the More (query expansion) function and if so whether they were finding it useful. The words "they felt" have been included in this paragraph to stress the users' point of view and how they perceived their searches and results and not necessarily what might be statistically true. For example, users have on occasions thought that they used the More function regularly when statistically they had only used it once.

Concluding questions were asked: whether there were any particular features they liked/disliked about the system; whether they had used any other online bibliographic retrieval system before

or performed the manual equivalent of the search (i.e. looking up search/key words in the abstracts index to find links to bibliographic details which can then be followed).

6.3.3 Terms Used by Individuals

Queries were parsed and the remaining words stemmed (e.g. "translating" and "translator" become "translat"). It is these stemmed terms that are used for information retrieval. Thus, their frequencies and distributions are used rather than those of the original words in the queries.

Programs were written to obtain listings of the terms used by any particular user, their weights, and the total number of times the terms have been used. An example of such a listing for a particular user can be found in Appendix E (Term Details for Frequent Users - An Example). For example, the particular user (NET-D) in the Appendix quoted used the term "robot" 6 times but "parallel" only once. Although the listings can be generated for any user, those for the frequent users have been concentrated on.

6.3.4 Variety and Frequency of Terms Used

Following on from the individual analyses, further programs were written to analyse the usage of terms over all users. They list all the terms used by users, along with their weights, total number of usages and the identification of those who used them.

Please see Appendix F (Term Usage - An Example) for an extract of a listing for the terms used by the library users. For example, the term "art" was used 9 times (over the network) by 5 different users.

6.3.5 User Queries

Some figures have been obtained on user search statements such as the number of words in a search statement and the number mispelt. In particular figures were obtained on user query modifications (not just AQE). The types of query modifications and the ways in which they have affected their previous or original query (narrow/broaden) have been analysed.

For this, a suitable size data batch could be obtained from the users using the "special number" when using Okapi from the library. The special number is a random number which users have to type in if they do not have their library card number available, the number is changed once it has been used. The version of Okapi running in the library accesses the City University Catalogue database.

In addition to the above, the query amendment patterns of users was analysed. For this, two batches of listings were obtained from 544 searches taken from a total of 300 session logs. One batch contained all queries which had been subsequently "edit"ed; the other those followed by a "new" search. These two functions will be referred to as <edit> and <new> searches. Both batches of searches were created to analyse just how many of the "edit"s could have been "new" searches and vice versa. The analysis was done both in the 'syntactic' and 'semantic' sense. This was done also to establish the genuine number of the new and edited searches from the users and the effectiveness/appropriateness of the <new> and <edit> functions.

Syntactically, the aim was to see whether words were generally being added or deleted. For example, the query "international banking and finance" could be modified to be "international banking and finance in the EEC" or simply "international banking" (an addition or removal of terms from the previous query).

Semantically, the aim was to identify any direction to the users' queries, such as a broadening or narrowing (or indeed whether they were still related). Adding more words to a query does not always result in a more narrow search. This depends on the nature of the words/terms added. In the above the example, the modification "international banking and finance in the EEC" results in a narrowing of the original query whilst "international banking" broadens the original. However, if the original query had been "international banking" then modifying it to "international banking and affairs" would not have necessarily narrowed the search.

6.4 Results of User Data Analysis

The types of user data analysis have been discussed in the previous section. This section shows the results and discusses to which degree they may be consistent.

6.4.1 Pattern and Context

As mentioned previously, the Appendix C (Assessment of Frequent Users' Logs) contains details of individual users and some overall comments. A summary of the general points on frequent user logs with indications of some deficiencies in the current system is given. These and the ways in which they can be improved are further discussed in Section 6.5.

General Points About Frequent Users:

Having analysed the logs for about twenty frequent users, some general observations are listed below. Appendix C (Assessment of Frequent Users' Logs) contains further details on these, particularly those relating to individual users. These points are summarised in the section below. Some of the points may relate to machine learning for Okapi; others may simply relate to altermative ways of adapting the system to further help users.

Nature of User Query Input

• It would appear that on average users do not have as wide ranging a set of queries as might be expected. In fact, often they focus on two or three 'subject areas'[71]. This is what the context learner aims to make use of and is also expanded on in Section 6.6.

• When the sessions are long, users sometimes lose their place and re-do some of the functions in order to remember what they have done or where in the current (long) task they are. This could be an indication of the usefulness of some kind of optional summary information. For example, details of how many More functions (query expansions) they have done and what their

[71]The phrase 'subject area' is used loosely here but the issue has been addresssed in greater detail in Chapter 2.

recent searches were (without having to go into Edit/New Search Menu which only show the last three queries).

Nature of Reference Lists Presented to Users

• Although the references are listed in descending weight order, often it is not possible (especially with INSPEC) to tell enough about the reference from the part of the title that is shown. Hence, it is not too uncommon for users to make their own relevance judgements by expecting words in their query to occur in the part of the title that they can see. A large number of records are looked at only in Brief form. Some frequent users interviewed have indicated support for the idea that more could be shown about a reference i.e. two lines per reference, as opposed to often having to see the record in Full and finding that simply having more information on the title would have been sufficient for the user to see if it was irrelevant (e.g. Users NET-B, NET-E in Appendix C (Assessment of Frequent Users' Logs).

• It would appear that, especially in the Inspec database, the relevance level points *("The rest of the references may match your search less well", "The rest of the references may not match your search very well")* do not appear until after several screens - in some cases about 25 (this is on the smaller size Inspec database). Within this list, it is quite likely (especially if only one or two general terms which have been used in the query) that there would be large "chunks" of records belonging to the same weight category. Hence, the user sees nothing but what seems like an alphabetical (by author) list of references for the first several screens.

This results in them sometimes logging out because of frustration before looking further, or missing some references which they may pick up in future searches, perhaps having less confidence in the relevance judgements of the system, or simply having to spend a lot of time looking through seemingly unordered references.

With the frequent users analysed, it was found that almost all of them had at least one such occasion amongst their searches. (e.g. users/logs NET-A, NET-B, NET-C/1, NET-C/9, NET-D/2, NET-D/9 NET-E/11, NET-E/12, NET-F/4).

6.4.2 Interviews to Expand Background Information

All users are informed when registering that they are using an experimental system which aims to provide a "real" service whilst learning about "real" users. They are informed that, to this end, data is kept and that they may be approached for further help with the research. It was, however, found that some users were slightly disturbed when shown an example log, in an interview. Hence, some discretion was required (Walker and Hancock-Beaulieu, 1991).

The interviews were also conducted as part of other Okapi experiments (Walker and Hancock Beaulieu, 1991). The purpose here, however, was essentially to observe if they were consistent with assessments made on logs of user IR sessions.

The background information obtained about the frequent network users looked promising in the sense that the interviews confirmed the major points about a user and their search behaviour

which had already been deduced from analysis of the logs. It was more difficult to get information on specific points as often users found it more difficult to remember queries done several weeks ago.

6.4.3 Terms Used by Individals

The term frequency listings for the terms used by the frequent users (see Appendix E: Term Details for Frequent Users - An Example) are consistent with the general impressions about the user subject areas as found in the analysis of their logs (Section 6.4.1). When these lists are used in conjunction with the Appendix C (Assessment of Frequent Users' Logs) they give a better idea of the pairing of the terms. For example, Net-D had queries on "robot vision" and music - "hifis", "sound", composers such as "Bartok". These term frequency listings provided an alternative summary view of the queries consistent with previous observations.

6.4.4 Variety and Frequency of Terms

The lists of terms with the users who used them provides information about the frequency of terms and their distribution amongst users (not only the frequent ones). Thus, it is possible to identify the more common and "less popular" terms.

Analysing these also confirm what is deduced by looking at the logs of frequent users. However, analysing individual users' logs and making notes and subsequently linking them to others' subject areas is painstaking and can be achieved more quickly with this method.

Unsurprisingly, terms used frequently tend to be 'general' ones from a subject point of view. Appendix F (Term Usage - An Example) also shows a sample of which terms are used frequently and those which are not. Further, studies might focus on the frequency of use in queries and frequency of use in collection and whether words used by more than one user could indicate potential of learning from one user to another.

6.4.5 Query Analysis

This section emphasises the analysis of existing queries and sessions rather than user interviews conveying what the users think or remember they did.

Queries had also been analysed on a previous version of Okapi. Following are some of the general statistics obtained:

In 10042 search statements there were
2.7116 words on average in each statement (including stop words),
5157 total unique words including spelling mistakes,
1842 words were not in the dictionary (either mispelt or proper nouns or simply unusual words),
3500 words were in the dictionary and spelt correctly.

On a current version of Okapi, the following figures were extracted from the users using the "special number" in the library.

300 Total searches contained
538 stems
303 unique stems

In one of the later Okapi experiments on the University's Library Catalogue, the effectiveness of the query expansion was also measured and it would appear that searches where the query had been expanded (and where the user chose to see any resulting references) led to a significantly higher number of items being selected as relevant. It does also seem that in about 50% of the cases, using the "More" option did not result in any further relevant items being retrieved. Other experiments also showed that users were equally likely to use the More option whether they were familiar or unfamiliar with the system (Walker and Hancock-Beaulieu, 1991).

This section looks into the process of query modification within a session. Thus providing more insight as to how and what extent queries follow on from each other. Apart from AQE (as described in Section 6.1.3), queries, in Okapi, can be modified in two ways. The user can either <edit> the query or type in a <new> one. In the <edit>ing option, the last query is copied accross and the user can then add to or remove from it to form the next query. We can analyse the queries which have been modified in either of these ways to identify the relations between consecutive queries in a session. The relations have been examined in two ways: syntactically and semantically (Section 6.3.5). In the case of the later, the 'meaning' and content of the query and its modifications were examined for semantic or contextual links. With the syntactic analysis, however, the queries were only compared to see whether words/letters were removed or added without checking for their content. Both <new> and <edit>ed queries were analysed in these two ways described.

Results

The query modification analysis was performed on 300 session logs comprising of 544 queries and the results can be categorised in the following ways:

- the <new> searches
- the <edit>ed searches
- syntactic and semantic mappings between searches

Table 6.1 below shows the number of <new> and <edit> queries and the sessions containing them.

Observation	Total No.	<New>	<Edit>
Sessions	300	75	45
Queries	544	165	72

Table 6.1: The number of <New>/<Edit> sessions and queries.

The figures in Table 6.2 are discussed in the sections on <New> and <Edit> searches/sessions which follow.

Function	Total No. of Searches (Queries)	No. of searches disregarded	No. of Potential Searches	No. of Potentially Genuine Searhces	No. of Genuine Searches
<New>	165	32	133	124	37
<Edit>	72	14	58	52	45

Table 6.2: The number of potential and genuine queries for <New> and <Edit> searches.

Potential Searches ("Practical" Searches): These are the remaining queries (in practical terms) after disregarding those containing logouts, timeouts, misunderstandings of the system.
Potentially Genuine Searches: After syntactic analysis of the 'potential searches' above, these are the remaining ones that fit their corresponding category. For example, for the queries initiated by the function <New> search they were compared with the previous query to see if syntactically it was a 'new' search.
Genuine Searches: Similar to the above category of searches but here the semantics of the query are analysed. For example, these are the remaining searches after the syntactic analysis which can also be considered to be semantically <new> searches.

<New> searches/sessions

Of the 544 queries 75 were followed by at least one <new> search. (this is equivalent to the number of sessions in which <new> was done). The <new> function was used a total of 165 times. Thus the relation between these 240 queries is discussed here (Table 6.2).

Although <new> was used 165 times, 32 of these were disregarded for analysis due to logouts, timeouts, non-meaningful searches etc. Thus we are left with 133 searches which can be potential <new> searches. The term "potential" is used to define the maximum number of searches which could belong to this category of <new> searches prior to any syntactic or semantic analysis. These 133 searches can considered as "potential practical" searches. The term "practical" is used to express whether the facility intended (in this case <new>) was the practical option given the search that followed.

E.g.1: 1) child psychology
2a) child psychology and Freud
2b) Freud and child psychology

E.g.1 shows that given query 1), <edit>ing it to form query 2a) would be more practical as it would involve less typing. On the other hand, if query 2b) was the intended one, choosing the <new> option would have been the more practical new (as would not have to delete old query before inputing again)

Inspecting the searches syntactically showed that with respect to their preceding search statements (usually the original query) 9 should have been expressed using the <edit> facility. These 9 searches do also contextually follow on from their preceding ones, however, <new> searches need not always be related to the preceding query.

Here, it is useful to talk of the remaining 124 searches (after removing the 9 which should have been <edit>s) as potentially "genuine" searches. That is the maximum number which would be contextually new searches, where the user appears to be changing search topic. The searches up to this point were analysed syntactically. However, analysing the <new> searches semantically showed that only 37 were indeed what might be called "genuine" <new> searches. In other words, only 22% of all <new> searches appear to fit the function.

Thus as also shown in E.g.1, E.g.2 shows that the <new> facility may be used due to ease of editing even though conceptually the query may not be "new" but rather an edit or extension of the preceding search statement.

E.g.2: 1) Business finance
2) Finance and the firm

Although search 2) could have been keyed in as a <new> search for practical reasons, conceptually it is not a new search.

<Edit> search/sessions

Of the 544 searches, 45 were followed by at least one <edit> (equivalent to the number of sessions in which <edit> was done). This function was used 72 times in this sample batch. Hence, 117 searches (including the originals) were examined to provide an insight into the <edit>ed searches.

After disregarding some of the searches (logout, timeout etc.), 58 were left as potential practical <edit> searches, as per Table 6.2. Analysing these syntactically, 6 should have been <new> searches considering the effort in deleting characters from the previous query and typing in new ones. Thus, there remains a total of 52 searches which could also potentially be genuine <edit> searches. Infact, 45 were genuine <edit> searches in the semantic sense.

Although it may be rare, it is possible to <edit> a search and change its semantics so as to conceptually make it a <new> search. Typically, this is done by deleting a search and typing an entire new one.

Summary Points of <New> and <Edit> Search Analysis

The above results can be summarised as follows:

<New> Searches

The <new> facility is used quite a lot but not for genuinely new searches. The tendency is to use it for syntactic (typing) convenience. It could be argued that this was expected. However, it is interesting to note that users do not seem to mix different topics of interest in one session. For example, of all the times <new> was invoked (including the later disregarded searches), only about a fourth (37/165) turned out to be genuine <new> searches.

Looking at sessions:

Although it seems that a fourth of the sessions have at least one <new> search in them, in "real" terms (after the logouts, timeouts etc. have been disregarded) this figure is reduced to 21% of all the sessions which have at least one <new> search. Furthermore, looking at the context of these searches it would appear that less than 8% of all the sessions have genuine <new> searches.

Looking at searches:

These figures are slightly higher if we look at the searches rather than the sessions. For example, 30% of all the searches are <new> searches. However, in "real" terms (disregarding logouts, timeouts etc.) this is about 24% of all the searches. After analysing the context of these searches only about 7% (37/544) of all the searches are genuine <new> searches. Based on the syntactic analysis, 30% of all those which could have been genuine ('potentially genuine searches') were genuine <new> searches.

<Edit> Searches

The <edit> facility is used quite a lot and very seldomly for (genuinely) new searches. This is quite consistent with the design of the facility. Very few of the original searches were changed completely, generally they were modified slightly. About 62% of all the times <edit> was invoked (even if we include those disregarded later) were genuine <edit> searches.

Looking at sessions:

A parallel analysis, to the above, shows that only 15% of the sessions have at least one <edit> search in them, in "real" terms (considering 9 sessions for ommission due to logouts, timeouts etc.) this figure is slightly reduced to 12%. Furthermore, 11% of all the sessions have genuine <edit> searches.

Looking at searches:

These figures are slightly lower if we look at the searches rather than the sessions. For example, 13% (72/544) of all the searches are <edit> searches (i.e. are edits on previous queries). However, in "real" terms (disregarding logouts, timeouts etc.) this is about 11% of all the searches. After analysing the context of these searches only about 8% of all the searches or about 86% of the potential genuine <edit> searches are in "real" terms genuine <edit> searches.

The Relation Between Syntactic and Semantic Modifications

This section takes the syntactic and semantic analysis done in the previous section further. The aim here is to identify the nature and effect of syntactic modifications to a query and the resulting semantic change. Each modified query is compared with the preceding query (although this usually is the original query it can also be a modified one). This analysis was done on the same batch of 544 queries for the searches modified both with the <new> and <edit> functions. The total number of searches for the <new> queries is 133 and 58 for the <edit>s, as per table 6.2 (with logouts, timeouts etc. disregarded).

The analysis was done in three stages: syntactic; semantic and the mapping between the two.

Stage I

In the first stage, each modified query was looked at purely from the syntactic point of view and placed into a category + or - where

+: Means any additions (new words) to the immediately preceding query i.e. if the query is lengthened (in letters not context) in any way. The query is replaced with the same terms plus some new ones.

-: Means any deletions (extracting words) to the immediately preceding query i.e. if the query is shortened (in letters not context) in any way.

Misc.: This miscellaneous category includes queries which have not changed at all (in relation to the immediately preceding query, perhaps just changed the last term or that it has changed totally.

The table 6.3 shows the distribution of the query modifications into these three categories. For example, 49% (35/72) of the <Edit>ed searches consisted of adding words/letters to the preceding query.

Syntactic Changes	Edit Searches	New Searches	Total
+	35	17	52
-	3	11	14
Misc.	20	105	125
Total	58	133	191

Table 6.3: <Edit> and <New> Searches for the three categories of syntactic changes.

Stage II

Secondly, the same queries were then analysed with respect to their semantic content, with the author's subjective judgement. Again the same category representations were used but this time to represent changes in the coverage of the query. Here,

+: Means that the query broadens (contextually) the scope of its immediately preceding query.

-: Means that the query narrows (contextually) the scope of its immediately preceding query.

Misc.: This includes all other cases. For example, the query could be unrelated to the immediately preceding query, not have changed its scope (i.e. neither narrowed nor broadened it) or indeed is a totally new one.

Table 6.4 below shows that 44% of the queries modified using the <edit> option narrowed the query.

Syntactic Changes	Edit Searches	New Searches	Total
+	7	11	18
-	32	26	58
Misc.	19	96	115
Total	58	133	191

Table 6.4: <Edit> and <New> Searches for the three categories of semantic changes.

Stage III

The results of the above two analysis were mapped against each other in order to show how changing a query syntactically affects whether it is narrowed or broadened (ie. its 'semantics'). Thus, following are the meanings for the various possible mappings:

+ → + Means a syntactic addition "broadened" the query.
+ → - Means a syntactic addition "narrowed" the query.
- → + Means a syntactic deletion "broadened" the query.
- → - Means a syntactic deletion "narrowed" the query.

The mappings below involve the miscellaneous categories quoted above. For brevity, they have been referred to as category '0'. They have been accounted for completeness but do not carry much meaning with respect to this analysis.

+ → 0 Means a syntactic addition resulted in a contextually miscellaneous state.
- → 0 Means a syntactic deletion resulted in a contextually miscellaneous state.

0 → + Means a syntactic change belonging to the miscellaneous category resulted in "broadening" the query.

0 → - Means a syntactic change belonging to the miscellaneous category resulted in "narrowing" the query.

0 → 0 Means a syntactic change belonging to the miscellaneous category resulted was also in the miscellaneous category after the semantic evaluation.

Table 6.5 below shows the nine mappings described above and the percentages of <edit> and <new> searches that fit into these categories. The table shows that 53% of the <edit> searches comprised of additions of words/letters (to a query) which resulted in narrowing the search.

Syntactic → Semantic Mapping	Edit Searches	New Searches
+ → +	1	0
+ → -	31	16
- → +	3	8
- → -	0	0
+ → 0	3	1
- → 0	0	3
0 → +	3	3
0 → -	1	10
0 → 0	16	92

Table 6.5: Percentage of <Edit>ed and <New> Searches for the mappings made between the syntactic changes in a query and any subsequent semantic change.

Summary Points of Syntactic/Semantic Analysis Results of Query Modification

The results in the previous section can be analysed as follows:

• Between the three categories of "+", "-" and "0" (Miscellaneous category) 60% of the <edit>ed searches syntactically were additions to the previous queries. There were hardly any (5%) deletions of words.

• Only 12% of all the <edit>s actually result in the broadening of the query while 55% result in the query being narrowed semantically/contextually.

• Looking at the mappings, adding terms is more likely to mean narrowing the query - as one might have guessed and the results show. When the syntactic to semantic links/mappings of <edit>ed queries were analysed, it would appear that 53% of all the <edit>s consist of term/word additions resulting in narrowing the query (+ → -), making it more specific. In contrast, no word/term deletions result in the query to be narrowed (- → -).

• 'Syntactically', 60% of the <edit>s were "+" and 'semantically' 55% of the <edit>s were "-", a similar value of 53% of the <edit>s reflect a " + → - " mapping. Similar percentage consistencies also seem to hold for the other categories and mappings.

• For the <new> queries, syntactically 13% turned out to be word additions ("+") to the preceding query (whether it be another new query or the original), 8% were deletions ("-") and a larger number (79%) were in the Miscellaneous category. One could at this point argue that this is expected in <new> queries and that they ought not be additions or deletions of preceding queries and that they should, therefore, also be syntactically "new". However, please see analyses below for further explanation of this figure.

• Semantically 20% of the <new> searches narrowed ("-") their preceding queries, while only half as many (8%) broadened ("+") them. The last category (Miscellaneous) accounted for most (72%) of the <new> searches. Like the <edit>s the "+" and "-" category percentages seem to be somewhat reversed when semantic analysis is done.

Looking at query amendments and how they follow on from the original query the evidence would appear to be consistent with the previous analyses, particularly relating to the content of frequent user queries. These are in line with the hypothesis (Section 4.4, hypothesis 2) that there is a 'contextual' link between a user's queries.

6.5 Deficiencies and Possible Enhancements of Okapi

This section presents some proposals and methods derived from the various analyses in Section 6.4. A subset of these were eventually considered feasible within the scope and context of this work, they are expanded on in Section 6.6.

Proposals

1. Adding to or updating the GSL.
2. Improving document ranking particularly when there are a large number of documents with the same score (*weight-block* problem).
3. Re-running old user queries. When queries and functions are frequently performed by a user, some feedback relating to this could be given upon the next update of the database.
4. Prompting users for expansion of abbreviations used.
5. Performing specific item searches explicitly through exact match techniques.

Methods

1. Analysing functions performed on words input in order to identify forms in queries. (e.g. to see whether certain categories of GSL terms are used more frequently than others).
2. Identifying 'subject areas' (or 'contexts').[72]
3. Identifying any change in these 'subject areas' or 'contexts'.
4. Identifying different levels and times of 'help' for users.
5. Identifying when to expand the query automatically.
6. Identifying user query formation strategies. For example, tracing user queries to see whether they are broadening or narrowing them. This in a sense depends on whether there is a thesaurus already incorporated in the IRS which will provide information about the links between its terms (such as narrow/broad term).
7. Identifying frequently used queries. For example, the queries could be kept in an optional list for subsequent retrieval and re-running; this could be done automatically and the result mailed to the user (as per text routing at intervals). Alternatively, the user could be prompted with a reminder of these frequent queries when logging on.
8. Showing the user a list of words to choose from. If an abbreviation/phrase (e.g. "ai") is used in a query and the initial letters of the words in a subsequent or previous query match exactly, then these words could be shown to the user to verify what they mean with the abbreviation.
9. Retrieving documents when a query string matches the title of a reference exactly. In this case, the reference could be put near the top of the list if it happens to be elsewhere in the descending weighted order of references (E.g. Appendix C: Assessment of Frequent Users' Logs, logs NET-B/9, NET-A/13, NET-L/10).

6.6 Enhancements Involving Learning in Okapi

So far, this chapter has included a description of the bibliographic IRS, Okapi, used to evaluate some of the concepts in this thesis. It also described some weaknesses in the current system and possible enhancements to it. The following three enhancements involving 'learning' in Okapi have arisen from those mentioned in Section 6.5.

These are connected with the *Weight-Block* problem. Although on the face of it the Okapi *Weight-Block* is not a *context* problem, it may be resolved by making use of a user's *context.*

The applications considered for Okapi come under the following three headings:

1) Automatic phrase identification
2) Dealing with homonyms
3) Topic/Subject linkage and identification

[72]The definitions of these were discussed previously in Chapter 2.

For each of the above applications, the learning characteristics, possible desired outcomes, the process, current information available, information required, the testing of the application and the associated problems with the application will now be discussed.

Regarding the 'learning' characteristics, machine learning applications can be classified in various ways. Chapter 3 contained some ways of categorising the learning algorithms according to the techniques involved and reasoning behind them. As mentioned earlier, one aspect is whether they improve system *speed, scope* or *quality* (Section 3.2.2). A characteristic of IR effectiveness criteria is that they can involve improvements in speed and other times improvements in system scope or quality. As far as the outcomes below are concerned, none will directly be (from the user's point of view) efficiency.

6.6.1 Phrase Identification

This involves identifying when a group of words form a meaning of their own and are not independent units. Examples of phrases include "human-computer interaction", "user modeling", "information retrieval", "artificial intelligence" and "knowledge based systems". Phrase identification has been investigated by others in various ways. These include automatic generation of literature abstracts (Paice, 1990) and the use of phrases in document indexing (Jones, et. al., 1990) where repeated phrases are located and ranked based on the statistical information about them.

Phrase identification can also be done using the information gained through relevance feedback and the original queries.

Learning Characteristics

Users sometimes have to work their way through what might be irrelevant information because the system is unable to identify phrases in the original query. Thus, this application involves improving system *quality*. Less time would be wasted if less irrelevant information was presented to the user. Obviously, the notion of what is relevant and what is not relevant is a much larger issue, but in this context the possibility of reducing *false drops* through knowledge of phrases is what is meant here.

False drops can occur when a user is presented with records that syntactically satisfy/match the query but are in the "wrong" context.

> E.g: A query with the phrase "user modeling" will list references concerning human-computer interaction as well as those about flight simulation. In the case of the latter, the words might not be adjacent to each other - somewhere in the abstract there might be sentences such as "the simulation models.....the user (pilot) then....".

The context in this case is considerably different to that of HCI.

Output Scenarios (Outcomes) -

What Can Be Done With The Learnt Information

The system can

• Suggest phrases "learnt", to the system administrator, as possible additions to the current "look-

up" table or a future thesaurus[73] term.

• Suggest the phrases identified to the user so that they may either be incorporated automatically in the query or a new search done by the user.

• Automatically incorporate the terms as being a unit in the search (i.e. without waiting for user verification/feedback).

The Learning Process

Following is a step-by-step description of the retrieval process with learning techniques.

1. The user types in the search statement (query).
2. The system then identifies if there are any phrases in the query (through learning patterns in terms).
3. The system performs the necessary functions in order to achieve one or a combination of the outcomes listed in the previous section.
4. The learner now has an additional case to learn from and incorporate if appropriate.

Current Information Available

The type of information that is available for the above process is as follows:

• Verbatim records of all queries entered by users in "natural language".

• Lists of the term stems generated from the queries and those terms added after query expansion using relevance feedback.

• Lists of existing phrases in the GSL (go-see-list or look-up table) for each database.[74]

• Existing statistics on word collocations.

• Details of the current database the user is working on along with those used previously. This would help identify phrases that are likely to be useful in certain contexts amongst particular user groups.

Typical Information Necessary

• Collocation information for queries in Okapi. Relating collocation information to size of queries with tables, graphs etc.

• For every time there is any relevance feedback, there is no connection at the moment between previous/existing terms and new terms. There are no explicit 'formal' links between terms that

---

[73] At the time of this experiment, the Okapi system did not have a thesaurus incorporated.

[74] Likewise, there would be a common base should an existing machine readable thesaurus be incorporated into Okapi.

have been used and those that are synonymous. (As synonyms in the GSL are conflated, there are some links albeit of a rather simple-minded kind).

Testing
Following are some tests which may help establish the effectiveness of phrase identification:

• Possibly manually going through and identifying some phrases in the recorded user queries and calculating how many iterations of relevance feedback was necessary in order to eliminate the (or a percentage of the) irrelevant data.

• Doing some exercises to see how long it would take to identify the phrases and then incorporate them manually in the GSL or thesaurus.

• A combination of the above two tests in order to establish whether the learner does actually speed up the retrieval process concerned or provide an improved document list.

Problems
Having discussed the potential benefits of phrase identification, following are some problems in applying this:

• There must be sufficient instances of uses of phrases in order to make it worthwhile to learn about them.

• A possible counter argument to phrase identification would be that it does not make much difference in retrieval. However, this could also be related to the statistical methods used to measure such differences.

6.6.2 Dealing with Homonyms

A homonym is when the same word is used to denote different things.

E.g: The term "blind" could have the meaning of a disability or a venetian blind. Through the previous terms used by the user or a particular group of users it might be possible to estimate which context is more likely. Though most ambiguities which cause real problems are much more subtle than this.

Sense disambiguations are complex problems. However, they are not a great problem in IR. Nevertheless, the application here would involve identifying which context the user is likely to require. Having identified the context, the user's search could be directed in a certain path which would eliminate time/effort wasted due to false drops.

Learning Characteristics
Like the previous application, this one may also help with "false drops" and can result in improving system *quality*. If homonyms are identified quickly, retrieal time may also be reduced thereby possibly increasing user satisfaction.

Output Scenarios (Outcomes) -

What Can Be Done With The Learnt Information

• Build vocabulary profiles in order to help identify likely contexts. It could be dangerous to presume that the user will always be asking for information in the same context so it might be necessary to ask for feedback.

• Suggest more words/terms to the user that would fit the intended context.

The Process

Good training sets would be necessary if any identification of homonyms is to be possible. This might involve setting up an experimental situation (the environment), for example, where certain sense ambiguities are bound to happen and must be resolved. Then the previously described stages encompassing an incremental learner would be followed.

1. The user types in the search statement (query).
2. The system then identifies if there are any possible homonyms amongst the terms used and those learnt previously.
3. The system performs the necessary functions in order to achieve one or a combination of the outcomes in the output scenario options.
4. The learner now has an additional case to learn from and incorporate if appropriate.

Current Information Available

• User registration information containing details of their status and course. This would help establish likely 'contexts' for that user and links with other users on similar courses.

• Previous terms used by user.

Typical Information Necessary

• More information on users interests, either through voluntary user feedback comments or explicit questionnaires.

• A machine readable thesaurus may help identify or create 'context' links for users.

Testing

Following are some ways in which the testing of the application and identification of homonyms for document retrieval can be done:

• Comparing situations where the learner is used with those where only relevance feedback is used to see if homonyms are clarified in the IR process or if they make a difference in the number of relevant records a user identifies from the document list.

• Asking for user feedback on how useful they found the system with the homonym learning option "on".

Problems

• Through some small scale tests it seems that relevance feedback itself performs very well without identifying the homonyms. Usually within the first two iterations of the feedback process almost all the irrelevant references were discarded. Thus, it would appear that there is less scope for improvement than might have initially been envisaged.

• Out of the three applications discussed in this section, this one looks like the more difficult in terms of gathering a reasonable quantity of useful data. In full, the problem needs natural language processing.

6.6.3 Subject/Context Linkage and Identification

The terms *subject* and, in particular, *context* refer to the ideas discussed in more detail in Chapter 2.

There seem to be contextual links between individual user's searches, as described previously in Section 6.4. Contrary to predominant assumptions, user searches do not seem to be as unrelated and independent of each other as previously envisaged.[75] Additionally, there are times when a group of users have common information requirements. In such situations a considerable number of terms may overlap producing similar sets of references. However, due to the relevance feedback from the user, each reference set does still tend to have an associated unique path and set of terms. By identifying the similarities and differences between these terms and (term) paths, some 'contextual' links or similarities might be identified.

An example of subject/context linkage is as follows: Within the space of a few days several of the network users from the Dept. of Computer Science wanted information relating to "wafer scale integration". Some indicated links to "parallel processing" in their queries. Others added terms such as "wafer technology". Not all users, however, seemed necessarily aware of the relationship between "wafer scale integration", "wafer technology", and "parallel processing", for example. Hence, this would seem to be a good situation for learning to take place. The information about term relationships could be shared amongst a group of users.

Learning Characteristics

This application falls more under the category of "learning in order to improve system scope and quallity". With this it is hoped that the levels of user satisfactions will increase.

Output Scenarios (Outcomes) -
What Can Be Done With The Learnt Information

Possible output scenarios from the learning process are:

• Suggest other possible useful terms to the user, by following the various term relationships in the queries and relevant documents.

---

[75] This was the case for the academic retrieval environment Okapi was implemented in.

• Distinguishing between what is relevant in one context and not in another.

• Identifying any similarities between different users' queries, particularly those working in the similar fields.

• Avoiding repeating searches by learning (rote) which records were retrieved in through which terms.

• Improving a thesaurus. An approach could be to analyse the links between the terms, identifying narrower, broader, related and synonymous ones. If a thesaurus does not already exist perhaps a mini-thesaurus could be created for groups of users or the existing GSL expanded to include such information.[76]

The Process

The iterative process incorporating the learner is as follows:

1. The user types in the search statement (query).
2. The system then identifies if there are any existing links/relationships amongst the terms used and those learnt previously.
3. The system performs the necessary functions in order to achieve one or a combination of the outcomes listed in the previous section.
4. The learner now has an additional case to learn from and incorporate if appropriate.

Current Information Available

• The current and previously used (by user) databases. This would help in relating to *contexts* and identifying user groups.

• Records of all terms used in user queries.

• Records of users in the same research group (if specified).

Problems

The possible problems with this application include:

• Situations where the groupings discussed do not occur very often in which case learning such term information may not have much use. On the other hand, if (as suspected) it turns out that there are peak times when these groupings occur then it is important the learner adapts quickly and the knowledge be applied as soon as possible to be used in subsequent searches. This situation seems to occur when courseworks are given and at the initial stages of research initiatives/projects.

---

[76]It was not original intention to make the GSL a complex file structure, but more like an intermediate/temporary solution.

• Deciding what kind of links (contextual) would be useful is not very straightforward nor is it easy to define boundaries for user 'subject areas' or 'contexts'.

Combining applications

The first two applications, involving phrase identification and homonyms, can be combined with the third concerning *context* linkage in the sense that a *context learner* may well solve problems which applications 1) and 2) were designed to solve.

6.6.4 Dealing with Weight-Blocks

Weight-blocks were defined and discussed previously (Sections 4.4, 5.2). The Okapi *Weight-Block* problem on the face of it is not a *context* problem but like the homonym situation it can be resolved in this way. It is possible that the weight-block problem may be rather peculiar to present Okapi. It is also possible that an alternative method such as within-document term-frequency component may get rid of most of a weight-block. Nevertheless, it is a current problem and *context* can help distinguish between document records in the list presented to a user. The ways in which the document ordering can be enhanced was also discussed previously (Section 5.2).

On average users type in two or three words (2.7) when representing their query (Walker and Hancock Beaulieu, 1991). In some databases, especially Inspec, this is not usually a precise enough description. Users are shown the results of their search in the form of documents records in descending document weight/score order. In such situations, however, it is not unusual for many documents to have the same weight for that query, thereby forming *weight-blocks*. This forces users to examine several screenfulls (about 30 in extreme cases) of references which are apparently in nothing other than alphabetical order (as the evidence presented in Section 6.4 would suggest). Users may never reach a system feedback point, between the document records in the list, such as "The rest of the references may match your search less well" before either giving up or realising that they need to provide a more specific query.

Items in a weight-block are items which the query (as originally given) can not distinguish between. As far as any user criterion is concerned it is in random order. Therefore, if the context has no relation to the query then the resulting document list should be just another random order. Date-order might be a user criterion (it may be something they like) in which case reordering the list might possibly make it worse than bare Okapi.

It is difficult to overcome the weight-block problem in the very first search performed by the user. However, once the user has used the system a few times, there is the potential to re-order the documents in the same weight-block in a way so as to better suit the user's information need in light of the *contextual* information the system would have acquired by then.

Chapter 7

Application of the Context Learner to Okapi

This chapter describes the implementation of the *Context Learner (CL)* in the Okapi IRS, its evaluation and results. Existing evaluation methods in IR are referred to where necessary and the two further experiments performed are detailed.

7.1 Implementing the Context Learner

This section describes the inputs, outputs in the applied CL and the way in which it is integrated with the Okapi IRS. Also discussed are the variations in the CL, their implementation and the associated problems.

7.1.1 The Process for Gathering Input to the Learner

The Okapi system provides logs of user online sessions (Appendix B: Okapi Transaction Logs). From these it is possible to extract various details relating to user queries. As the logs get updated after each query they can also be used for a *live* online learning system as well.[77]

The inputs to the CL were also discussed in Section 5.3. Here, however, there is an emphasis on the more practical aspects of the details required for the *learner*. Thus, the following are used for the *learner:*

- the stemmed query terms
- the documents chosen to be relevant by the user
- the index terms from the relevant documents
- the previous context terms

Of these, the first three are extracted from the logs. Section 7.1.3 describes how all four components are integrated with the IR process.

[77]It is also possible to do the learning offline in *batch* mode, as addressed later in Section 7.1.4.

7.1.2 The Output from the Learner

After each query with relevance feedback, the *context learner* updates the user's *context*. The *context* consists of two sets of terms: the terms to be *Used (U)* for the next iteration and those on temporarily on *hold (H)* (Section 5.6). The *U* terms affect the document ordering. They are used to break-up the *weight-blocks* i.e. reorder the documents within *weight-blocks* (Section 5.2).

Once a *context* exists for a particular user, the *U* terms are used in the next query. They are added to the user's query terms and like the query terms are given weights. However, the purpose of these terms is to reorder documents *within a weight-block* only and not to reorder the whole document list or to produce a different list altogether. Therefore, the terms are given very low weights. This is a practical way of ensuring that documents indexed by these terms do not move from one (old) *weight-block* to another. Each document has an associated *document score* which now is derived by the sum of the weights of the query and *U* terms that index it. The scores of the documents in a *weight-block*, by definition are the same. After employing this method, the scores do not vary greatly from their original values but are enough to break up ties (in the scores) and enable a reordering of the documents according to the user's *context*.[78]

7.1.3 Integrating the Learner with Okapi

The way in which an *incremental context learner* can be incorporated into an IRS such as Okapi was discussed earlier (Section 2.1.4). Briefly, the process is as follows:

- User types in query.
- Query is parsed.
- System displays document list.
- User gives relevance feedback on documents.
- The feedback is also used by the *context learner*.[79]
- The *context* is ready for use should the user type in another query.

Taking the CL as a focal point, the processes can be viewed in the following way:

- Gathering and parsing the input data for the *context* learner.
- The *learning* process or adaptation algorithm itself.
- Restructuring the output document list before showing it to the user.

The merged stages are shown in the Fig. 7.1.

The Okapi IRS is written in the C-Programming Language. Figure 7.1 illustrates that the Okapi system produces log files for each online session containing the details of the queries input, the documents looked at and those chosen to be relevant (e.g. Appendix B: Okapi Transaction Logs). This is used as the raw material for the CL, which is written in Prolog. This facilitates the

[78]There is another way of reordering the documents within a *weight-block*. If details of *document rank positions* and *document scores* were kept for each query, then the documents within the same *weight-block* could be re-sorted according to the number of *U* terms that index them.

[79]This is in addition to the query expansion provided by the IRS.

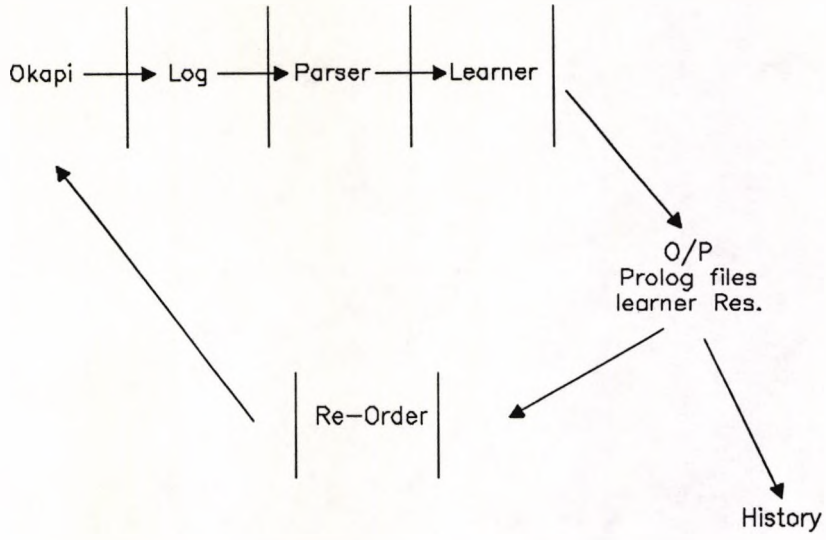


Fig.7.1: The components involved in applying the context learner to the Okapi IRS.

transition from specification to code.[80] The content of the log files are parsed (after each query is completed) to acquire the relevant data for the CL in the required form. Parsing programs were written in Awk/Perl which is suitable for pattern/string matching on data file lines. After the *learner* has completed its task the output is kept so that the results can be used by the system to modify the order of documents (as described in Section 7.1.2) presented to the user. The CL's results are in the form of prolog generated files containing *context* term information (Context Files). This is used both for re-orderding the documents and for a historical record (History Files) of the *context* terms acquired after each *learning iteration.*

7.1.4 Possible Variants

Chapter 5 described the CL modules and their variations in detail. Briefly, the *context learner* consists of four components:

- **Module *A***: Create set *R* of relevant documents.
- **Module *B***: Create *Active* and *Passive* sets of terms *(A,P)*.
- **Module *C***: Merge these newly acquired terms with the *old context*, which consists of terms to *used* and those put on *hold (U, H)*.
- **Module *D***: Use the past performance of a term to determine its role in the current *context.*

Variations within these modules, as discussed in Section 5.6, depend on:

- What should constitute the set R (documents from the last query only or documents from all queries).
- Whether duplicate document records should be treated removed or not.
- How to deal with the number of relevant documents a term has been found in (involving issues such as *minimal cover*, term weights and any limits in the number of terms in corresponding sets).

[80] CL versions can be obtained through The Department of Information Science, City University.

- How to merge newly acquired terms *(A,P)* with the old *context (U, H)* terms and whether any limits on the number of terms in any of these sets apply.

Various tests incorporating these differences can be constructed. The rationale behind such a series of tests was described in Section 5.6.4. All of these were implemented as discussed in the next section. However, during development a few other versions were tested and later abandoned. In some cases this was because the resulting sets of terms over several iterations were identical to other versions. In other cases, initial tests on alternative parameters did not indicate that there would be significant differences in the results. Therefore, given the current experimental constraints including limited time availability of the selected user groups, these particular tests were not furthered.

The tests have been referred to as T1, T2 and so on. However, the implementations of these tests (the different *context learners*) will be referred to as L1, L2 and so on, in this chapter.

7.1.5 Implementing the Variations

Although a variety of languages (Awk/Perl, C, Prolog, Shell programming) were used for the overall process as described above in Section 7.1.4, this was not necessarily done to make any statement about a particular language being 'better' than any other. The decision to choose a language was generally based on what might be efficient, plausible and possible for this purpose, although it was also governed by the availability of resources and certain technical constraints.

Where there are structured definitions such as those made for the CL modules and their variations, Prolog is more suited in coding as the resulting code has a closer resemblence to the original definitions. However, any amendments to the Okapi system for gathering the input data or reordering the list of documents were done in the C programming language as the IRS was written in it.

Initially, the *context learning* Okapi system was implemented to work interactively online or *live*. In this system, *contexts* were derived in real-time as the users performed their queries. However, due to unforeseen difficulties with both user registration and database availability during development, a *batch* system was written to reuse previously performed queries. For consistency, this system was used in both stages of evaluation described later in this chapter.

There are some differences between running the CL *live* and in *batch* mode. For example, in the *live* system, parallel communication processes (such as pipes in Unix) need to be set up. Also, in the *live* system there are two important points regarding the integration of the CL. Firstly, the method of deciding when to *learn* has to be established. Secondly, a decision has to be made as to when to use it. In a *live* environment, the system has to rely on the user for some indication as to the end of a query such as performing a new query, editing the old one or exiting the system. It is important to have a delimitor of a query before the start of the next one.[81] Any (positive) relevance feedback obtained from the user during that query is then input

[81] For the Okapi IRS, this was when there was a New or Edit query function initiated.

to the CL. Naturally, within the *batch* environment identification of the start and end of queries is predefined.[82]

In both systems, the input data to the CL, the data generated in the process of context learning, and the output data were kept separately. The data consisted of

- User session logs.
- Query datafiles. These are the queries after the logs have been parsed in the format ready for the CL.
- Context files and History files.
- Document (document identification number, *doc-id*) lists for queries after the CL has been put to use.
- User relevance judgements for documents (*doc-ids*), used in evaluation (as addressed later in this chapter).
- Precision values for the document lists (also discussed later in this chapter with results).

The above were been kept for all CL versions and used during development and for checking consistency checking.

7.1.6 Problems in Implementation

There are difficulties in coordinating any new version of an experimental system with the original or other versions already implemented such as the case of the Okapi IRS. Part of the difficulties are due to different projects on the system running in parallel and not always with the same targets and requirements. Other difficulties arise due to insufficient hardware resources and the management and coordination of having several CL versions. They are listed as follows:

- Each Okapi IRS version, usually part of a different project, has its own requirements, specification, targets and deadlines whilst also having to share some common code with the others. Such systems tend to be constantly amended particularly during development. It is not uncommon for changes relating to one project or version to have a resultant effect on the others - not always planned or desirable.

- Databases also get updated during this time and new indexes are generated. This results in compatibility problems between the programs written for processing the old indexes and those written for the new ones.

- There are hardware resource constraints, in particular the available disc space. Therefore, it is often not possible to keep several versions of the databases and their indexes for different purposes and comprises have to be made. Additionally, there are a large number user logs which put further constraints on the disc space if they are to be readily accessible (which is necessary for the CL to perform its task). Depending on the number and size of userlogs, the

[82]The start of the next query means the end of the previous one.

processing (CPU power) time can also be a problem, particularly in a *live* version of the system.

• A compromise has to be made between the amount of different parameters encoded in one CL version and the ease with which it can be altered and tested. It is more flexible to have several different CL programs, as more complex tests can be created using the simple components, as opposed to trying to build all possible permutations into one CL.

• Similarly, a balance has to be made between the ease with which a user log can be read (by humans) and the ease with which it can be processed by the computer.

7.2 Evaluation in Information Retrieval

Evaluation in ML, as discussed previously in Chapter 3, tends to concentrate on the 'correctness', functionality and efficiency of the particular technique employed. There is an emphasis on demonstrating that the technique or program works and that it is feasible. From an IR point of view, however, this is not really sufficient. In fact, it could be argued that this is not satisfactory from an ML point of view either. The ML community tend to bemoan a general lack of 'real world' data for applying and evaluating their techniques. If such data is to be used then evaluating the techniques an their usefulness in the context of their 'real world' applications becomes more important.

Thus, in investigating the applicability of any learning technique (whether it be in *symbolic* or *statistical* form) in IR, we need to go beyond simply demonstrating a system application and should test for its 'usefulness', bearing in mind the real users for which the systems are intended for.

The range of existing evaluation methods in IR are discussed with a view to describing and justifying the eventual method of evaluation chosen. The evaluation work in this thesis falls into three categories.

• The first, already discussed in Section 6.4, was in the form of a fact-finding pilot study to establish the potential uses of ML in IR and which existing problems in particular (in the Okapi system) could be solved with this general approach.

• The second stage of evaluation (as discussed in this chapter) was performed to establish which of the candidate CL algorithms was likely to perform 'better'. Thus, a comparison was made between the CL versions.

• The third stage of evaluation (also discussed in this chapter) was performed to observe whether any of the candidate (remaining) CL algorithms were an improvement on the existing 'plain' system.

From a general point of *context*, the purpose was to see whether a *context* formed from previous queries was in any way usefully related to the present query and therefore would help with

document ranking. Two inferences were required to be drawn here. Firstly, whether the *context* formed from the past queries is *usefully related* to the present one. Secondly, whether a *context* formed from one query (or all past queries) can *usefully* be applied to the next query.

The first of these inferences is a restatement of the weak hypothesis (Section 5.1). It is also possible that the contexts turn out to be equal (related in a stronger way), in which case the strong hypothesis would be confirmed. The second inference relates to the application of context to improve document ranking. The strong hypothesis asserts this more strongly than the weak one.

The purpose of this section is to provide the background for the last two experiments performed. In doing so, the nature of evaluation in IR, the methods available along with some of their strengths and weaknesses are discussed.

7.2.1 The Nature of Evaluation in Information Retrieval

Evaluation has attributes such as experiment, test and investigation. The distinction between these is not very clear in operational environments but generally speaking, investigations are primarily descriptive. Experiments, however, tend to be explanatory and have a high degree of control over factors being studied. The purpose, with experiments, is to test a hypothesis as objectively as possible (scientific method) with various measurements and quantitative evaluation. An experiment therefore consists of several tests to make these measurements (Sparck Jones, 1981a). Hence, the module combinations in Section 5.6 were referred to as T1, T2 and so on. To test is not to evaluate but they reveal to what extent an IRS performs in a certain way. Therefore, we should test the performance necessary for the aim of the system and do so to find how well it achieves this aim.

Before going on to discuss the ways of evaluation in IR, however, it is necessary to be clear about what exactly is being evaluated. In the field of IR, *information systems* can encompass system components (e.g. people, equipment, software), information processes (e.g. formulation query, printing results), products and services (e.g. on-line systems, publishing), information functions (e.g. literature searching, document retrieval), comprehensive information systems (e.g. library, information consultants) and the information systems environment (the organisation/population served) (Roderer, 1982). Bawden (1990) simplifies these and groups them under the following three categories:

- Information source

  A single source of information e.g. journal, book, on-line database, cd-rom.
- Information systems

  Something (a system) which gives access to information such as an index, a document/information retrieval system.
- Information service

  Consists of information resources and systems but also tends to include a user point-of-view i.e. the whole operations of a library/information unit.

In all of these categories there is a wide range of things to evaluate. The work of this thesis focuses on the evaluation of an *information system* or more precisely an *information retrieval system* (Okapi). Evaluation of *information systems* can be done for the provision of better products and services or the justification of existing services or the improved understanding of systems (Bawden, 1990). However, specifically which aspects of these are evaluated and the method chosen tends to depend on the purpose of the experiment and point of view taken. Laboratory experiments, such as using test collections, are useful for developing and evaluating new models to a certain extent but they can not be regarded as conclusive evidence of their success or failure. For this, eventually, it is necessary to have a user-oriented approach as by the nature of the information system it is the users who will utilise them. In this aspect, IR is more like a social system.

The evaluation work in this thesis was originally investigative, in that through the pilot study (Sections 6.3, 6.4), current problems in an operational IRS were identified to see which would seem likely to benefit from an ML approach, and particularly if there was any scope for *context learning* as described previously. The second experiment was more analogous to a laboratory test. Although users were approached for relevance assessments they were used in much the same way as those in the test collections, providing a good test-bed for the comparison, developing and tweaking of different models. Thus, various hypotheses relating to the application of *learning to improve ranking* (by breaking up *weight-blocks*) were tried at this stage. The overall hypothesis is that *context* is useful for an IRS in that it can improve document ranking. However, particular details such as whether it is something that is stable or constant or is continuously changing constitute more particular aspects of this hypothesis. The evaluation approach is empirical so far. After this, the remaining preferred candidate algorithms were evaluated to see if they did indeed result in an improvement of performance. These improvements are based on user relevance judgements but the way on which these are measured and analysed will be discussed later in the chapter.

Evaluating the performance of an IRS involves considering its efficiency and effectiveness (Salton, 1983). The efficiency of operations, the cost, coverage and currency of a system may be fairly easy to determine but the effectiveness of a system may be more difficult. Moreover, an IRS consists of many different components (the retrieval mechanism, the indexing systems, the user) and although one may be interested in improving the performance of only one of these components (the retrieval system in this case) it is difficult to evaluate them in isolation. Thus, some form of overall system evaluation is required. These include various assumptions about *information need* and *IR seeking behaviour* (also see Chapter 2).

Evaluation of IRSs has taken many forms in the last 30 years, partly due to the advent of new technologies that have enabled the efficient and effective logging of search activities. In that time, it has generated much controversy, with issues ranging from the validity of laboratory and operational experiments, the importance of the user in the evaluation process, to the most effective measures of performance. The following sections provide more detail on some of these.

7.2.2 Relevance

The aim of an IRS system is "to find information items relevant to an information need" (Bartschi 1985). Hence, the concepts of *relevance* and *information need* are the foundations of much of the work in IRS evaluation. However, the pressure to evaluate has forced researchers to deal with these theoretical problems by stating the interpretations used and the assumptions made (Smithson, 1989). The concept of an *information need* was discussed previously (Section 2.1) but *relevance* is discussed further in this section.

The view taken here pertains to document-retrieval systems and so the concept of a user's information need is expressed in the form of "Give me what (document) I need" (Fairthorne, 1965). Whether evaluation is in an operational environment (where real needs of users have to be met) and/or experimental one, some measure for user *satisfaction* needs to be used. Often it becomes necessary to breakdown the aspects of the system and attempt to deal with them individually (Bawden, 1990). Evaluation in an IRS (like for information services and systems in general) can involve measures for cost benefit analysis. These include cost-effectiveness comparisons involving user/system response times and user/system costs for time spent and possibly number of units of items found. Other aspects of evaluation of information services include the coverage and currency of documents (e.g. such as that provided by an SDI service) (Robertson, 1981).

Evaluation in terms of user information needs, requirements and their degree of satisfaction involves the concept of *relevance*. *Relevance* in the context of a document-retrieval system, according to Robertson (1981) corresponds to the question of how well the document matches the user's need. As the need is expressed in the form of a query, it is about the document and the query. Robertson also points out that both the notion of *relevance* and its appropriateness to retrieval tests are the subject of much debate and experiment. However, within the category of subjective responses to system output, *relevance* enables a more formal or 'harder' form of analysis than any other assessment (e.g. It is possible to ask the question as to why the system failed on a particular document). Thus, it is used in the user-oriented evaluation that follows in this chapter.

For evaluation, of this work, when obtaining the users' relevance judgements the documents were deliberately randomised. This was to ensure that the judgements were made outside or independent of the IRS.

The objective view mentioned above focuses on the relationship between a given query and a particular document. Thus, *relevance* is considered as a logical property (measurable by the degree to which a document deals with the subject of the user's information needs) between a pair of textual items. This definition, however, does not take into account the particular state of knowledge of the user who may have seen the item before or might be familiar with it through earlier searches or indeed might possess it already.

A more subjective view is the idea of *pertinence*. It considers the user's state of knowledge at the time of retrieval and the other documents retrieved (or available) that the user already knows about. This notion of relevance depends on the *utility* of the item to the user. Thus, it is a subset

of the stored items which is appropriate to the user's information need at the time of retrieval. A document may be *relevant* (if it deals with the appropriate subject matter) but not *pertinent* if the user is already acquainted with its contents, has retrieved it earlier or if other documents retrieved already cover its content. To help overcome this problem, when requesting relevance judgements, users were asked to make their judgements irrespective of whether they had previously seen or obtained the document. Section 7.4.3 describes this more fully.

Similarly, documents (therefore the terms in them) may be good for query expansion but not necessarily relevant or pertinent from the user's point of view. This situation was often encountered in the Okapi-for-TREC experiments (Robertson et. al., 1993; Robertson et. al to be published) where some documents appeared to be very useful as a basis for query expansion but did not fall within the experts' definitions of what constituted a relevant document.

7.2.3 Measures for Evaluation

This section provides only a brief review of the evaluation measures as this has been done more extensively by others (Bawden, 1990; Sparck Jones, 1981c; Salton and McGill, 1983; Lancaster, 1979; van Rijsbergen, 1979). It includes the reasons for the measures chosen in this work and some indications as to the problems with them. However, these have not been taken further as it is not the purpose of this thesis to focus on evaluation measures in IR.

Two traditional measures at the forefront of IR research are *recall* and *precision*. Both these measures, which relate to *relevance*, were introduced with the Cranfield experiments (Sparck Jones, 1981b) and since then have been the focus of much evaluation work in IR. Behind these measures is the assumption that the search universe may be partitioned into two subsets with respect to a specific query, as Table 7.1 indicates. These are the set of relevant documents *(R)* and non-relevant documents *(N)*. It is also assumed that these judgements are made independently for each document (which ofcourse is not necessarily the case in reality). Within a database, there are documents which are retrieved and those which are not retrieved. Thus, there are documents which are relevant to a specific query and are retrieved and those which are relevant but are not retrieved. *Recall* is defined as the proportion of relevant material retrieved ($a/(a+c)$) and *precision* as the proportion of retrieved material that is relevant ($a/(a+b)$).

	Relevant	Non-Relevant	
Retrieved	a	b	a+b
Not Retrieved	c	d	c+d
	a+c	b+d	a+b+c+d (Total Collection)

Table 7.1: The 2x2 contingency table (Cleverdon, 1967).

For example, let us assume there are 100 relevant items in a collection (or database) and a search produces 70 items of which 60 are relevant. The precision would then be 83% (60/70), but the recall would be 60% (60/100). To evaluate the performance of an IRS, the *recall* ratio can be plotted against the *precision* ratio or *fallout* ratio (Cleverdon, 1967). However, there are differences between set retrieval systems (e.g. Boolean) and ranking systems. In set retrieval systems we only have one value for recall and one for precision. In ranking systems, it is possible to draw precision-recall graphs as several precision and recall values can be obtained (at various cut-off points). Each cut-off point divides the set of documents into retrieved and non-retrieved documents.

There are problems when using precision and recall values to compare between the performance of different systems. Single numbers combining both values have been suggested (Pollock, 1968; Swets, 1963). Swets' formal model, based on statistical theory, allowed performance to be predicted. He defined recall, precision and fallout in probabilistic terms and proposed a single-valued[83] measure of retrieval performance. Reducing the system performance comparison to a single number results in the loss of information which may highlight which system has better recall and which has better precision values (possibly at certain cut-off points). Assigning arbitrary weights to derive a single measure of overall system performance may not necessarily highlight the strengths and weaknesses of the systems.

In this work, a ranking system has been evaluated. The document cut-off points have been fixed for evaluation. For example, 20 documents (for a particular CL version) have been shown/retrieved to the user for relevance judgements. A single measure, precision, is sufficient because it completely specifies the result (at least for a single query).

Many variations of *recall* and *precision* have been suggested, essentially involving the same basic data. These include *fallout*, *generality* and *noise factor* (Lancaster, 1979; Robertson 1969; Tague, 1981; Van Rijsbergen, 1979).

[83]When measures such as those in the 2x2 contingency table are used on their own they are referred to as *single measures* of effectiveness. When combined in pairs, they are referred to as *twin variable measures*, e.g. recall-precision graph. *Composite measures* are also derived from the table but combine two separate measures into a *single-valued* measure (Efthimiadis, 1992).

Nevertheless, by far the most frequently referred to measures (possibly as they are the least problematic to determine) are that of *precision* and *recall*. In an ideal situation, both should be a 100% but in reality there is usually an inverse relationship between these two measures (Sparck Jones, 1981b; Lancaster 1979), as was discovered in the Cranfield experiments.

Precision and recall values provide useful comparisons but they also have their disadvantages. One disadvantage is that they hide the fact that absolute values can be important. For example, a recall of 70% may appear as a significant proportion and may indeed be so if it is 7 out of 10 documents. However, if it is 700 out of 1000 documents, in reality the remaining 300 not retrieved can have a more serious effect on the performance of an IRS. This may be very important for a user whose information need necessitates an exhaustive search (e.g. checking a patent application). The problem is not too dissimilar to that encountered in the fields of medicine and social sciences. In medicine, although ratios may be 'impressive' in statistical terms, even if there is a figure of 99% this may not be 'good enough' in a drug testing environment, for example. The remaining 1% may involve a large number of people and may not be acceptable in legal and social terms.

These measures also have their estimation problems. For example, when no relevant documents exist *recall* can not be defined. Likewise, when no documents are retrieved *precision* is undefined. Another disadvantage is that they do not represent any ranking. In such partial match systems the user can choose to stop searching at any point. Once they stop (cut-off point) the set of documents are divided into retrieved and non-retrieved sets. As experimenters, on the whole, we do not want to tell user where to stop (principle of ranking systems) and therfore want the best documents to be at the top.

In the two evaluation stages described in this chapter, as it is not really possible to fully calculate recall (since we do not know the total number of relevant documents in the database), precision values are used as the primary measure of performance. Tables are made for these values at various fixed cut-off points (rank positions) such as the precision for the top 5, 10, 15, 20 documents (Appendices O and P: Precision Values at Various Cut-off Points - Evaluation 1 and 2). The TREC projects (Harman, 1993) also have such a measure.

With different cut-off points, in the same system, on average we would expect to get an inverse relationship i.e. the larger the cut-off point the lower the *precision*. If on the otherhand, we keep the same cut-off point but compare across different systems one would expect a directly proportional relationship (providing we use absolute number of documents which is the case). A good system should give high *precision*. Generally, the approach here does not favour averaging over various cut-off points as some systems perform better at top cut-off points and some at bottom cut-off points.

There are other problems with cut-off points. For example, at cut-off 5, if there are 7 total relevant in the collection these should be treated differently to if there are 700 total relevant. If on the otherhand there are 3 total relevant in the collection at cut-off 5 then can at best we can get a 60% (3/5) precision which is less than 100%. There are substantial differences betweeen the queries and how strictly or loosely the users interpret relevance and how many relevant

documents might be in a collection. The differences between queries also lead to problems of averaging across queries. Hence, there is good reason not to have one fixed cut-off point.

In precision and recall ratios assigned values for *relevance* must be either 0 or 1. There is no scope for a spectrum of *relevance* and hence no ordering (by the user) of the documents according to degree of satisfaction (Pollock, 1968; Raghavan, 1989). A three-point relevance scale (Saracevic, 1971) has been suggested as follows:

> "A *relevant document* is any document which on the basis of the information it conveys, is considered to be related to the user's question even if the information is outdated or familiar to you.
> A *partially relevant document* is any document which on the basis of the information it conveys, is considered only somewhat or in some part related to your question or to any part of your question.
> A *nonrelevant document* is any document which, on the basis of the information it conveys, is not at all related to your question."

Users' relevance judgements are subjective and we should take this into account when using them in any measures. There may well be a 'grey' area between relevance and non-relevance of documents. The three-point scale described above is one way of addressing this issue, which has been done in this work. However, to calculate the precision, recall values the partially relevant documents in the end have to be marked as relevant or non-relevant. Thus, two sets of precision and recall values could be obtained.

In summary, the evaluation measure chosen in this work involves precision (the two types, depending on how the partially relevant documents are treated), cut-off points and relevance judgements.

7.2.4 Test Collections

A test collection consists of[84]:

- A set of documents.
- A set of queries that can be searched against the documents.
- A set of relevance judgements that state which of the documents are relevant to which of the queries.

The rationale for focusing on a very small subsystem, as Willet (1990) points out, is that it is then possible to carry out extended investigations of the characteristics of the subsystem under study and hence to obtain detailed insights into its utility and applicability under a wide range of experimental conditions.

The Cranfield experiments (Cleverdon, 1962; Cleverdon et. al. 1966), designed to measure the efficiency of indexing languages, had a major influence on the evaluation of IRSs. They emphasised the importance of test collections and their use for comparative evaluation. However, over the years, although researchers have used the same collections there has been a lack of

[84]Cleverdon (1990) also describes some associated variables with these components.

consistency in using the same data and evaluation measures which makes it difficult to compare results across systems. More recently, the TREC projects (Harman, 1990) have addressed these issues.

Although more recent test collections are considerably larger in size, on the whole they have not tended to be realistically sized. The problem with this is that evaluation using small collections may not reflect performance in larger collections. They may also not reflect the situation in a real-world IR environment. For example, test collections tend to contain bibliographic information, while commercial systems increasingly offer full-text newspaper, journal articles, prices, figures and reports.

The Cranfield collection, in the 60s, contained 1400 documents and 225 queries. Since then other, considerably larger (e.g. 30-40 Megabytes) collections have been built. Examples include the ACM test collections on Virginia Disc, Reuters, TREC. They vary in length from single line bulletins to multipage articles. They include formatted and unformatted tables of numeric data as well as text and their content includes not only reports of recent events, but also long feature stories, quotes on market prices and corporate earnings reports.

A test collection would have been useful at the development stage of the CL and its variations, although at some point it would have still been necessary to make some user-oriented evaluation (see Section 7.2.5). The problems mentioned above are contributory reasons for not using test collections. However, the main reason for not using them is that current collections do not contain the kind of information necessary in this work.[85]

7.2.5 User-oriented Evaluation

User-oriented evaluation tends to involve largely qualitative methods to help understand some complex IRS environments. Thus, individual case studies are performed to provide the depthe of analysis. The result is usually a small scale evaluation which through its in-depth analysis can highlight wider aspects of the system (Ellis, 1990). In case studies, users can be observed as to what they do or they can be asked. However, difficulties arise as often what people say they do is not exactly the same as what they do (Lancaster, 1978).

By nature, user-oriented evaluation focuses on *pertinence* which is why results are qualitative. In these experiments, although users are approached for relevance judgements this does not constitute a case study as such. Thus, it may be more appropriate to refer to it as *user-based evaluation*.

Users of an IRS are extremely important for a realistic final analysis of a system's performance. However, the experiments described here have a degree of objectivity built in. When users were

[85]Queries in test collections are independent of each other. One of the major arguments in this work is that there is a connection between queries and that is why the notion of (user) *context* may be useful in (users') subsequent queries. Therefore, evaluating *context learning* techniques would require a group of consecutive (or possibly related) queries from particular users to be supplied in the test collection, which currently is not the case.

approached for relevance judgements care was taken to help ensure that users gave *relevance* and not *pertinence* judgements. This enables the usage of measures such as *precision* and *recall.*

7.3 Design of the Experiments

The types of evaluation methods and measures preferred were discussed earlier (Section 7.2). Evaluation was necessary not only to see the effectiveness (and performance) of the *learning* (CL) algorithms but also to help clarify some questions relating to the notion of *context* and its role in document ranking. The two experiments performed are described in this section.

It is hoped that the results would shed some light on questions such as the following: Is a *context* C_n derived after n queries useful for the subsequent query Q_{n+1}? Are these two (C_n and Q_{n+1}) usefully related? Is *context* something that is *constant* or is it *changing*? What is the role and effect of *minimum coverage*? What is the role of term frequency (in relevant documents) in identifying *context*? Does document ordering, particularly within the same *weight-block* improve with the use of *context*? How does the frequency of users using the system affect *context* and document ranking?

7.3.1 Experiment 1: Deciding which Learning Algorithm

In Sections 7.1.4, 5.6, various learning algorithms were described. They represented theoretically differing viewpoints regarding the notion of *context.* Having defined these versions the next step is to perform the experiments to test them.

The purpose of this experiment was to decide which of the CL algorithms was a better candidate for improving the original version of the IRS (Okapi-plain). Although the document lists produced by Okapi-plain were used as a basis, the aim was not to compare with Okapi, for that experiment 2 was devised. At this stage, the aim was to compare CL versions amongst each other. For this, as explained in Section 7.2.4, a user-oriented evaluation was chosen due to test collections not holding the type of information necessary.

The user-oriented approach involved using an implemented IRS (such as Okapi) and its users. In an ideal situation, to evaluate the CL variations more fully, user relevance judgements would be obtained for all documents output from all of the different algorithms. Figures for comparisons would then be calculated. A reasonable sized subset would be at the very least 10 documents for each algorithm. With 14 algorithms this would mean 140 documents if there were no overlapping ones between the algorithms. This is a very high number to expect a user, or even an an expert, to assess and make relevance judgements on. Over all the CL versions, the total can be as high as 1400 if evaluation is done on 10 users.[86] This was not very feasible due to the constraints on resources - particularly users' time and availability.

Regarding the choice of users, a group of users can be determined either by some mechanism to ensure a certain amount of uniformity (homogeneity) or by frequency of usage. If we are

[86] In reality there were 11 users for this experiment.

aiming for uniformity, the users can be students preparing their MSc dissertations (to help ensure some similarity in the type of information need) or they could be MSc students required to do some coursework. On the whole, however, this would not give enough scope for many queries to necessarily be generated. They may be more likely to do more searches for their dissertations but the timing of this is very restrictive. In this case, not only was it not the time of the academic year for preparing dissertations but the users were not using the system as frequently as they were prior to the pilot study (chapter 6). The frequent users since then have changed. There have been new releases (not related to the work of this thesis) of the system with some being preceded by new registration schemes meaning that many users have had to be re-registered in order to continue using the system. These difficulties are characteristic of working in an experimental IR environment and can affect the operational details of evaluation for real users with real needs.

In choosing users, it is also possible to focus on those using the IRS more frequently. This can be done in two ways i.e. either at the beginning of their using the system or after they have performed a considerable number of searches. If they are identified at the start, they can be encouraged to use the system (due to thesis motivation, for example) and their subsequent queries monitored over a period. If, on the other hand, they are identified later this could be done by looking at their frequency of system usage. Subsequently, they can be asked (after some reasonable number of queries for the system to learn from) to make relevance judgements on their last query. This would make for somewhat artificial queries. Even though the general need may be real, the frequency and nature of queries may well be different.

One way would be to have, for example two versions of the system running - the plain version and the one with learning. A fairly homogeneous group of users with real information needs such as MSc students doing their dissertations could be used to compare for any improvement (e.g. Smithson (1989)). Such a method can only usefully be applied for this stage of evaluation if there were a significantly larger number of such users, in the order of 150. This is because the purpose, at this stage, is to decide amongst a fairly large (14 plus Okapi-plain which is used as a basis) number of algorithms as to which is likely to result in a more significant improvement on the current system. Therefore, 15 *live* versions of the system running would be required with 10 or so users allocated to each version (without prior knowledge of which version they are using). Therefore, even though these versions have been implemented, to have them up and running in parallel for the users was not considered feasible.

There was also the additional issue as to whether to, in operational terms, perform the evaluation in *live* or *batch* (see also Section 7.1.5). *Live* would mean having the CL versions of Okapi and its original (plain) form up and running with a group of users (unbeknown to themselves) being allocated to use a particular version each. *Batch* processing would mean accumulating the queries and their corresponding relevance judgements and after some reasonable number of them perform all types of learning for each user. This method of evaluation is the one chosen.

Evaluating the learning aspect of a *live* system has its difficulties mainly in terms of system organisation management, support and maintenance. Having several learning versions and the

'plain' version of the system running in parallel adds complexity to the evaluation. Users need to be allocated to a version and for a 'fair' comparison, all versions should be used roughly the same number of times. If the group of users chosen were MSc students then with this method, a large number of students would have been required for this stage of evaluation, in order cover all 14 CL versions. Under the current experimental environment, operational conditions and due to the number of students available, this would not have been feasible (especially considering that all versions should be used available simulataneously). This would require some homogeneity in the groups of users and their system usage for later comparisons.

The documents generated by Okapi-plain for the last query was presented to the user for relevance judgements. Obtaining relevance feedback for this *last* query can be done in three ways. Firstly, the user can be approached after a series of queries and asked to specify what they think their next query would be. Secondly, the user can be left to use the system as normal but as soon as they have typed in this *last* query they can be approached for relevance judgements (before seeing the ranked documents produced by the system). Lastly, the user can be approached after inputting the query and seeing the ranked document list as produced by the system. However, in this case, if users are approached immediately after performing their query, it is more likely that they will be prejudiced in their relevance assessments. Waiting too long, on the other hand, before asking for their judgements may mean they have already forgotten the query and therefore the *context* in which it was made. Thus, a time period of within the week of the query made was thought to be reasonable.

The 11 queries for the different users for this experiment can be found in the appendix (Appendix K: Queries - Evaluation 1). After performing the experiment, precision values were obtained for each user and CL version and the top 5% values were marked at each cut-off point to identify those with better performances (Appendix O: Precision Values at Various Cut-off Points - Evaluation 1).

7.3.2 Experiment 2: Does the Algorithm Improve Document Ranking

The purpose of this evaluation is to establish whether the candidate algorithm(s) as determined from the previous experiment are an improvement (preferably significant) on the 'plain' version of Okapi originally installed. In a wider sense, however, the aim is to show whether *context* is useful or not for the present query. The experiment was similar to experiment 1 but was different in the way in which the document list for obtaining relevance judgements was produced.

Initially, it was anticipated that one specific CL algorithm would be the clear winner. However, in the end there were four algorithms considered worth evaluating further and comparing with Okapi. This situation is not untypical in IR, where often some approaches/ techniques appear to work better for certain situations whilst others are better for different ones. Thus, rather than deciding which of the four would be the eventual candidate for final evaluation on theoretical grounds (with perhaps some arbitrary assumptions), all four and Okapi-plain were used for this stage of evaluation.

Test collections were not seriously considered for this experiment, partly because of the reasons explained in experiment 1 and also because particularly at this stage it was even more important to have a user-oriented evaluation for a more realistic view of the performance of a CL.

Similarly to experiment 1, frequent users were identified and approached for relevance judgements for their latest query. Unlike Experiment 1, however, they were shown the (top 10) documents produced by each remaining CL and Okapi-plain. This meant a maximum of 50 documents to be shown to the user. In practice, because of some documents being retrieved by more than one version, there were overlaps which resulted in 30-40 documents per user. The precisions for these were calculated at various cut-off points and the highest ones identified. The queries for different users for this experiment can be found in the appendix (Appendix L: Queries - Evaluation 2). The precision values obtained are in Appendix P (Precision Values at Various Cut-off Points - Evaluation 2).

7.4 Operational Conditions of the Experiments

The aim of the experiments was to eventually establish whether learning in IR, as implemented in this work, is useful or an improvement on the existing system. Particularly, it was to see the effect of applying *context learning* for breaking up ties in *document scores* i.e. breaking the *weight-blocks*. Experiment 1 was done in order to reduce the number of algorithms (algorithmic options) to do the final evaluation. Comparisons were made between the versions of the system (and not with Okapi-plain). Experiment 2 was done to see if any of these remaining versions improved on the plain IRS.

The experimental method for both stages of evaluation was essentially the same. Thus, this section will describe the operational details, test environment and test procedures of both experiments together. Some numerical figures may vary between the two experiments and these will be indicated appropriately.

7.4.1 The System

Both experiments were done with the Okapi IR systems (on Sun workstations) using a subsection of the Inspec database relating to computer science, information technology, engineering abstracts. Some 95,000 records were used in experiment 1 and around 224,000 records were used in experiment 2. In both cases, the document records consisted of title, author, abstracts, subject headings, year of publication and so on. In the Okapi IRS, the result of user searches are shown in the form of ranked documents. Initially, only brief details relating to the documents such as the title, author and publication year are shown. From these the user may choose to look at a particular documents in greater detail. In which case he/she is shown the abstract, subject and index headings for that record. After this, the user is asked to make a relevance judgement on that document as to whether it was appropriate to his/her need (The question "Is this the sort of thing you are looking for" is asked). All queries and their respective relevance judgements are recorded in logs and all information necessary to repeat searches for batch processing is kept.

7.4.2 The Users

The users were identified from the pool of people using the system *frequently*. These were

> For experiment 1: Those using system in last 3 months at least 5 times. This does not necessarily mean all had positive relevance judgements. The remaining users were 11.
> For experiment 2: Those using system in last 3 months at least 5 times. This does not necessarily mean all had positive relevance judgements. The remaining users were 9.

They consisted of people in the academic environment such as teaching staff, researchers MSc and undergraduate students. Due to the nature of the database, they were predominantly from Computer Science, Information Engineering, Information Science, System Analysis / Business Computing Departments.

The frequent users and the IR sessions and queries indicated above did not necessarily mean that they could be used for the experiment. This depended on what they did in their sessions. *Context learning* does relies on positive relevance feedback (i.e. some relevant documents to be identified). Thus, although the user may have looked an extensive list of documents but if none were chosen to be relevant (or looked at in more detail in the first place) this did not contribute to the *learning*. Likewise, if they exited the session before seeing the lists, these sessions did not contribute to *learning*.

Considering these factors, the resulting number of frequent users for the first experiment was 11 and for the second 9. The users and their queries for Experiments 1 and 2 can be found in Appendices I and J (Number of Online Queries and Sessions: Evaluation 1 and 2).

7.4.3 The Procedure

Section 7.1 discussed the implementation of the CL in the Okapi IRS. With this background, the procedure for performing any learning for both of the experiments is done in *batch* mode and is as follows :

> 1. All online sessions, for a particular user, that fall within the originally specified time period are identified and their chronological order kept.
> 2. Each session is broken down into the queries that form it. Each query 'chunk' includes the query and any relevance feedback that goes with it. Any amendments to a query in Okapi result in a new query that is searched for.
> 3. From these chronologically ordered 'chunks', the ones from which there was no positive relevance feedback are removed.
> 4. The remaining query 'chunks' form a *query data file* each. These are in the form that the learner (written in Prolog) can accept as its input. The data files are fed to a particular learning version of Okapi (see also Fig.7.1). After each iteration a new *context* is formed. This keeps changing iteratively after each query. However, the *context* terms are used to affect the ordering of documents

only in the last query which is the one that CL evaluation is performed on. The learning is done retrospectively (batch).

For both experiments, the above procedure of performing learning is the same. However, the way in which the final list of ranked documents is produced for each evaluation stage and the meaning behind the results for the two experiments are different.

For Experiment 1, although the query datafiles were put through all CL versions, the users actually evaluated the results of the plain version of the system. They were asked relevance judgements on the top 20 documents presented by Okapi-plain.[87] As explained earlier, the purpose of this experiment was not to evaluate the algorithms against Okapi but to compare them with each other to identify which would be likely to perform better when finally compared with the standard system (version without *learning*). Thus, the relevance judgements for the output of the standard system were used as a testbed.

For Experiment 2, the query datafiles were run on a total of five versions of the system (including the plain version). The top 10 documents from all lists generated were merged. They were then put into random order so as not to prejudice the users in their relevance judgements. The resulting list consisted of a maximum of 50 documents.[88] The users were then asked to make relevance judgements on these documents and the resulting precision values were used to compare against the plain IRS. The aim here was to identify if any of the learning versions were indeed an improvement on the existing (plain) system and which *learning* criteria played a role in this.

The relevance assessments in both experiments were gathered for users' *last* queries on the system and they were approached to make these judgements within the week of performing it. Prior to this, they were not aware of these particular experiments and the participation required from them specifically.[89]

The users were shown an instruction sheet (Appendix H: Evaluation of Offline Print - Relevance Assessment Instructions) explaining how to make their relevance judgements. The judgements consisted of three categories: *Relevant (R)*, *Partially relevant (P)* and *Non-relevant (N)*. They were asked to make these judgements irrespective of whether they had actually seen the original document. For example, if they already had obtained the document and knew it to be relevant and therefore did not need it again, they were still asked to mark it as relevant. This was to try and ensure that *relevance* judgements and not *pertinence* judgements were obtained. However, despite this, it is unavoidable that if the user has previously seen the document that they would be judging it on a different basis.

---

[87]If any query expansion had been done then the top 10 (half of 20) from the original query and the rest from the expanded query.

[88]This is if there are no overlaps in the document sets produced by each version.

[89]However, when the users were registered on the system initially, they were informed that the system was experimental and that they may be approached about their online sessions.

7.4.4 Evaluation Measures

The reasons for the measures preferred have been discussed in Section 7.2. Therefore, this section provides a summary of the measures chosen and the way in which they are used.

The results for both experiments consist of precision values generated from the *R, P, N* figures in the users' relevance judgements. Calculating recall is not possible as (unlike in test collections) there is no figure for the total number of documents relevant in the database for a particular query. In fact, even finding the total number of relevant documents in the retrieved set is not very simple as it would require the users to make judgements on documents usually in the order of a thousand. Hence, in ranked systems, the point at which the user stops searching (cut-off point) is used to divide the set into two; retrieved and non-retrieved.

Various cut-off points (5, 10, 15, 20) were used for analysing the precision at certain rank positions in the document list. In the absence of recall values, this approach helps identify some trends in relevance e.g. how many items are relevant near the top of the list compared to those further down the list.

The results were essentially in the form of relevance judgements by frequent users of the (Okapi) system on a top subset of ranked documents (Appendices M and N: Relevance Judgement Results Evaluation 1 and 2). For each algorithm there is a ranked set of documents produced. A fixed number of documents was taken from each set. These were merged and presented in a way (randomised) so as to help users make fairly unprejudiced relevance judgements. From these judgements, precision values were calculated for the cut-off points decided. The analysis that follows mainly stems from these precision values.

7.5 Results and Evaluation

There are two sets of results described in this section, one for each experiment. As described in section 7.4, the method of obtaining these figures are similar for both experiments and so this section focuses on the nature of the results for the experiments, their analysis and meaning.

7.5.1 Results (Experiment 1) and Analysis

Experiment 1, as mentioned previously, was done in order to decide amongst the *learning* algorithms as to which were more likely to perform better when eventually compared to the existing system in Experiment 2.

The precision values for each learner (over all users) at cut-off points 5, 10, 15 and 20, for the first experiment, are shown in Appendix O (Precision Values at Various Cut-off Points - Evaluation 1). Two precision values are given: for relevant (R) documents; for relevant together with partially relevant (R+P) documents. As discussed earlier (Section 7.2.3), this is to cover for a spectrum of relevance judgements.

Thus, each learner has two precision values at each cut-off point. For this purpose, we can refer to a 'slot' as all precision values of the same type (either P_R or P_{R+P}) for a particular cut-off

point. For each slot, the highest precision value was identified and those values within the top 5% (and including the highest value) were marked. For example if the highest precision value amongst the context learners, calculated by P_{R+P}, for rank position 10 is 88% then all those between (and including) 83-88% are considered for that slot (as per Table 1 in Appendix O: Precision Values at Various Cut-off Points - Evaluation 1). The process is applied independently for each slot.

Table 7.2 shows the slots concerned and the respective range of percentage values considered (top 5%). For example, with cut-off point (rank position) 15, in the **R+P** column, the highest precision value (reached by L17) is 82% so the range considered for precision P_{R+P} is 77% to 82%. The later table, Table 7.3, is an example of the precision values obtained for a context learner (L3). In this case, only two slots had a precision value within the top 5%, as described above. At cut-off 15, precision P_{R+P} of 79% is highlighted as it falls within the range of values considered. Likewise, at cut-off 20, the value for precision P_R is within the range of values considered for that slot. Appendix O (Precision Values at Various Cut-off Points: Evaluation 1) contains the precision values for each learner and highlights the ones which fall within the respective ranges described.

Rank Position	R/A (%)	(R+P)/A (%)
5	52 - 57	85 - 90
10	47 - 52	83 - 88
15	45 - 50	77 - 82
20	42 - 48	79 - 84

Table 7.2: The range of percentage values considered for each rank position for the two types of precision values, for Experiment 1.

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	9	44	77
10	16	38	81
15	19	42	79
20	29	45	72

Table 7.3: Precision values at various cut-off points for Context Learner L3 for Experiment 1.

A baseline figure for the plain version was also calculated, mainly to identify if any of the algorithms appeared to be performing considerably worse than Okapi.[90] The precision figures were 47% when considering only R documents and 70% for R+P. These values were obtained by averaging all users' precision values for the plain system.

Considering these top (5%) precision values, the algorithms which fell into this category using R and R+P are shown in Table 7.6 below. The criteria that were tested in these CL algorithms were discussed in Section 7.1.4 and 5.6. However, the remaining ones chosen from these for the second experiment are given in Table 7.5.

Cut-off Point	Precision P_R	Precision P_{R+P}
5	L1,L2,L15,L18	L1,L3,L16,L17,L18
10	L1,L2,L15,L18	L1,L2,L17,L18
15	L1,L2,L12,L14,L15,L17	L1,L2,L3,L4,L13,L17
20	L1,L2,L3,L15,L17	L1,L2,L17

Table 7.4: The Context Learners with precision in the top 5%.

Table 7.6 indicates that when using R as a basis for precision calculation

- L1/L2, L15 performed consistently well for all rank positions
- L18 performs well at top ranking positions
- L17 performed well on the lower rank positions

When using R+P as a basis for precision calculation

- L1/L2, L17 performed consistently well for all rank positions.
- L18 performs well at top ranking positions.

The precision values for L1 and L2 were identical (for both R and R+P), therefore indicating that there is no difference in performance between these two algorithms. The difference between these two algorithms was whether they included any duplicates (over the user's relevance judgements so far) of relevant documents. Given the identical performances, L2 is a better candidate algorithm as it is appropriate to both the strong and weak hypotheses (as argued in Section 5.6.1).

[90]The CLs resulted in higher precision values. However, this is not indicative of how they will perform when the corresponding document lists for the CLs are shown to the user.

Thus, algorithms L2, L15, L17, L18 were used for the second experiment alongside the plain Okapi system.

These algorithms are more fully defined in Section 5.6. Nevertheless, their main characteristics can be summarised as follows:

Context Learner	Modules
L2	A^{II}, B^{I} (T_1=1), C^{I}, $D_{\alpha}{}^{I}$, $D_{\beta}{}^{II}$
L15	A^{I}, B^{III} (P=1/3), C^{II} (T_2=30), $D_{\alpha}{}^{I}$, $D_{\beta}{}^{II}$
L17	A^{IV}, B^{II} (T_1=1), C^{II} (T_2=30), $D_{\alpha}{}^{I}$, $D_{\beta}{}^{II}$
L18	A^{IV}, B^{II} (T_1=1), C^{II} (T_2=6), $D_{\alpha}{}^{I}$, $D_{\beta}{}^{II}$

Table 7.5: The modules for the Context Learners used in Experiment 2.

Modules		Description
A	A^{I}	R_{latest} with duplicates
	A^{II}	R_{latest} remove duplicates
	A^{IV}	R_{total} remove duplicates
B	B^{I}	A term's doc count has to be greater than *threshold1* ($m.dc>T_1$), cover all docs*
	B^{II}	A term's doc count has to be greater than *threshold1* and limit *ntA* ($P \times T_2$)
	B^{III}	Sort terms according to doc count and weight then limit *ntA* ($P \times T_2$)
	$T_1=1$	*Threshold1* = 1, applies to no. of relevant documents a term occurs in *(m.dc)*
	$P=½$	*Proportion* = ½, *proportion* × *threshold2* sets a limit for *ntA*
	$P=⅓$	*Proportion* = ⅓, *proportion* × *threshold2* sets a limit for *ntA*
C	C^{I}	$(t \in A \rightarrow t \in U) \wedge ((t \in P \rightarrow t \in U) \vee (t \in P \rightarrow t \in H))$
	C^{II}	$(t \in A \rightarrow t \in U) \wedge ((t \in P \rightarrow t \in U) \vee (t \in P \rightarrow t \in H))$ with *threshold2*
	$T_2=6$	*Threshold2* = 6, the max. no. of terms in the *Used* set *(ntU)*
	$T_2=30$	*Threshold2* = 30, the max. no. of terms in the *Used* set *(ntU)*
D	D_{α}^{I}	$[(t \in U' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in U' \wedge t \in P) \rightarrow t \in U]$
	D_{β}^{II}	$[(t \in H' \wedge t \in A) \rightarrow t \in U] \wedge [(t \in H' \wedge t \in P) \rightarrow t \in H]$

Table 7.6: Summary table of the modules and variables concerning the Context Learner versions for Experiment 2.

* To cover all documents, if first criteria does not already, select from remaining terms with highest document counts and weights (in the case of ties).

Abbreviations:

R_{latest}: Set of relevant documents from last query
R_{total}: Set of relevant documents from all queries
A: *Active* set (for current iteration *n*)
P: *Passive* set (for current iteration *n*)
P: Proportion of threshold2 which sets the maximum value for *ntA* - for modules B^{II}, B^{III}
T_1: *Threshold1*, applies to no. of relevant documents a term is found in - for modules B^{I}, B^{II}
T_2: *Threshold2*, max. no. of terms in the *Used* set - for module C^{II}
ntA: No. of terms in *Active* set.
ntU: No. of terms in *Used* set.
U: Current *Used* set (also referred to as U_n for current iteration *n*)
U': Previous *Used* set (also referred to as U_{n-1})
H: Current *Hold* set (also referred to as H_n for current iteration *n*)
H': Previous *Hold* set (also referred to as H_{n-1})
α: U' or U_{n-1}
β: H' or H_{n-1}
D_α: *D* modules concerned with α set
D_β: *D* modules concerned with β set

Note: For brevity, the sets U, H, A, and P at iteration n (current iteration) have not been referred to as U_n, H_n, A_n and P_n.

On Module A:

Evidence for the difference between A^{I} and the others (A^{II}, A^{IV}), considering the precision values for L1 and L2 does not really exist. It can also be an indication that duplicates do not occur often as the terms extracted also indicate.

On Module B:

The preference is between a more simple case of a threshold value (1) being applied or a proportionate amount of total terms (in the *context*) being renewed with the latest *active* terms.

On Module C:

The same criteria for merging has been used in all CL algorithms concerned. However, some CLs used a limit for the number of terms in the *(used) context* set. The lower the number of terms allowed in the *used* set the more likely that *passive* terms will not have much chance of being included in the *used context* terms. The numbers in the sets will vary according to decisions made in Module D relating to the history of a term.

On Module D:

If a term is in the *active* set for the current *learning iteration* then it ought to be put into the *used* set, irrespective of whether it was in use or on *hold* before. In the case of a currently *passive* term, its previous position in the *context* is the determinant. If it was on *hold* before, it remains so and if it was *used* in the previous iteration then it is kept in the same set.

For the CL algorithms precision values for the two most frequent users (User no 6 and 13 with 48 and 13 queries respectively) indicated the following for the above mentioned algorithms. One would expect on average for the CLs to give better performance results for the more frequent users.

For the specific *learners*, the following can be deduced from their precision values (see also Tables 2, 10, 12, 13 in Appendix M: Relevance Judgement Results - Evaluation 1):

L2: Not much overlap between the document sets retrieved for Okapi-plain and the version with L2 (Table 2, Appendix M: Relevance Judgement Results - Evaluation 1, shows the high 'O' (Other) values). Hence, precision values tend to be near 0%, for both users.

L15: Not much overlap between (top) documents retrived by Okapi-plain and this algorithm for User 6 but more overlap occured for User 13 (Table 10, in the above described appendix).

L17: Not much overlap between document lists produced by the plain system and the version with this algorithm, for both users.

L18: More of an overlap between the document list produced by this algorithm and Okapi-plain. Thus, in comparison with the other three algorithms, this one was more like Okapi (as retrieved some same documents).

The performance of the algorithms was also looked at specifically for frequent users of the system and for users whose last query (for which the evaluation was done) was long.

The purpose of this experiment and analysis was not to compare the algorithms with Okapi-plain but with each other, even though using results obtained from the plain system were used as a basis. Without some documents being common to both sets, however, there are no precision values for this experiment as was the case for the above algorithms for these users.

7.5.2 Results (Experiment 2) and Analysis

Experiment 2 was done in order to see how the learning algorithms (L2, L15, L17, L18) performed in comparison with Okapi-plain.

The precision values at cut-off points 5 and 10 for the four learners and Okapi-plain can be found in Appendix P (The Precision Values at Various Cut-off Points - Evaluation 2).

The results were analysed in the same way as those for experiment 1. However, rather than considering the precision values in the top 5%, the priority was for values higher than those for the plain Okapi system. Below are the precision values for the plain Okapi system (Table 7.7) and those for context learner L18 (Table 7.8). The highlighted values indicate that for both types of precision values (P_R and P_{R+P}) at both cut-off points the values for L18 were higher than Okapi-plain.

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	45	42	71
10	85	32	68

Table 7.7: The precision values at various cut-off points for the plain Okapi system, for Experiment 2.

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	45	47	80
10	90	44	77

Table 7.8: The precision values for context learner L18 for Experiment 2.

Regarding the four learners used for this experiment, the findings are as follows:

- L2 did worse than Okapi-plain.
- L15 was equivalent in performance to Okapi-plain.
- L17 performed the same as or worse than Okapi-plain (only one precision value was better, albeit marginally).
- L18 performed better than Okapi-plain. On average, L18 gave a 10% increase in the precision values at both cut-off points / rank positions (5 and 10). (E.g.: From 71% to 80% at cut-off 5, for both types of precision values, P_R and P_{R+P}).

Thus, it would appear that a more basic algorithm (involving a decision based on term document count threshold) with a lower limit on the *context* terms performs better. The lower limit ensures that there is a regular changeover in the *context* terms and hence the *context* is uptodate. This would enhance its use in subsequent queries.

7.5.3 Problems and Weaknesses

Some problems and weaknesses of the results are as follows:

- Documents retrieved from a query with one term all have the same *document score* (they fall into the same *weight-block*). Thus, in such cases *context* is applied to the whole of the retrieved document set, as that forms a single *weight-block*. There may be a difference in the precision values due to a difference in the number of documents retrieved and the number and size of *weight-blocks*.

- Although *minimal coverage* seems like a theoretically good idea, in practical terms, it is very difficult to ensure complete *minimal coverage* as there will always be some arbitrary criteria which has to be satisfied. For example, when two terms cover the same documents and occur the same number of times then the choice is random as to which term is to enter the *context* and be used or not. Additionally, there is the problem that most ways of obtaining minimal coverage produce too few a number of terms to effectively break up weight-blocks.

- It is possible that due to the small number of documents which were in both the set of documents which users' relevance judgements were based on (based on the output of the plain Okapi system in experiment 1) and the set produced from each context learner, an accurate assessment of the performance of the context learners may not have been obtained. More relevance feedback data may indicate that some of the algorithms eliminated for experiment 2 may not necessarily perform as badly as was previously predicted.

• In experiment 2, having less than four algorithms to test for would have made it possible to get relevance judgements on a larger number of documents. For example, the top 20 documents instead of 10, for each version could have been merged (for this evaluation it was a merge resulting from document sets of 10).

• It is difficult to make strong assertions about frequency of usage and query length with regards to a *learning* version of the system. To do this, much larger number of frequent users (with significant amounts of queries) is necessary. It is not enough to simply have a large total number of queries. Although the more terms in the query the more likely that the *weight-blocks* would not be large.

7.5.4 Overall Interpretations

Overall interpretation of results are as follows:

• There does appear to be a context for users' queries. Although it is not constant it does not change too quickly either. This makes it possible to do some useful learning on a batch of queries.

• At the query level, some queries will belong to that context for the user. Some, however, will not. For those that do, individual documents in a weight-block can be distinguished if context is used (Section 6.6.4). At worst, if the query is not in that context, the new document order in the weight-block should not be worse than the original (base) Okapi system.

• *Context* is based on term coverage in relevant documents indicated by the user, the term frequencies in these documents and the database. It would appear that on average the more simple the way of generating it the more likely it would be of use, particularly in document ordering.

• As the starting point for the work in this thesis was from a machine learning standpoint this was reflected in the empirical observations and subsequently the nature of the algorithms developed. This is not to say that a probabilistic approach (such as the one in Okapi), or indeed any other approach, cannot deal with context and/or the weight-block problem. However, there is no obvious way of giving less weight to terms resulting from previous queries and relevance judgements in a probabilistic model. Thus, such an approach could only be replaced with modules B and C. *Context learning* criteria such as those described in this work would be required for module D concerning the history of terms.

• The more frequent a term appears in a user's relevance documents, the more likely that it will remain in the *used context* term set.

• A CL is likely to be more useful in cases where two or three terms (which is the average) are typed in a query.

• The longer the query, the less there will be a need for such a CL as the query itself may be narrow enough to break up the *weight-blocks*. Sometimes in the plain Okapi system, even with four or five query terms the first 30-50 references can have the same document score i.e. belong to the same weight-block. This is shown in Appendices K and L which contain lists of queries and an indication of the number of references which fall into the weight-blocks (the number with maximum possible weight refers to the first weight-block). Weight-blocks can be broken up by using a CL, alternatively other information such as within-document term-frequency might also be used if available. However, the purpose behind such approaches does not involve an explicit intention to reorder documents to tailor for users' individual needs and contexts.

• The CL could be taken further to reorder the original document list or produce a new list (as opposed to reordering within weight-blocks). However, the argument (previously mentioned in this section) of not being able to worse than base Okapi does not apply in this case. More testing would be required for this.

Chapter 8

Conclusions

Earlier, several hypotheses were put forward regarding machine learning, information retrieval, user *contexts* and their applications (Section 4.5.). Briefly, two of these were general hyptheses, the others more specific. Moving from general to specific, the first hypothesis related to whether the fields of ML and IR could be of use to each other. The second addressed the notion and existence of *context*. The third hypothesis focused on the nature of context whilst the last referred to the application of context for document ranking. The following paragraphs discuss to what extent these hypotheses are true and the reasons for this.

The first hypothesis that ML can be of use to IR (or vice-versa) relates strongly at a higher (general) level to this work. This work, in the general sense, was an investigation into the application of ML in IR. To this end, developments in IR which have aspects of 'intelligence' were inspected and deficiencies were highlighted. From an ML point of view, existing techniques were examined for their applicability to an IRS environment. Of these, the general approach of learning by examples was seen to be the most suitable for IRS applications. It would appear that some *symbolic learning* techniques or strategies, as defined in the field of ML, can be applied to IR. However, they would require some tailoring. While there may be something in their main principles, existing algorithms do not fit IR automatically. Others have worked on learning in IR in a statistical way. Whatever the method, in the end we can only *learn* from terms, users and documents.

The context learner described in this thesis is based on symbolic learning. Although it has not been possible to apply an existing ML technique, nevertheless the CL developed shows that the general approach is viable.

The second hypothesis stated that, in an IRS environment, users have a particular *context* in mind when performing their queries. Although, the queries themselves may not be directly related to each other (for a specific user) they are nevertheless likely to be a part of the same *context*. Such a *context* spanning the user's queries creates the possibilities for the use of *learning* over user sessions/queries to overcome problems in an IRS.

Evidence for the existence of context falls into two categories. There is the initial evidence from the user logs which suggests that a common theme can be followed through a user's online searches and it is not likely that there will be many themes. This indeed was the basis for developing a CL. The second form of evidence is found in the CL versions developed. The fact that any of them work is evidence of the existence of context (either the strong or weak statement in the later hypothesis H3). If a single CL version had been tested and did not work this would not necessarily be evidence against the existence of context - there could have been some methodological problems associated with the development and testing of context. However, as there are CL versions that do work we can infer that context does exist and that there have not been any methodological problems for this purpose.

The third hypothesis focused on the nature of context and how it may or may not change after each query. The strong hypothesis asserted that the context after each query remained the same i.e. was stable. The weak one asserted that there may be a shift in the context. From the CL versions evaluated, the one that performed the best (L18) gives some kind of evidence against the strong hypothesis but supports the weaker hypothesis that context does change (albeit gradually). However, the evidence here is subject to methodological problems. Particularly, the way the experiments were set up does not provide evidence for the shift or otherwise of context. Ideally, the first experiment would also have been set up like the second where documents from all CLs are shown to the user for relevance judgements (not just those for the Okapi-plain system). Although the number of queries were large the number of users/cases and documents (Experiment 1) for each each case was relatively small.

The last hypothesis stated that context could be used to improve document ordering/ranking. Similarly to the previous hypothesis, there was an associated strong and weak argument for this. In the current Okapi system information from one search, even if it is in the same online user session, does not carry forward to the next one. In this sense, the probabilistic model is interpreted narrowly in the current system: from the present experiments, information from earlier searches by the same user *is* relevant to the estimation of probability of relevance in the current search. An extension of the probabilistic model to make use of this information might render the CL less useful. However, it is not clear how such an extension would work.

Here, the emphasis was on the *weight-block* problem involving a large number of documents having the same *document score* in a ranked document list. Several CL techniques representing differing theoretical viewpoints have been presented and tested to see their effect on reordering documents within a weight-block. Evaluations were performed to identify better candidates from the CL versions and later to make a comparison with the existing system.

It would appear that the way context is used is critical. The methodological problem of which CLs to use also applies here. Therefore, more research is required on why the other CLs in Experiment 2 (apart from L18) did not work. The current system focuses only on changing the ordering within weight-blocks. It should not be possible, on average, to do worse than the current random order of the documents within the weight-blocks (unless a chronological ordering is definitely what users prefer). If, however, we go beyond weight-block reordering, the problem of how to use/apply context is more critical and reordering may seriously degrade performance.

Relating to the context learner, the following conclusions can be drawn:

- Regarding the notion of *context*, it would appear that if users' contexts do change the rate of change is not too great.

- A simple algorithm based on term frequencies in related documents and a limit on the number in the *context* appeared to be more successful.

There are some difficulties in applying *learning* to IR, the main ones are as follows:

- Finding training cases or examples in IR has its problems in theoretical assumptions and in practice. There are difficulties in applying examples accross different users and different queries.

- The importance of the size of data in an IR problem is different to that in other domains. Not only are there practical problems involving resource constraints but also that there is no guarantee that techniques which give good performance results in small scale applications will perform in the same way on realistically sized databases.

- There are difficulties of developing, implementing and evaluating in a 'real' IRS. These are due to constant changes on the system related to other projects and its daily operation. Additionally, there are problems with user availability and unpredictability of user performance.

There are a number of areas in which this field of investigation can be furthered, these include:

- The CL described has some core modules. However, there is scope for variation in the techniques concerning each module, as not every permutation was covered in this investigation.

- Further investigations particularly on the set of terms put on *hold* in the current CL can be made. Whilst terms are on *hold*, they do not have a role in improving document ordering. They may, however, later be put into use if it is decided that they need to have a more important role in the *context*. Currently, terms remaining on *hold* for a long period are not discarded. It is possible to have some further criteria for eliminating terms which do not appear to have much use for a long period. Alternatively, they could be ranked.

- Some modules, such as that involving the allocation of terms from a new query into *active* and *passive* sets (Module B), can also be replaced with a probabilistic approach. However, it is not clear how once terms have been

classified for the current query they can be merged with terms relating to previous queries (and therefore *contexts*) with such an approach. Thus, inevitably, criteria such as that described here (Module D) would be necessary.

• In some cases (such as academic research environments) groups of users are also likely to have a *common context*. Such a CL over several users could be investigated further.

• At the development stage of a CL, test collections would be very instrumental in investigating particular techniques prior to a fully-implemented version. Recently, there has been an emphasis on forming test collections with a view to encouraging a common basis for evaluating IR techniques (e.g. TREC - Text Retrieval Conferences). However, the queries in the test collections are independent of each other. This makes them unsuitable for developing a CL. Future work in collections may incorporate a series of related queries i.e. groups of consecutive queries for particular users or at least topically related queries marked by experts.

• Terms used for *context learning* can be enhanced with the use of an online thesaurus.

• Evaluation can be done with more users, time and human resources permitting.

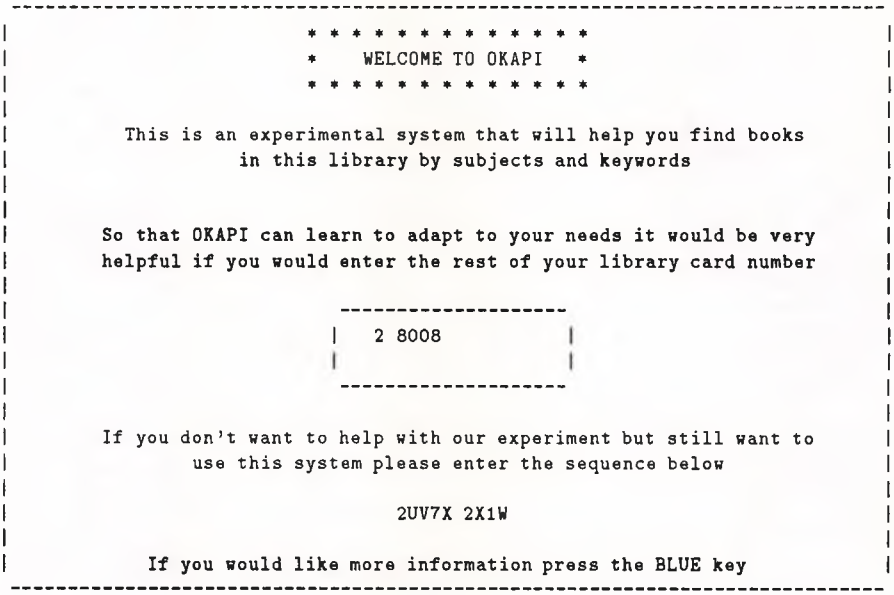
Whilst the CL can be implemented in any IRS it is very unlikely that the same version preffered in this application should be appropriate in others. However, a method for developing such a CL has been shown which also covers a wide range of decisions that are likely to made in a *context learning* IRS. A method for testing and evaluating these has also been given. Thus, it is possible that the methods presented in this thesis can be used as a basis for further investigations into the application of ML in IR.

Appendix A

Okapi System Description

This Appendix illustrates and describes the systems used in the City University library and for registered network users. All the Figures except A.21 and A.22 illustrate the system used at the terminal in the library. There are minor differences in the wording of messages and prompts between the library and the network systems.

Figure A.1: Welcome screen



If the user presses the BLUE key the message shown in Figur A.2 is superimposed over the "Welcome Screen"

If users enter a library card number a small window (or *snippet*) appears telling them the last time they used the system. If they enter the "special number" the snippet says "Thank you for your cooperation". The special number is intended to be unmemorable so that users are more likely to use their own library card number. It is randomly generated each time the

Figure A.2: Information window for welcome screen

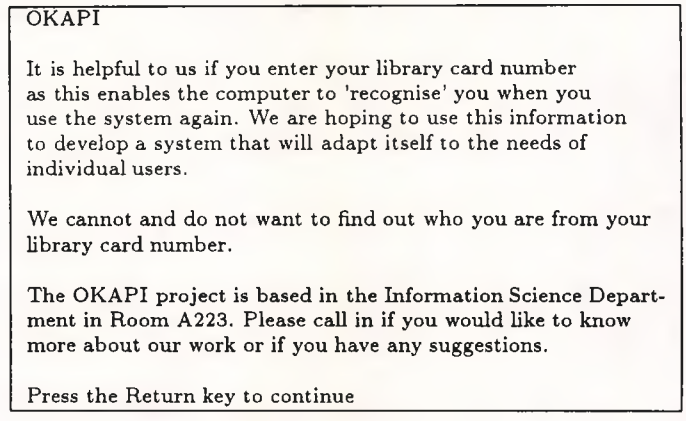
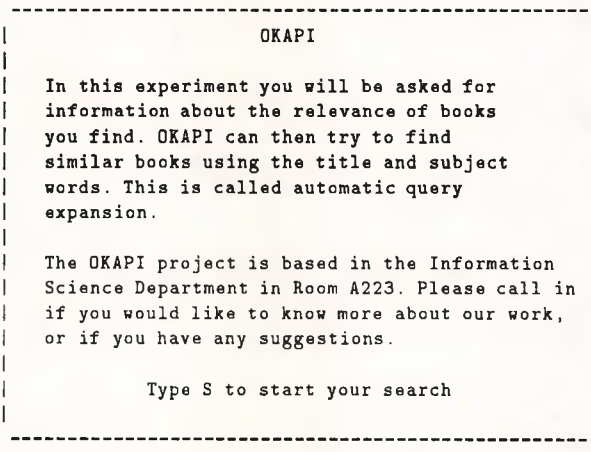


Figure A.3: Information offered after user identification



system returns to the welcome screen.

Users are given three attempts to enter either their own or the special number correctly. If they fail a snippet saying "sorry" appears under the number box, the screen goes blank for a few seconds and the welcome screen reappears. Once a number has been entered correctly users are offered the choice of either starting their search or seeing more information. The information at this stage is shown in Figure A.3.

If users do not press the RETURN key after typing in their search a snippet appears reminding them to do so.

When the search has been entered the "Results" screen of Figure A.6 appears

The "BLACK key to quit" snippet appears on every display as it is important that the user logs off at the end of the session.

A spelling mistake or the entering of a word not in the database results

Figure A.4: Search input screen

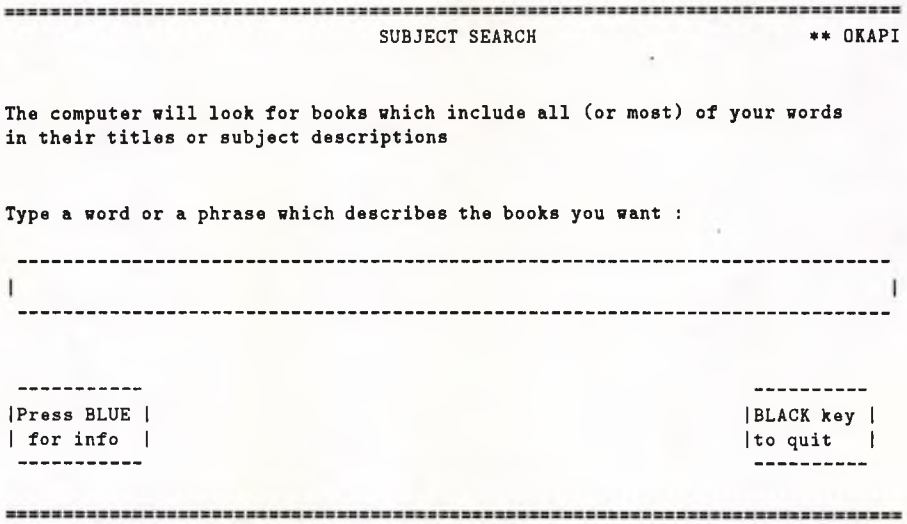


Figure A.5: Information offered from search input screen

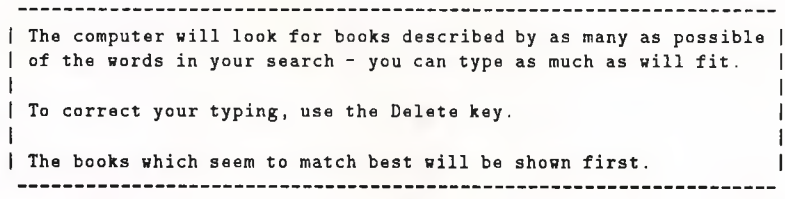
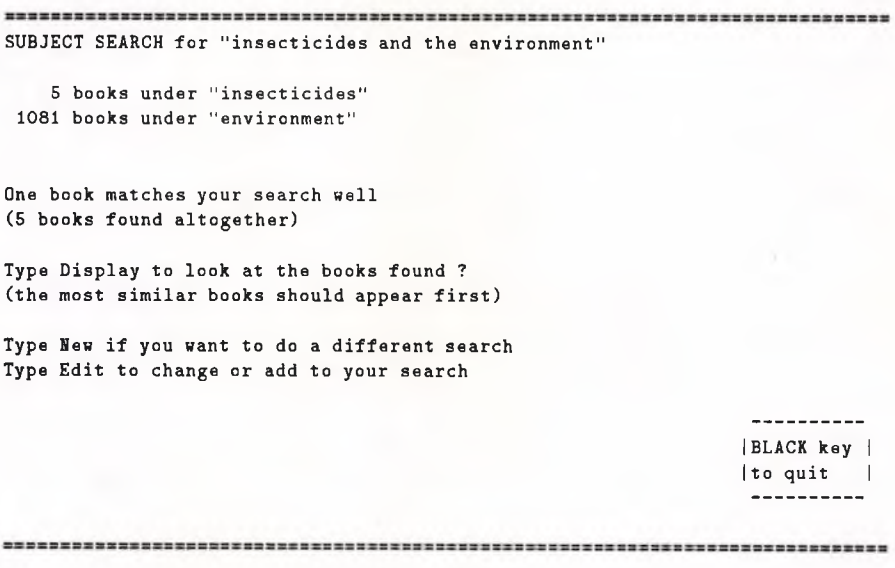


Figure A.6: Search results screen



in Figure A.7.

Figure A.7: Replacement of misspelt word

```
SUBJECT SEARCH for "insecticides and the enveronment"

    5 books under "insecticides"
CAN'T FIND "enveronment"


Press the return key if you want to change this word

Press the space bar to continue without this word

or type New if you want to do a new search

                                                          ----------
                                                         |BLACK key |
                                                         |to quit   |
                                                          ----------
```

Figure A.8: Brief record display

```
LIST OF BOOKS                                              Books 1 to 5 of 5
Search: "insecticides and the environment"

-------------------------------------------------------------------------------
No. Title                                          Author              Date
-------------------------------------------------------------------------------
1   Organochlorine insecticides : persistent organic.. (MORIARTY F)       1975
THE REST OF THE BOOKS MAY NOT MATCH YOUR SEARCH VERY WELL
2   Chlorinated insecticides.                      BROOKS G T          1974
3   Safe use of pesticides : twentieth report of the.. World Health Org..  1973
4   Insecticide and fungicide handbook for crop prote.. British Crop Pro.. 1972
5   Insecticides : action and metabolism.          O'BRIEN R D         1967
    ** END OF LIST **

-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
Type its number to see if a book is relevant

Type Options to see other things you can do                ----------
                                                          |BLACK key |
                                                          |to quit   |
                                                           ----------
```

In Figure A.10 the BLUE option informs users that if they choose some books the system will look for similar ones if they type MORE.

When the user chooses the book of Figure A.10 as relevant appropriate fields of the record is "indexed" and the terms extracted from it placed in a list with the original query terms. After this book has been chosen the

Figure A.9: Options from brief display

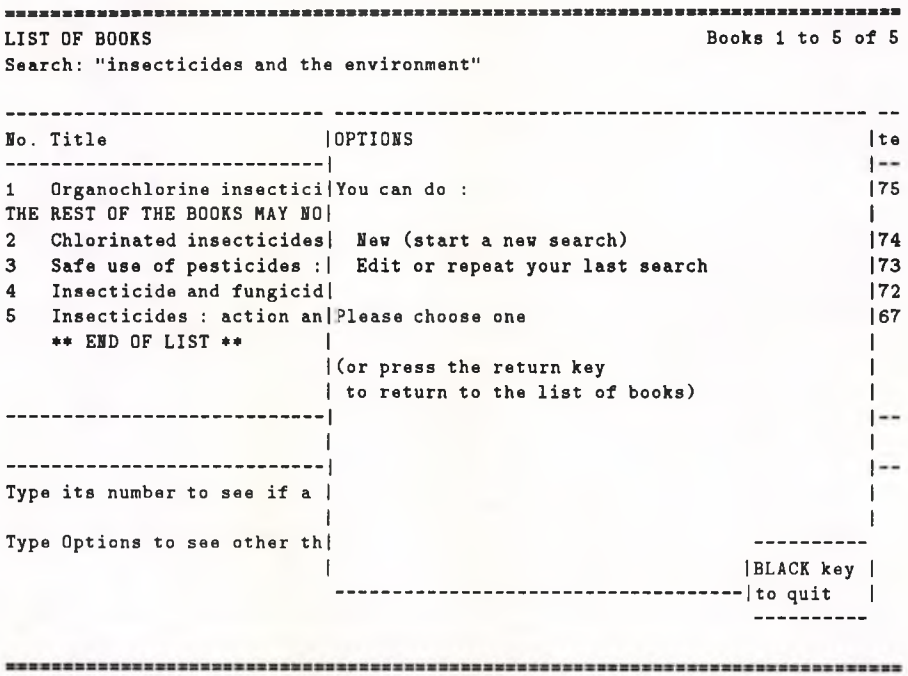
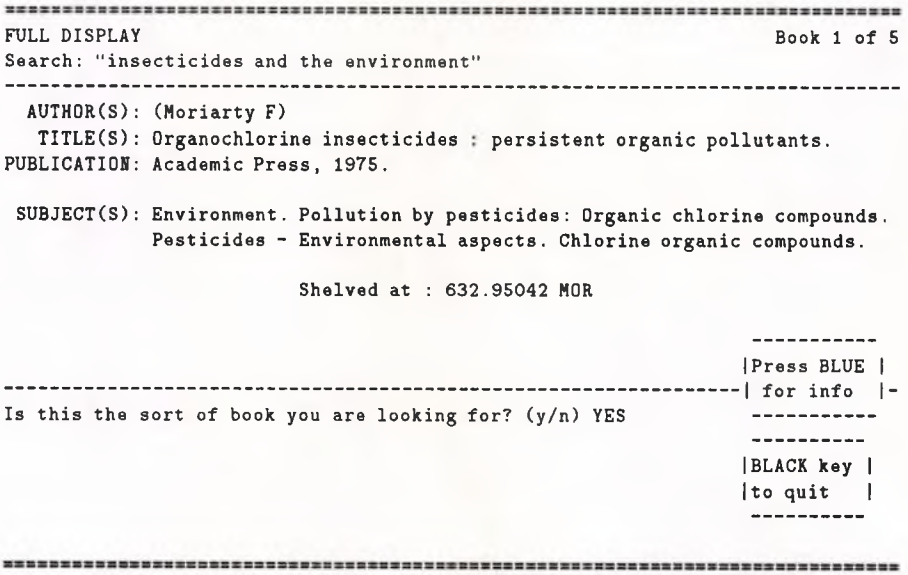


Figure A.10: Full record display



list will contain "insecticides" and "environment" from the original query, together with "organochlorine" and the other title words, "pollution", "pesticides" etc from the subject headings, and the Dewey number 632.95042. This list of terms, which is added to every time a record is chosen relevant by the user, is the list from which the "best" terms will be used for query expansion.

As only one book, in this example, was a good match to the search and as that has been looked at and chosen as relevant, the user is now given the choice of doing MORE (Figure A.11). If the original query results in at least three good matches MORE is not offered until several of the books have been looked at as the automatic query expansion usually works better with a larger pool of terms.

Figure A.11: Brief record display with "MORE" option

```
================================================================================
LIST OF BOOKS                                              Books 1 to 5 of 5
Search: "insecticides and the environment"

--------------------------------------------------------------------------------
No. Title                                          Author              Date
--------------------------------------------------------------------------------
1** Organochlorine insecticides : persistent organic.. (MORIARTY F)       1975
THE REST OF THE BOOKS MAY NOT MATCH YOUR SEARCH VERY WELL
2   Chlorinated insecticides.                      BROOKS G T          1974
3   Safe use of pesticides : twentieth report of the.. World Health Org.. 1973
4   Insecticide and fungicide handbook for crop prot.. British Crop Pro.. 1972
5   Insecticides : action and metabolism.          O'BRIEN R D         1967
    ** END OF LIST **

--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Type its number to see if a book is relevant

Type More to look for books similar to the one you chose MORE     ----------
Type Options to see other things you can do                       |BLACK key |
                                                                  |to quit   |
                                                                   ----------

================================================================================
```

When MORE is chosen the system takes the original search terms together with those added from the title and subject fields and the classmark of any chosen books and automatically expands the original query. The result of the new search is displayed in a window superimposed on the brief record display (Figure A.12).

Figure A.13 shows the two books found by the first iteration of query expansion in this search.

Figures A.14 and A.15 show that the records from the expanded query contain many of the terms extracted from the originally chosen record (Figure A.10). Although both these books are indexed under "environment" the records do not contain "insecticide" and the searcher would not have found them easily without query expansion.

Figure A.12: Results of query expansion

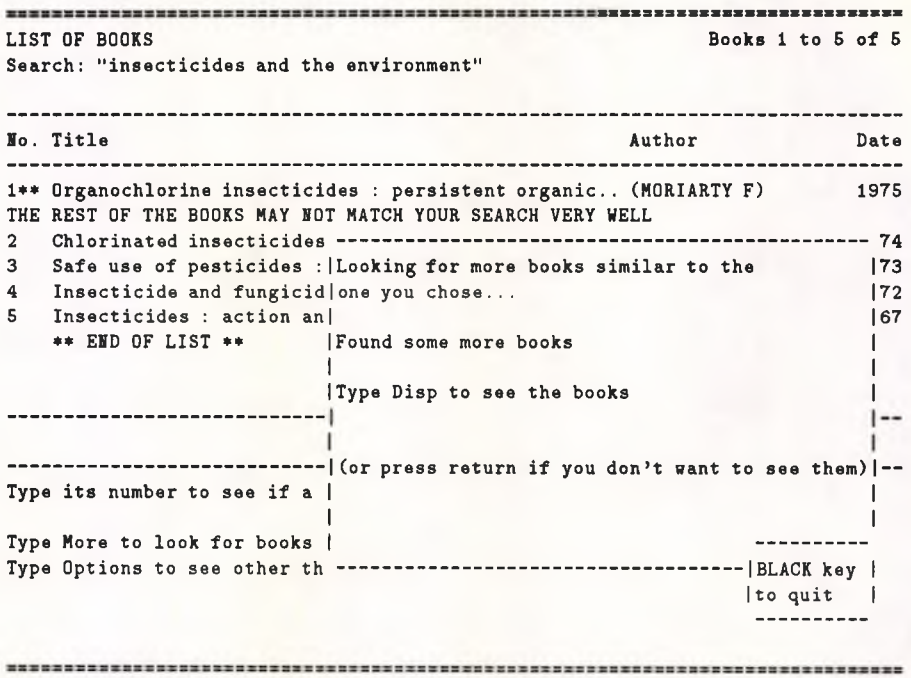


Figure A.13: Brief display from query expansion search

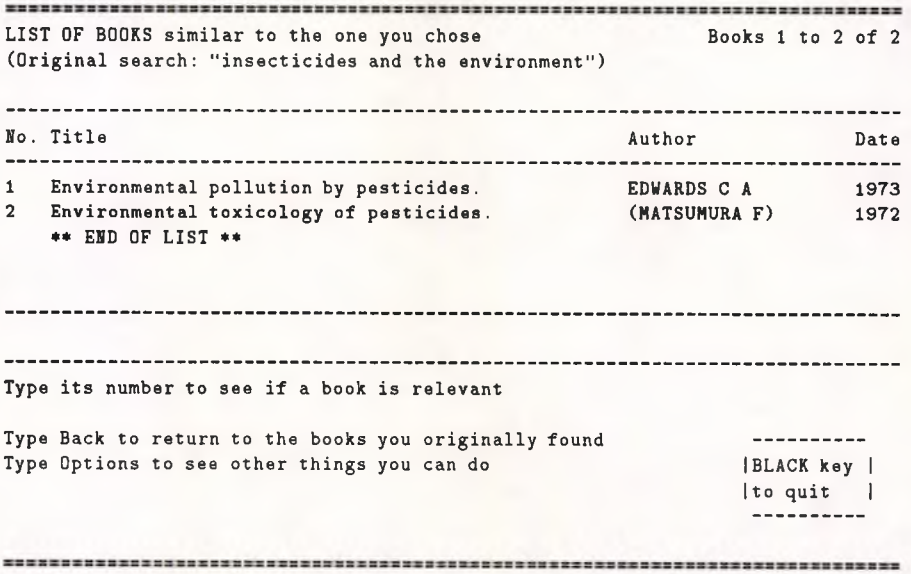


Figure A.14: Record from query expansion

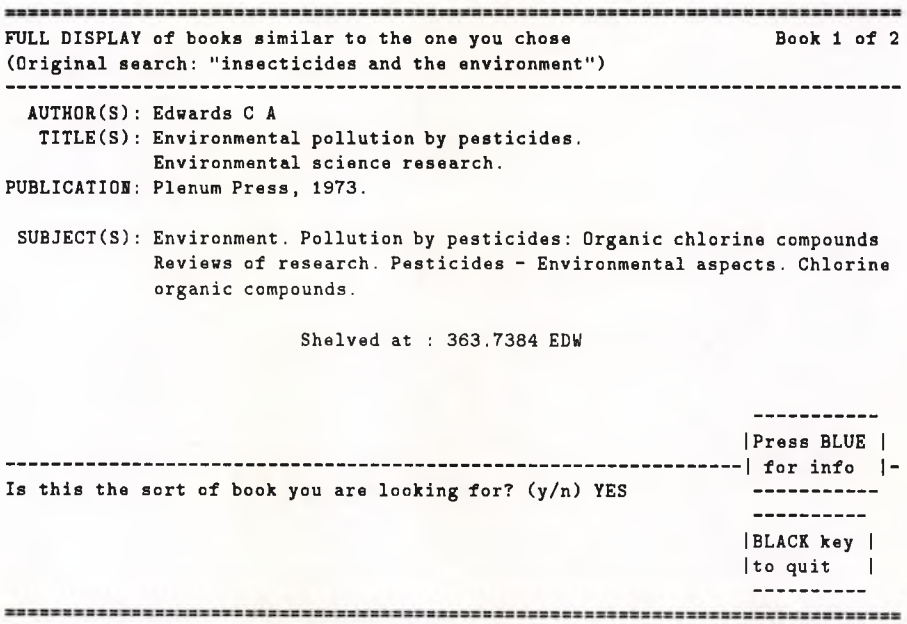
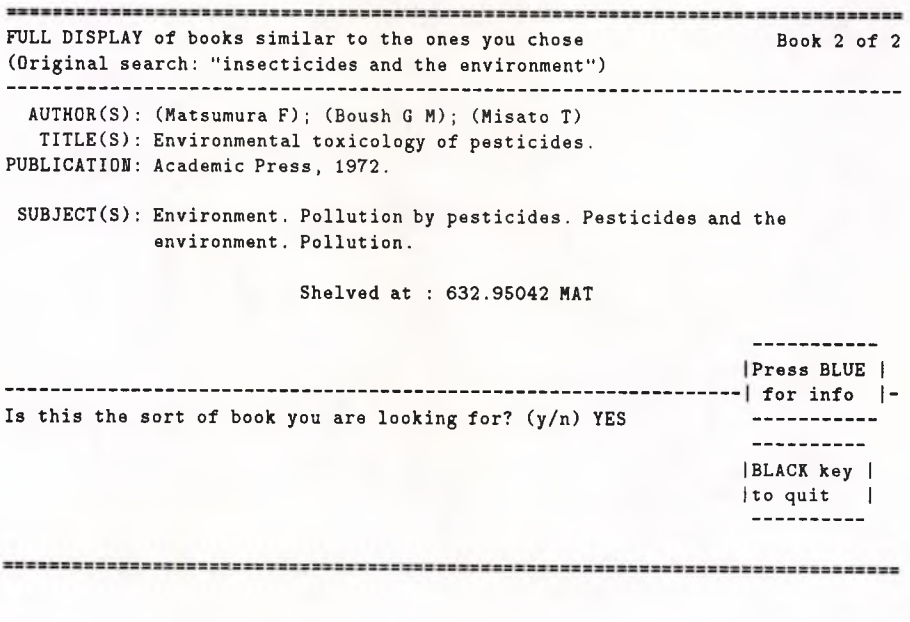


Figure A.15:



The user chooses both the books shown in Figures A.14 and A.15. A second iteration of query expansion results in the list shown in Figures A.16 and A.17.

Figure A.16: Records from second iteration of query expansion

```
LIST OF BOOKS similar to the ones you chose                    Books 1 to 9 of 85
(Original search: "insecticides and the environment")

No. Title                                              Author              Date
1   Organometallic compounds in the environment.       CRAIG P             1986
2   The use and significance of pesticides in the en.. MCEWEN F L          1979
3   Lead pollution : causes and control.               HARRISON R M        1984
4   Lead pollution : causes and control                HARRISON R M        1981
5   Lead and health : the report of a DHSS Working ..  Great Britain. ..  1980
6   Methodological approaches to deriving environmen.. CALABRESE E J      1978
7   Analytical aspects of mercury and other heavy me.. (FREI R W)          1975
8   Environmental pollutants : selected analytical ..  International Co..  1975
9   Further review of certain persistent organochlor.. Great Britain. ..  1969

Down (next)

Type its number to see if a book is relevant

Type Back to return to the books you originally found
Type Options to see other things you can do
                                                        ----------
                                                       |BLACK key |
                                                       |to quit   |
                                                        ----------
```

The initial "options" window (Figure A.9) just gives the user the choice of doing a new search or editing the existing one. Once a book has been chosen there are more choices as shown in Figure A.18.

The VIEW option displays a list of the titles which the user has selected as relevant. Choosing the PRINT option results in Figure A.19.

The EDIT option displays the original search and allows the user to add or delete terms. The screen of FIgure A.20 shows the information snippet for this screen after the BLUE key has been pressed.

The NEW option allows the user to type in a completely new search while allowing the user to see up to their last three searches

Figure A.17:

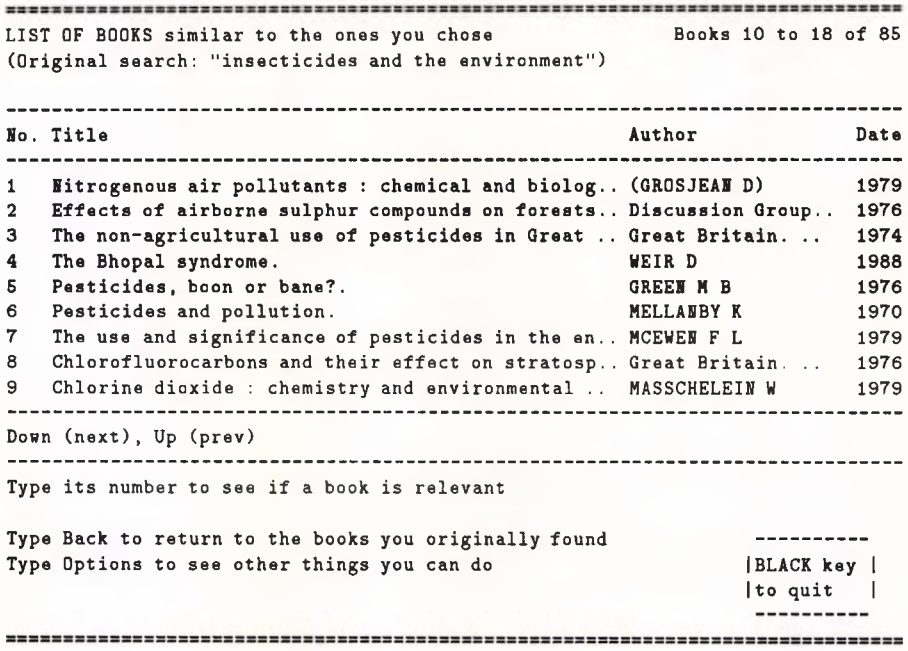


Figure A.18: Options after query expansion

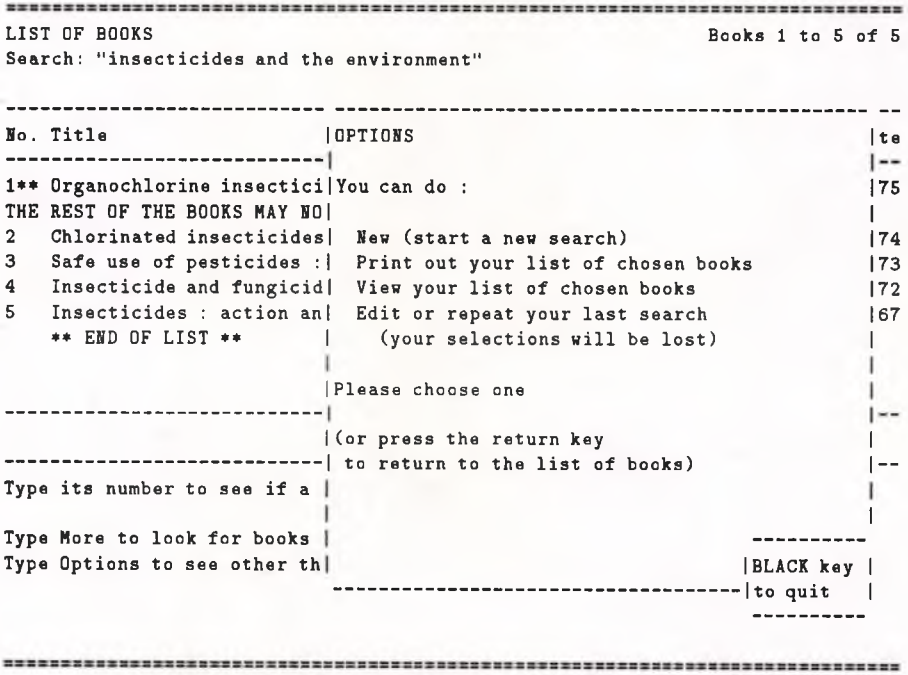


Figure A.19: Choosing the PRINT option

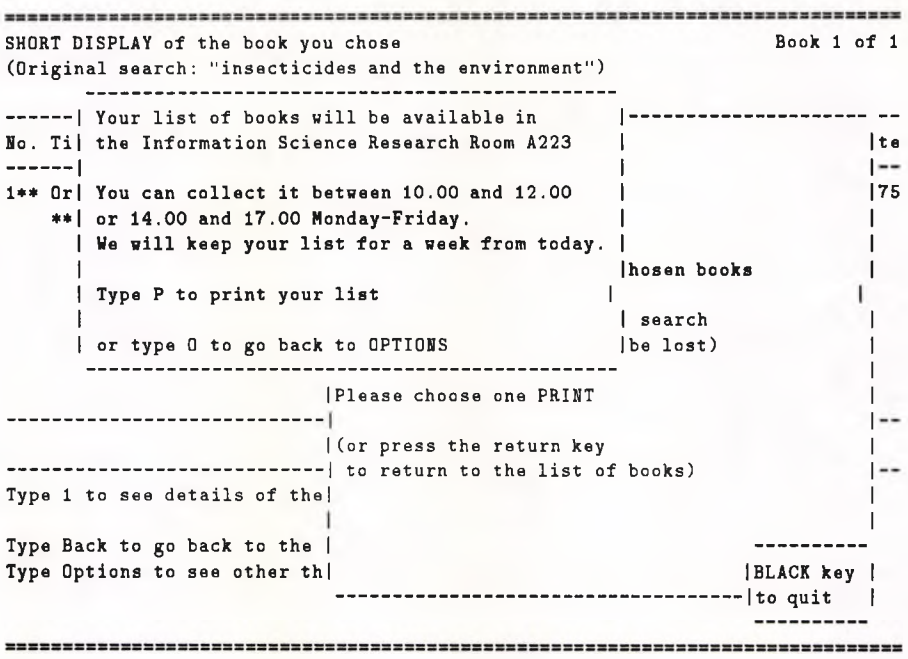


Figure A.20: Editing a search

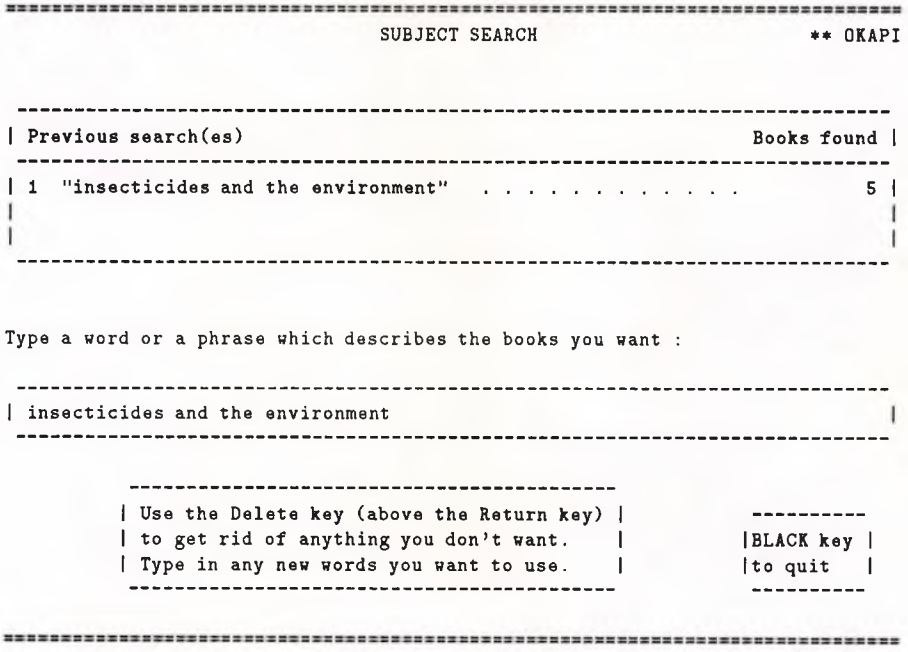


Figure A.21: Full record display: INSPEC

```
FULL DISPLAY                                                    1 of 568
Search: "machine learning in information retrieval"
--------------------------------------------------------------------------------
      TITLE: Inference technique for distributed query processing in a
             partitioned network.
     SOURCE: (Report) 1990)

   HEADINGS: Incomplete knowledge; knowledge based approach; query processing;
             network partitioning; entity-relationship model; correlated
             knowledge; attributes; database content; machine learning
             techniques; rules; data inference; objects; correctness
             criterion; toleration; algebraic tools; distributed database
             system; ship database; model-based knowledge acquisition
             methodology.

   ABSTRACT: "A new knowledge based approach to query processing during
network partitioning has been proposed. The approach is based on the given
query and the use of available domain knowledge to infer inaccessible data.
--------------------------------------------------------------------------------
Down to see the next page
                                                          --------
                                                         |CTRL-Y  |
or press Return to continue                              |to quit |
                                                          --------
```

Figure A.22: Brief record display: INSPEC

```
LIST OF PUBLICATIONS                                          1 to 9 of 568
Search: "machine learning in information retrieval"

--------------------------------------------------------------------------------
No. Title                                                Author         Date
--------------------------------------------------------------------------------
1** Inference technique for distributed query processing..                1990
2   Using type inference and induced rules to provide in.. Chu, W W       1990
3   Structured information management using new techniqu.. Gibb, F        1990
4** Proceedings of the 11th BCS IRSG Research Colloquium..                1989
5   Machine readable information systems of learning opp.. Allred, J      1989
6   Semantics of user interface for image retrieval: pos.. Crehange, M    1989
7   Image progressive retrieval from a videodisk: a mach.. Crehange, M    1989
8   Machine learning for 'intelligent' information retri.. Goker, A       1989
9   Automatic thesaurus construction by machine learning.. Guntzer, U     1989
--------------------------------------------------------------------------------
Down (next)
--------------------------------------------------------------------------------
Type its number to see if a publication is relevant

Type More to look for items similar to the ones you chose         --------
Type Options to see other things you can do                      |CTRL-Y  |
                                                                 |to quit |
                                                                  --------
```

Appendix B

Okapi Transaction Logs

Log text is in Roman, annotations in italic

```
I User special_no Database city.feb91 Date Wed Jun 19 13:12:32 1991
  Program /usr/local/bin/s.city.3.1.3 Description Okapi rf
  Version 3.1.3 Rev date 31 Jan 1991 Pid 12278
```

Identification

```
T 910619131232
```

Timestamp. The E *lines hereafter are elapsed times in seconds from the previous* E *or* T.

```
U s
C ** start
T 910619131239
S SCR7.1
```

SCR7.1 *is the search input screen.*

```
U strucu<LEFT>ture<LEFT>al vibta<LEFT><LEFT>ration<RET>
E +16
L structural (744), vibration (232),
E +0
```

User has typed a search with corrections, taking 16 seconds to type it. The system found 744 postings for "structural" and 232 for "vibration[s]".

```
O NMPW = 10 NGW = 10 NAW = 10
```

System has reported "10 books match your search well." User types d *for Display and looks at a screen of brief records for 27 seconds.*

```
U d
S BRIEFS
1   Sound and structural vibration : radiation, tran.. FAHY F J          1985
2   Structural vibration analysis : modelling, analy.. BEARDS C F        1983
3   Structural vibrations due to wind. Wind tunnel ..  (FLINT A R)       1980
4   Stresses, vibrations, structural integration and.. BEITCH L          1979
5   Formulas for natural frequency and mode shape.     BLEVINS R D       1979
6   Flow-induced vibration                             BLEVINS R D       1977
7   Vibrations of engineering structures.              (BREBBIA C A)     1976
8   Free vibration analysis of beams and shafts.       GORMAN D J        1975
9   Flow-induced structural vibrations : proceedings.. (NAUDASCHER E)    1974
```

```
U 5
E +27
S FULL
  AUTHOR(S): Blevins R D
   TITLE(S): Formulas for natural frequency and mode shape.
PUBLICATION: Van Nostrand Reinhold, c1979.

 SUBJECT(S): Structures. Vibration. Formulae. Structural dynamics.
             Vibration.

                          Shelved at : 624.171 BLE


U y
E +33
```

User has looked at the fifth record in full and answered "y" to the relevance question.

```
S BRIEFS
U 7
E +6
S FULL
  AUTHOR(S): (Brebbia C A)
   TITLE(S): Vibrations of engineering structures.
PUBLICATION: Computational Mechanics Ltd, 1976.

 SUBJECT(S): Structures. Vibration. Structural dynamics. Vibration.

                          Shelved at : 624.171 BRE


U y
E +7
S BRIEFS
U d
E +12
S BRIEFS
1   Structure-borne sound; structural vibrations and.. CREMER L          1973
    ** END OF LIST **

U o
E +9
S OPTIONS
U v
E +12
S VIEWING
S BRIEFS
1** Formulas for natural frequency and mode shape.     BLEVINS R D      1979
2** Vibrations of engineering structures.             (BREBBIA C A)    1976
    ** END OF LIST **
```

Reviewing chosen records

```
U o
E +4
S OPTIONS
U e
E +14
```

User chooses to edit the search, adding the term "aircraft".

```
S SCR7.2 (edit)
U structural vibration aircraft<RET>
E +22
L @0023 (430), structur (2433), vibration (232),
E +0
O NMPW = 0 NGW = 1 NAW = 50
U d
S BRIEFS
1   Introduction to the study of aircraft vibration .. SCANLAN R H       1951
THE REST OF THE BOOKS MAY NOT MATCH YOUR SEARCH VERY WELL
2   Mechanics of flow-induced sound and vibration.     BLAKE W K         1986
3   Mathematics of random phenomena : random vibrati.. KREE P           1986
4   Blast vibration monitoring and control.            DOWDING C H       1985
5   Sound and structural vibration : radiation, tran.. FAHY F J          1985
6   Dynamics and vibration of structures.              FERTIS D G        1984
7   Structural vibration analysis : modelling, analy.. BEARDS C F        1983
8   Vibrational spectra and structure : a series of .. (MARTIN A E)      1980
9   Formulas for natural frequency and mode shape.     BLEVINS R D       1979


U 1
E +37
S FULL
  AUTHOR(S): Scanlan R H; (Rosenbaum R)
   TITLE(S): Introduction to the study of aircraft vibration and flutter.
PUBLICATION: Macmillan, 1951.

                        Shelved at : 629.132362 SCA


U y
E +7
S BRIEFS
U 6
E +6
S FULL
  AUTHOR(S): Fertis D G
   TITLE(S): Dynamics and vibration of structures.
PUBLICATION: Krieger, 1984.

                        Shelved at : 624.171 FER


U ^I
U y
E +9
```

User requests information, which at this point informs user that the system can look for more records similar to the ones which have been chosen.

```
S BRIEFS
U 7
E +7
S FULL
  AUTHOR(S): Beards C F
   TITLE(S): Structural vibration analysis : modelling, analysis and
```

```
            damping of vibrating structures.
            Ellis Horwood series in engineering science.
PUBLICATION: Ellis Horwood, 1983.

 SUBJECT(S): Structures. Vibration. Structural dynamics.

                          Shelved at : 624.171 BEA


U y
E +4
S BRIEFS
U m
E +1
S MORE
E +1
O NMPW = 0 NGW = 58 NAW = 58
```

User has chosen three records and now requests query expansion ("MORE"). System reports that some records have been found. It doesn't say how many unless there are less than a screenful.

```
U d
S BRIEFS
1   Formulas for natural frequency and mode shape.     BLEVINS R D        1979
2   Vibrations of engineering structures.              (BREBBIA C A)      1976
3   Flutter suppression and structural load alleviat.. (Agard. Structu..
4   Stochastic methods in structural dynamics.         (SCHUELLER G I)    1987
5   Schaum's outline of theory and problems of dynam.. TUMA J J           1983
6   Analysis of structures by the force-displacement.. SMOLIRA M          1980
7   Sound and structural vibration : radiation, tran.. FAHY F J           1985
8   Flow-induced vibration                             BLEVINS R D        1977
9   Several techniques for flight flutter testing : .. (Agard. Structur. 1960


U 9
E +21
S FULL
  AUTHOR(S): (Agard. Structures and Materials Panel); (Agard)
   TITLE(S): Several techniques for flight flutter testing : papers
             presented at a special symposium during the eleventh
             meeting of the AGARD Structures and Materials Panel held
             in Athens, Greece... 1960.
             Agard AG56.
PUBLICATION: Agard, 1960.

                          Shelved at : 629.132362 AGA


U y
E +23
S BRIEFS
U d
E +11
S BRIEFS
1   Stresses, vibrations, structural integration and.. BEITCH L           1979
2   Structural dynamics for the practising engineer.   IRVINE H M         1986
3   Dynamics of structures and machinery : problems .. SZULADZINSKI G     1982
```

```
4   Advanced structural dynamics.                    (DONEA J)          1980
5   Structural dynamics : theory and computation.    PAZ M              1980
6   The component element method in dynamics : with .. LEVY S          1976
7   The dynamical behaviour of structures.           WARBURTON G B      1976
8   Dynamics of structures.                          CLOUGH R W         1975
9   Dynamics in engineering structures.              KOLOUSEK V         1973


U 8
E +20
S FULL
  AUTHOR(S): Clough R W; (Penzien J)
   TITLE(S): Dynamics of structures.
PUBLICATION: McGraw-Hill, [1975].

 SUBJECT(S): Structural dynamics

                          Shelved at : 624.171 CLO


U y
E +4
S BRIEFS
U o
E +5
S OPTIONS
U v
E +3
S VIEWING
S BRIEFS
1** Introduction to the study of aircraft vibration .. SCANLAN R H      1951
2** Dynamics and vibration of structures.             FERTIS D G        1984
3** Structural vibration analysis : modelling, analy.. BEARDS C F       1983
4** Several techniques for flight flutter testing : .. (Agard. Structu.. 1960
5** Dynamics of structures.                           CLOUGH R W       1975
    ** END OF LIST **
```

Now reviewing the three records originally chosen and the two chosen from query expansion.

```
U 4
E +10
S FULL
  AUTHOR(S): (Agard. Structures and Materials Panel); (Agard)
   TITLE(S): Several techniques for flight flutter testing : papers
             presented at a special symposium during the eleventh
             meeting of the AGARD Structures and Materials Panel held
             in Athens, Greece... 1960.
             Agard AG56.
PUBLICATION: Agard, 1960.

                          Shelved at : 629.132362 AGA


U <RET>
E +11
S BRIEFS
U ^Y
```

Quit, finished.

```
E +3
C ** Record display summary - this search
  Total unique briefs: 25
  Fulls 5, chosen 5, rejected 0
```

Appendix C

Assessment of Frequent Users' Logs

Frequent User: LIB-A
Logs: 1-11 Total Bytes: 272 K

SUBJECTS:

Control engineering, control systems

English grammar

VAX/VMS, 80286

GENERAL COMMENTS:

-Adapted to and used the More function quickly.
-Used the Display and View fascilities from early on in order to see what he/she had done.
-Tried most functions available in 1st session.
-Later did quite a few Up and Downs when looking at Brief records.
-Initially, seems to search for individual subjects and tries to make links by going through lots of logs rather than adding terms.
Eg: The three queries 'vms', 'os' and 'users guide' (in one session) resulted in the user looking at 441 screens of Brief records of which 55 were looked at twice.
However, in a subsequent more specific search the user looked at 7 records in Brief and 7 records in full. The search strategy did seem to change after this point.

SPECIFICS:

-Very long 1st session.
-Please note details of example mentioned in the general comments.

Frequent User: LIB-B
Logs: 1-16 Total Bytes: 86 K

SUBJECTS:

Mathematics (related queries).

Water analysis, hydrology.

Geology, meterology.

GENERAL COMMENTS:

-Used More twice (logs 9 and 16).
-Seems just to note what he/she has selected and find what he/she wants quickly. No prints - perhaps screen dumps.
-Seems to stick to short and simple/straightforward sessions with only a few things attempted per session.
-Subject interests indicate perhaps a civil engineer student.

SPECIFICS:

Frequent User: LIB-C
Logs: 1-15 Total Bytes: 96 K

SUBJECTS:

AI related topics, conference proceedings, specific languagess, theorem proving.

Interviewing skills, report writing, presentation.

GENERAL COMMENTS:

-Does brief statement searches.
-On whole fairly short sessions.
-Early on in the searches recognised that the ordering of the terms in a query was immaterial.
-Chooses quite a lot of Full records but only displays (no print).
-Has not used More.
-Occasional problem with triggering Information screens.

SPECIFICS:

Frequent User: LIB-D
Logs: 1-5 Total Bytes: 15 K

SUBJECTS:

Mathematics (and related queries).

Management.

Shell (OS).

GENERAL COMMENTS:

-A recent frequent user, hence not as much data available as for the others.

-Started using More early in the searches.

SPECIFICS:

Frequent User: LIB-E
Logs: 1-9 Total Bytes: 35 K

SUBJECTS:

Positive thinking, DIY, learning, house purchase, pottery.

GENERAL COMMENTS:

-Short sessions indicating match/satisfaction on 1st find.
-Later on in the sessions user started to choose from the Brief records list.
-Subjects in queries indicate that the user uses the system in following up references for every day and/or general problems (DIY, etc.).

SPECIFICS:

Frequent User: LIB-F
Logs: 1-12 Total Bytes: 164 K

SUBJECTS:

Electronics, wireless.

Computers, C (programming language).

Islam.

GENERAL COMMENTS:

-In 1st log just seems to be experimenting with the system. Then looks at quite a lot of screens (with Up & Down) of Brief records.
-Has phases of doing long sessions and then short ones.

SPECIFICS:

Frequent User: NET-A
Logs: 2-14 Total Bytes: 112 K

SUBJECTS:

Information and software technology, midi (Atari), security.

GENERAL COMMENTS:

-Tends to search across all databases. Hence, repeats some searches in other databases but also sometimes

in the same one as well (possibly forgets if had performed that search on that particular database).
-When repeating search in another database possibilities for using information about what kind of things this user searched before (log 6).
-Quite few occasions where queries consist of general terms. As a result when get a large list of Brief records, he/she keeps going Down the screen initially just to get an idea of the size of the relevant batch (log 7, log 11). As the user uses all the databases the same terms may be specific enough for the City University Library Database but too general for INSPEC ie. 'information software technology'.
-Used More only once which was in log 5.
-Some problems with triggering More and Logout.
-Some occasions where one title of a reference matches the query string exactly but has not always appeared at the top of the list (for weighting reasons). Perhaps some reordering of the references could be done in such cases of exact match of query to title (maybe for queries of > 2 words).

SPECIFICS:

-Log 11: Looking at a lot of Brief records (only for 1-5 secs). Perhaps not totally aware of how much is available in the database for that query. ie. that system has produced a lot of Brief records. (The total number of Brief records is displayed to the user although the various levels of relevant records are not seen until the user goes throuh the list). Often problem with INSPEC and LISA. With INSPEC, for example, has looked at some 30 screens before proceeding to choose the relevant records.

Frequent User: NET-B
Logs: 1-23 Total Bytes: 315 K Total Online Time: 1 hr 41 mins 41 secs

SUBJECTS:

Conceptual models, expert systems (& related proceedings), air traffic control, temporal reasoning, logic.

GENERAL COMMENTS:

-User seems to be trying out the system in first few sessions.
-First used More in 3rd session.
-Seems to analyse the references and then proceeds to use More. User seems to have reached a compromise when analysing the Brief records list. For example, striking a balance between reading all the records presented and choosing a few sample references and then using More on them.
-When there are too many screens of relevant (as judged by the system) records he/she looks through few screens and then goes back and adds to (via Edit) the original search.
-Sees the number of relevant records before looking at any in Full.
-Generally reading all pages to Full records before choosing them to be relevant and then returning to the first page of the Full record.
-Did seem to use "THE REST OF YOUR RECORDS MAY MATCH...." as a guideline when looking at the Brief records. Sometimes choosing a few more records after More and then Viewing and Printing those selected.
-In early sessions tried different ordering of the words in the queries (which user realised/suspected) and subsequently tried inputing merely the stems of the words (log 4).
-In cases where user is explicitly looking for a workshop, conference proceeding etc. it might be useful to mention this in addition to showing the Brief records list. The user would not have to spend time looking further down the list to see if it existed but would also be aware of the alternatives.
-When not satisfied with records presented sometimes going through a lot of screens seemingly just

browsing and chooses relevant records on the way back up the list. Could this not be cut down by using of 'context links'.

SPECIFICS:

-Log 4: Words used in the queries in this session:

1) es international workshops

2) es their applications

3) es international workshop applications their

1 AND 2) == es

1 OR 2) == query 3)

From the number of records that are retrieved in the above searches perhaps rules like the following could be deduced:

if # recs retrieved by 1) < # recs retrieved by 2) then

retrieved by 3) -the combination of 1) and 2) - is that

retrieved by 1)

At least as far as the most relevant records are concerned.

-Log 6: Tried query "knowledge base verification" and subsequently added "review" and got the same list.

-Log 13: If the "THE REST OF THE..." point is only a few screens down then analyses a few records after this point as well.

-Log 23: Got a lot of records with the same weight and quite a few false drops ie. Business related records on "practical planning" in the query "planning" as opposed to planning in the "artificial intelligence" context.

Frequent User: NET-C

Logs: 1-14 Total Bytes: 449 K Total Online Time: 1 hr 34 mins 47 secs

SUBJECTS:

Solid state physics, circuit analysis.

Fluid dynamics, turbulence.

Hearing research, cochlear (implants).

Cosmic web, NAG.

Rugby.

GENERAL COMMENTS:

-Uses system for quite diverse subject interests.

-Generally uses the City University Library Database but has tried others as well.

-A thorough approach in browsing the Brief records.

-Has done a few author searches.

-In some sessions have a few short queries with not many references produced.

-Repeats some searches (eg: "circuit analysis" and related topics three times using City and Inspec).

-Used More twice, once in log 12 and once in log 13.
-Seems to spend quite a lot of time browsing throught the screens of Brief records list but does not proceed to View, Print or do More.
-On some occasions (eg: Log 1 on "hearing" chose records from bottom of list belonging to other subfields.

SPECIFICS:

-Log 1: To choose relevant records, goes all the way down the list of Brief records and then back up, spending 1-2 secs per screen. Perhaps trying to see the number of relevant records and then viewing the ones that chose. Did 23 Ups and 40 Downs, selected 9 references and did one View, but no Print or More.
-Log 4: Similar search and similar strategy to above log. Additionaly, those records looked at in Full are nearer the bottom of the list. Most of records looked at in Full were not found to be relevant. Some references chosen were the same as those in log 1.
-Logs 1 and 9: Same sort of problem where there are too many screens of Brief records to look through before the "THE REST .... MATCH LESS WELL" point. Seems like kept Down held until got to this point. However, in some queries ("circuit analysis") this point did not exist in the list and the user exited when reaching the End of List. In some cases the queries are a different combination of terms to a previous query. The list produced in these cases is the same but the list is still checked thoroughly.
-Log 11: Could be disregarded.
-After using More in Logs 12 and 13 repeats a previous search and goes through all screens as did previously ("hearing research"). Chose not to use More to home in on request.

Frequent User: NET-D
Logs: 1-25 Total Bytes: 293 K Total Online Time: 1 hr 38 mins 38 secs

SUBJECTS:

Hi-fis, sound.

Bartok, Bach, St. Mathew passion, celesta, concerto, post horn serenode

Unix, OS.

Robot vision.

Environmental geology, chemistry.

GENERAL COMMENTS:

-Searches INSPEC and City Databases.
-Used More in early sessions.
-Views after choosing references and then Prints.
-Quite a few short sessions with resulting short reference list from query. Possibly screen dumps or makes notes of references.
-Seems to read most of the references on the screen when looking at Brief records.
-Has few sessions with quite several "new" searches which are contextually new as well.
-Repeats some searches.

SPECIFICS:

-When typed in "sound" spent 1-6 secs looking at the Brief records' screens called by Up and/or Down (There was no "THE REST OF ..." point) Chose audio related references on whole and then did More. From the list generated after the query expansion (More) the user then chose "hi-fi" and "speaker" related topics. Perhaps some 'savings' could have been made if a link was formed with previous usage of hi-fi.
-Log 2 and 9: Again seemingly looking through a lot of screens (Downs) to find the "THE REST OF..." point. In log 9, however, the user chose a few of the records to be relevant and did More twice for the same query. Most of the ones that the user looked at in Full were generally chosen to be relevant (query: "robot vision" and subsequently "robot vision control motion") .
-Log 13: Would have been possible to detect the most likely context of the query "speaker" from previous searches ie. "loud speakers" as opposed to "Mr. Speaker" and so on. The references could have been presented in the appropriate order of the context identified, for example.
-Logs 14, 15, 16 and 17 can be disregarded. "^S" seems to be repeated not just in start of session but in mid of a session as well (log 16).
-Log 18: As per log 13 could have reordered the Brief reference list for the query "transmission speaker" for the same search.
-Searches on "loud speaker" repeated in form of short sessions.
-Log 21: One very good match of a reference title to a query. Again, perhaps putting such a reference at top of list might be an "improvement".
-Log 25: An author search ("Bartok") on the City database. However, as the system does not enable an author search on the City database the user was presented with a list of biographies of Bartok. Hence, the user then typed in the query "celesta" as this is part of the title of Bartok's books.

With another search of "Bach St. Mathew passion", in this session, the user did not seem to be able to find what he/she was looking for.

Frequent User: NET-E
Logs: 1-18 Total Bytes: 235 K Total Online Time: 1hr 36 mins 44 secs

SUBJECTS:

Object tracking, target tracking, multiple target tracking, (structured) computer vision, recognition cones, dynamic scene analysis, label inspection.

Parallel computer, transputer, Occam, CSP

Tree and pyramid

GENERAL COMMENTS:

-Uses City and Inspec databases.
-Repeats some searches that did on Inspec in City (perhaps to check availability).
-Sometimes does not seem to make note of reference (in same database) and then repeats search with all the Downs etc. (log 17, log 18).
-Does use More.
-Viewed results of long selections.
-Can get quite few false drops with "dynamic scene analysis" in City.db, could be avoided with context information.
-Have some explicit title match with query (log 14, log 17). Perhaps re-order reference list when get such

explicity exact matches between reference title and query. This could avoid going through many Downs in some cases.
-There are links between designing efficient algorithms for parallel computers and Occam, transputer and CSP. Likewise, there are links between time varying image processing and moving objects and dynamic scene analysis. Perhaps these links could have been established and helped in the later queries of the user. (log 9)

SPECIFICS:

-Log 1: 20-55 secs elapsed time when looking at Brief records, possibly thoroughly looking through the list. Used Information early on in this log, then used More and looked back up original list. Some were looked at in Full and some were rejected.
-Log 2: Did a lot of Downs, looking at many Briefs, before reaching the "REST OF YOUR ...MATCH..." point. After going down 11 screens, eventually chose 2 references one of which had the title "Real-time 3D object tracking". This connection could have been made before from the 1st log. Perhaps could have presented list in this log in different order.
-Log 4: Repeated same search twice and chose same references and did More then printed the references (perhaps re-did search as did not print out list the first time round - "dynamic scene analysis").
-Log 6: Repeated same search (in city.db) then narrowed query (by adding "transputer") and got same list of only one relevant record. Did not seem to realise that was not broadening query or not going to get more references by being even more specific. Later searched for "transputer" in log 7.
-Log 8: Similar situation to that in log 2. Searches concerned were on "target tracking" and then "target tracking and dynamic scene analysis".
-Log 9: Similar situation to log 2. Searches were on "label inspection" and "multiple target tracking" where apparently there is a connection between the two. Again, problem of not realising effect of narrowing query by adding terms ie. not getting any more references when that specific.
-Log 14: Explicit title match - "machine intelligence".
-Log 17: Explicit title match - "structured computer vision".

Frequent User: NET-F
Logs: 1-24 Total Bytes: 863 K Total Online Time: 7hrs 31 mins 3 secs

SUBJECTS:

Computer grammar, graph grammar (related to parallelism and transputers mentioned in 3rd paragraph below), algebraic theory.

Computer-aided Software Engineering (CASE), software development, Z (a formal method for s/w dev.), software design methodology, dataflow software.

Strand programing language, Strand parallel programming, Meshix Unix Operating System (par. prog), visual programming, concurrent/parallel programming, parallel programming tools, graphical environments and Occam programming (also related to 1st paragraph of subjects above), virtual reality.

Computer music digital interfaces, ebu interface, interface digital music

GENERAL COMMENTS:

-Has also used system from the library with (own) library card.

-Did More in 1st log and chose quite a few from list after More then did View and Print.
-Searches City.db in 1st few logs but then starts to use Inspec as well.
-Generally tends to try queries in both databases (City and Inspec).
-Hints of thorough approach to query formulation. ie. Puts "computer" in queries even when it is implied (eg: (computer) software design methodologies).
-Usually tries More but asseses the list and sometimes chooses more references but sometimes not.
-Had done search on "computer aided software engineering" then on "CASE and computer aided.....". Would have been useful to make the connection between "CASE" and "computer ... engineering". Would have saved a lot of Downs (Log 3).
-Sessions include extreme examples of user having to look through many screens of Brief records in order to find relevant items which are sometimes further down or even towards end of the list. Most of the list also appears (to the user) to be alphabetical as a lot of references fall into the same weight category or are within the same threshold. (See specifics below for examples).
-Generally not many Prints (perhaps doing screen dumps).

SPECIFICS:

-Log 6: This is a very long session (about 1/4 th of the total logs for this user so far). When gets to the "REST OF YOUR ... MATCH .. LESS WELL" point goes back and checks what he/she has selected. Very thorough approach to looking at list. After the "REST..." point looks at about 10 references in detail then when getting a lot of Ns ("No"s) starts to revert back to assesment from Briefs. Occasionally dips into some records in full then if find they are not relevant continues skipping.

Generally not being able to tell much from Briefs in Inspec.

Getting more Ns past 1st "REST.." line, as expected.

On whole spending 5-10 secs on Brief list screens (not too far off the user's overall pattern of spending around 5-15 secs on average per Brief screen). However, as progress down the list the time is more around 5 secs and appears to be more like alphabetical ordering (due to lot of weights in same weight category - do not have weights written in these older logs).
Eg: Does a More then a View around 1/2 way through log of session (but before end of the 1st query in that session). After View looks at references 1-4 again then repeats references 1-4 once more. Looks at almost all others in chosen list (around 12).

After search of "computer aided software engineering" user did one on "visual programming". Seemed to first choose references which contained "visual" and "graph" in the part of the title that was visible. Establishing this connection from previous queries might have been useful in the ordering of the references. User also concentrates on the related conference proceedings. Not all references containing "visual" were chosen to be relevant but they were at least looked at. If the visible part of the title did not contain the "keywords" of the query it seemed that the reference was less likely to be looked at.

Again gets similar situation where has a lot screens to look through before reaching the first relevance point ("REST...") does not go down all the screens before going back to choose. Seems to go through in a very sequential way.

In going down further in the list chose some of the same references as did in the 1st search in this session. ie. The list starts sometimes to become relevant again further down the list. Choice of references enforce link between the query and CASE and parallel systems.

Generally not too difficult to guess which Full records were found not to be relevant but more difficult to judge if it would definitely be relevant (going by user's previous queries).

After the above, the user then did More. However, before looking through in more detail and choosing the user did 53 Downs (ie. looked at 53 screens) then did a View. Looked at 14 out of 15 references that had chosen previously again and in 4 cases looked at the reference twice in a row. Either forgot place in list or forgot some details and wanted to check.

In conclusion, this session had 113 Downs in total, 2 searches and 2 Views but one Print (suggesting user did not know that would loose list as soon as did another search).

Frequent User: NET-G
Logs: 1-22 Total Online Time: 5hrs 2 mins 51 secs

SUBJECTS:

Database produces perception of end users, end users perception of their information sources, discreet maths, people's perception of their information environment, investigative methods in information systems, professional services to information sector, marketing of business products in CD Roms, marketing of CD Roms, CD Rom marketing and business information, communications and interactions among information users.

Communism and business information, Eastern European business, business opportunities for Eastern Europe,

GENERAL COMMENTS:

-Used More in first search.
-Searches are descriptive, meaningful sub-sentences with prepositions and generally getting short Brief lists.
-Quite varied area of interest.
-Generally using City.db tried Inspec and Lisa.
-Quite few short sessions.
-Tends to read both pages of Full record before answering (Y/N).
-Generally, resumes until end of list.
-Tries most important searches in all three databases.
-Uses Okapi with library card as well.
-In sessions where does not find much information does quite few New searches.
-Generally seems to type in queries that imply wanting a specific paper that answers a question. However this is not usually possible or the case.
-Trying many permutations of searches in different logs.
-Usually choosing some references from <"REST..."> point although sometimes seem irrelevant. Hopefully the user does not feel obliged to choose references that might otherwise not choose because of the ordering.
-Have repeats of some searches.
-New searches are usually not contextually that new.

SPECIFICS:

-Log 7: Looking at all first screenfuls in full (I) most of time but not always. Getting less and less relevant further down the list, again only about three screens seem relevant. This is probably due to the section of Inspec not covering all of user's interest. The More is not so useful here. User did do Print. Similar pattern can also be seen in log 8, except choosing more sparsely and there is a longer list.
-Log 8: User goes back Up screens seeing what chose. Perhaps not aware of Viewing facility. User then continues to choose from where left off (not case in log 7).
-Log 13: Trying to do datastar type query. E.g: 1 AND "ENGLISH" where 1 = "EASTERN EUROPEAN

BUSINESS INFORMATION 1989-1990". Looked through quite few screens then did new search ("EASTERN EUROPE 1990"). Also as 1 found nothing, adding to search that was already very specific was not going to improve the retrieval. User continues to use previous search statements (3 AND ...) though sometimes still finding some relevant information.
-Log 15: After typing a search in Inspec ("East Europe...Business Information") it seems that user changed what he/she was looking for slightly as in analysing the references he/she seemed to find technology-oriented references that were relevant. While looking at this information also found something on "CD Rom for European libraries" which also happened to be relevant (probably to previous queries). User Prints results.
-Log 18: In this Inspec search more references were relevant further down the list (previous connections could have helped). Quite a lot of Downs and quite a few chunks of references with same weights. Views and Prints. (The search concerned is "people's perception of information environment").
-Log 20 and log 21: These seem to be "test" searches.

Frequent User: NET-H
Logs: 1-16 Total Online Time: 1hr 3mins 31 secs

SUBJECTS:

Speckle, electronic speckle pattern interferometry, laser, holography, fourier transforms, fourier speckle, digital filters, aerial elasticity, fringe analysis, contrast enhancement.

80386, Atari in numbers 1040ST, computers, basic computing.

Bartok, Shoenberger, cello, xylophone.

CAD, 386, VS100, MS-DOS.

GENERAL COMMENTS:

-Uses Inspec and City and tried Lisa as well.
-Generally, could tell when record was not relevant within first screen of Full record displays.
-Does View and then More but no FULLs after.
-Generally short sessions.
-Has example of least likely to be relevant record to be looked at last (log 1).
-Does have quite a few searches in which does more than two New searches.
-On whole not choosing reference (Y) after FULL. The number of non-relevant records is greater than the number of relevant records.
-In looking at Briefs, user seems to get disheartened at its length then realises that has looked through but chose little/nothing and therefore dips in FULL at seemingly random order. The references chosen are usually towards the middle or end of the list and therefore not many of them seem to be relevant when found.
-Used More in second search (for first time).

SPECIFICS:

-User repeats log/search on Inspec.
-Log 1: The titles in the references chosen match the subject heading and abstract and descriptors and are generally consistent. User seems to choose references on finding such statements in the title i.e.

holography and interferometry (link to speckle).
-Log 2: Link in second search could have been made after analysing relevant references chosen in the first. However, user chose nothing, only looked at first screen.
-User does some composer searches (Schonberger, Bartok - in log 4). Previously another user had done the same search from the library and also found nothing. Would it have been possible to learn something from this? However, they could have been looking for different things.
-Log 6: Seems to be a repeat of log 1. Perhaps as hadn't printed it. However, this time didn't print it either. Possibly as can do screen dumps. Tried More but had difficulty invoking it.
-Log 7: User did no FULL until last screen (screen 10) then chose 1 and did More. Perhaps thought it would be sufficient enough to converge query rather than going back up all screens. Then user chose 1 and followed it up with More. Possibly user is adding to list one reference/More at a time but does not realise the generalising/widening effect of this as well.
-Log 10: Regarding this log it seems that searching for programming languages results in several false drops. E.g.: if programming language is BASIC then searching for basic computing can result in terms such as introductory being searched for as well hence user is obliged to search for the programming language i.e. BASIC PROGRAMMING or BASIC COMPUTING. We are aware of these type of false drops on the OKAPI system, however need to decide on a way of eliminating this. Anyway, user did not seem to like any of the references available in the library on this search.
-Log 11: User seems to find more relevant records towards bottom of the list. User also uses the first Print in this log. He/she makes link between "digital filters and tomographic reconstruction" (see NET-K) in this log through reference choosing.
-Log 13: Repeated "speckle" again but didn't do much.
-Log 15: User seems to be respectively speaking more thorough, maybe reorganised (context) as a lot of Downs. Link established (from reference = Y) between "fringe analysis" and "Fourier", "digital filters" and others. User tried Print after five New searches. Perhaps did not realise that would be losing all details for those except the last search.
-Log 16: Search on "speckle" again, as per log 1.

Frequent User: NET-I
Logs: 1-19 Total Online Time: 2hrs 18mins 39secs

SUBJECTS:

ellipse, elliptic, conic, non-parametric, line arc, optical triangulation, curve segmentation, range finding.

Vision, image and vision computing.

Tunnel profiling.

Hoffman, Brady, Richards, Robert, Finkelstein, Leyton, Rosen, Rosin, West Germany, poro, Inspec.

GENERAL COMMENTS:

-User seems to find the City and Inspec databases to be most useful.
-Seems quite thorough in search approach.
-Generally attempts More.
-Has some fairly long searches and looks at most first view screens in detail (Logs 5, 6 and 7).
-Some searches are repeated e.g. ellipse (quite a few times) perhaps wants to have updated knowledge or does not record what has previously retrieved.

-When doing author searches, tries various permutations to see if would get more information (log 7).
-Repeats author searches as if to be looking for updates (log 12).
-Seem to be many searches in a session.
-Quite diverse searches in one session (could be a specific author searches are quite frequent).

SPECIFICS:

-Log 1: 7 out of 8 total references were relevant (FULL=Y). Then did View and Print.
-Log 2: User went down to the end of list choosing, then did More and looked to the end of this list but not choosing much. User then did another More looking through all references following by another More and looking through quite a number of the references.
-Log 3: User seems to have quite a few New searches which are actually author searches generally in Inspec. This is the case in this log. Would it be possible or indeed necessary to clarify the author names given in the reference list i.e. when co-authors names appear on the screen instead of the author typed in by the user him or herself. Perhaps have it so that the author's name himself or herself is actually on the screen.
-Log 4: "Optical triangulation" also used by another user.
-Log 5: User choosing until end of list then doing More (probably finding good examples for context, this is also the case in log 6). There are many new searches (must contextually be new as well) in this log. User does generally attempt to get to the end of list but if the list seems to be getting too long then stops. Common pattern throughout all users searches.
-Log 7: User had intended to do an author search by typing in the word "west". (Could have detected this from previous searches). However, in this case got all other subject matters as well regarding to west as a direction etc.
-Perhaps ought to have links between words like "ellipse" and "elliptic" (log 7).
-Log 7, 13: Also quite a few Downs in this session (image vision and computing) before user seems to find something relevant.
-Log 11: Repeats search again in thorough way looking at FULL as go along.
-Log 13: Looking for "Lowe" as author but it got stem to "Low". Hence the user found lots of irrelevant information i.e. false drops.
-Log 14: User searches for same author "Roberts" as a previous user (user NET-F). Seemingly just wanted to see if those authors had publications as did not identify reference to be relevant. On the search in the City database there were quite few false drops.

Frequent User: NET-J
Logs: 1-9 Total Online Time: 1hr 38mins 51secs

SUBJECT:

Extensible natural language systems, learning unknown words, processing novel natural language, a general explanation-based learning mechanism narrative, narrative understanding, unknown natural language words, marker passing, passing spreading activation, extended lexicon, word acquisition, lexicon acquisition, relational models of the lexicon, language learning, natural language learning, inference and natural language understanding systems, English dictionary.

GENERAL COMMENTS:

-Only 9 logs (so far to date) but quite a bit of data.
-User tried Inspec, Lisa.

-Generally quite descriptive queries.
-Did More in log 3.
-Is trying many approaches with different descriptions in order what he/she wants (possibly adapting search behaviour according to the reference list shown).
-Again a lot of Downs (after More). Possibly, as was the case with other users, could have been less with a re-ordering of the reference list.
-User does not seem to like long lists and therefore tries to describe queries so that references 1 to 10 roughly are relevant and list is not too long to look at in full either.
-Has Viewed chosen references but no Print (log 4).
-As user is trying many descriptions for some of these queries, this seems to imply that the user in interested in an inter-disciplinary area.
-User spends considerable (10 to 50 seconds) time looking at Briefs.
-Some searches appear to be paper or coursework titles. User looks at them in FULL carefully before answering whether it is relevant or not. It is almost as if the user would rather answer N than Y if its not a very good hit.
-Users general area of interest seem to be natural language (loosely), word acquisition, unknown word parsing.

SPECIFICS:

-Log 3: User seems to know what he/she wants generally but not much appears in the Brief lists of references and there are also quite a few false drops. False drops can be identified often from the title, here. E.g. Title may not directly be relevant although can guess relevance through equivalence grammar "unknown natural language words". Again it is possible to guess those more likely to be chosen.
-Log 4: The search "marker passing" performed much better (chose 3 out of 4 relevant records).
-Log 5: Found nothing on "marker passing and spreading activation".
-Log 6: In the search "language learning" after 29 Downs looked at a reference. Then user went to the end of the list without choosing any relevant reference. The implication was that the user had meant learning in the artificial intelligence sense and that these references were not until the bottom of the list. Possibly, the re-ordering of the list based on user's previous searches could help.
-Log 7: A long search (natural understanding systems). If the user could not get exact matches then he/she looked for applications of the same issue in different areas ("knowledge-based understanding of radiology tests" in title). User looks at about one full record per screen of Brief list of references. After this More did not seem relevant. Some searches were repeated (i.e. word acquisition). This was possibly because user did many new searches and might not have necessarily remembered all the details of the searches. User seems in this session and generally to be quite thorough in approaching going through the Brief list. This could be because the user does not have many directly relevant references to topic of interest and is trying to work round that. Again towards the end of the session, the user starts to seem to find more relevant references again. As per other users, could the re-ordering of this list not have been more helpful to the user. Session consists of comprehensive searches. There are not many threshold points (REST OF REFERENCES...). This implies that quite a few references have the same weights. User did quite a variety (a comprehensive use) of functions i.e. More: 6, New: 4, View: 2, Print: 1. The proportionately less number of Views and Prints seem to imply that some relevant records were not worth recording possibly because had already obtained these before.

Frequent User: NET-K
Logs: 4-22 Total Online Time: 2hrs 47mins 17secs

SUBJECTS:

Flow imaging process tomography, process tomography, linear stepper motors, nuclear reactor control drives, eddy currents, coin detection by eddy currents, magnetic field modelling, electro-magnetic fields, crack detection by eddy currents (EC), non-destructive testing by EC, monocular vision, reconstruction algorithm, reconstruction algorithm in capacitance tomography.

GENERAL COMMENT:

-First 3 sessions are very short. User seems to log out without doing much - perhaps getting the feel for the system only.
-Second log could possibly be deleted as there does not seem to be anything in it.
-User uses both Inspec and City databases.
-User's general interest seem to lie within physics, however, it seems quite diverse within that subject field itself.
-Used More once only in 22 searches and this was in the 16th log.
-Quite a few logs are missing which imply that user logged off as soon as he or she logged on without doing anything. This could possibly be due to some logging in problems.
-User seems to leave search at "observing Brief reference list" on the whole.
-If could have linked "current", "magnetic field/current", "eddy current" then maybe the Brief list in log 14 could have been re-ordered (however the whole list did fit on two screens anyway).
-User does tend to put in longer (more than 3 words) in queries, therefore generally does not get long Brief lists.
-User has not done any Prints (to date). However, could possibly be doing screen dumps.
-Possibilities for guiding merge terms with context term information (log 16).
-As is the case with other users, the whole log 17 is based on links which could have been previously made from the searches. In this particular log, the user tried the searches in the cell but found nothing relevant.

SPECIFICS:

-Log 8: User went down 7 screens before reaching any threshold point ("REST OF REFERENCES MAY NOT MATCH YOUR SEARCH VERY WELL"). User then went to the end of the list and typed in a new search (contextually an edited search).
(1) Linear stepper motors.
(2) Linear stepper motors, nuclear reactor control systems.
This was quite a specific search, there was only one very good match and this was found not to be relevant.
(3) Nuclear reactor control drives.
In this case, chose only one relevant record and Viewed it.
After these searches, there seemed to be some problems with the network.
-Log 10: A specific title (electro-magnetism) searched again. Re-ordering of references might have helped. User chose two references but did no print-outs.
-Log 16: Although "electric" was not in any search until this point, this was strongly implied by the context of the query and the other terms used previously in other searches. Therefore, could have helped in the re-organisation of the Brief list, as there were 24 screenfuls of records with the same weight (97). User seemed to choose records in those 24 screens by matching "reconstruction" (which was in the query) to part of the title of the reference that could be seen. This is a case where, by analysing the references that have been chosen to be relevant, links between the terms "image", "tomography" and "reconstruction" could have been made. Having made such links by title and previous search statements analyses could be made in order to predict which way the likely references to be chosen. It seems that predicting those

which the user is not likely to look at nor find relevant would be easier.
-Log 18: This session includes another link which could have been made ("image reconstruction"), tried on City.db.
-Log 19: "Monocular vision" is more loosely connected with "image" and "reconstruction" in these searches. This can be deducted from the references that are chosen to be relevant by the user. This session also includes the case where the user has typed in a specific query and not found any or much relevant documents. Subsequently, the user tries another very specific query also related to the previous one which also results in no relevant references being identified. E.g: the first query, "coin detection by eddy current" found little and the second query "crack detection by eddy current" had similar results. If the first query did not find much information it is unlikely that the second (which is just as or more specific) would find more. User may not have realised this.
Note: "non-destructive testing" does actually occur in one of the abstracts in the references chosen. These terms are also searched for in subsequent searches by the user.

Frequent User: NET-L
Logs: 3-15 Total Online Time: 1hr 47mins 23secs

SUBJECTS:

Optical triangulation, profiling, range, laser range, speckle, photogrammetry, machine vision range, robotics, sub-pixel, camera calibration, photographic film deformation, photographyton reproduction, analytical geomorphology, laser scattering particle, particle-size shape scattering classification, laser scattering (airborne), laser range distance measurement, laser electro-optics system, rapid topographical mapping, monchaud's bickel, tunnel profiling, (prism), optics, prism non-linearity, linearity triangulation, optical triangulation line, (true colour) camera.

GENERAL COMMENTS:

-Uses all three databases. The City database seems more useful and found little in Lisa. Inspec does not seem to cover users' interests that well on some searches.
-There are quite a lot of New searches but these are usually contextually related.
-Although the user does not often get long Brief lists when this does happen the user goes down the screens seeing how many records there are or trying to see where the threshold (REST OF REFERENCES...) points are. User then sometimes proceeds to chose the references on the way back Up the screens.
-User has not used More (in 15 logs to date).
-User has some searches which are repeated.
-Like some other (user RS2) users, this user also seems to be looking for very specific references sometimes, hence are getting many searches with little finds.

SPECIFICS:

-Log 5: Quite methodical in this session, got more references than on previous searches. User looks at most of the references in full (only three screens) and about 50% of them are relevant. On another search in this session, user goes through the whole of the Briefs list and finds no "useful" references. Maybe this is partly due to the fact that Inspec does not cover the user's field of interest and not necessarily the performance of the system (similar case in log 7 where only found 1 relevant record).
-Log 9: There is quite a bad false drop ("1990"). This resulted in the retrieval based on "1990" occurring in the date that the reference was published. Hence the user was not looking at the seemingly more

relevant references in this search but perhaps the more "curious" ones.
This was a long session with quite a few New searches (more than 5). User seems to look through all Briefs until the end of list. User chose quite a few relevant references and did View, Print but no More.
-Log 10: User seemed to think that capitals made a difference in the query. (Tried query in both upper and lower case). Session includes many searches where the user is getting one reference in total if lucky.
-Logs 11, 12, 13, 14: Not many relevant records found (on "prism", "optics", "triangulation", etc.) in Inspec.

Frequent User: NET-M
Logs: 1-22 Total Online Time: 1hr 53mins 29secs

SUBJECTS:

Mach, dataflow, neural networks, dataflow networks, (optical) computing transistor, network VLSI computing, (fuzzy logic/computer, ISIS page tables (inverted), sprite, AFS and other file systems, object architecture file systems/eiffel, AIX, inverted page tables, memory allocation, storage allocation, cloud (object), clouds operating system, choices operating system, object operating system, Hoare, sequential process, networking computer, topologies, garbage collection, TOPSY, communication topologies, communication networks.

Pascal, GNU, amoeba

GENERAL COMMENTS:

-Uses Inspec and City databases and has also tried Lisa.
-User's own interest area seems quite diverse.
-User tends to look at all references in full that appear on the first screen of the Brief lists (e.g. log 5).
-User does use More occasionally and seems to find it useful.
-User tends to View and Print.
-There are quite a few short sessions of the form:
1) a (got too many references)
2) a + b (got some references)
3) a + b + c (got less references)
where a, b and c are search terms. However, user did not seem to realise that if is in a search 1, term in "a" was the most important issue to be looking for then adding terms b and c would not necessarily have helped and possibly diversified the reference list (log 6 and 7).
-The user seems to read the information about the number of records that have been found as he/she proceeds to edit the queries at earlier stages narrowing it down if it is not particular enough.
-User has tried.some author searches (e.g. Hoare).
-When the user does More if the "useful" references do not appear in the first screen or so then the user does not continue.
-Generally, if the user gets the information he/she wanted in the first few observations then he/she is quite satisfied prior to doing More.
-User typed in "1" meaning the query 1 in order to repeat this search. This implies that the user has done some on-line searching before. (log 2) User also types in "question mark" after wanting some information/help.
-User has also done View and Print.

SPECIFICS:

-Log 1: A short session, a one-word search. Looks like user was testing.
-Logs 2 and 3: User did search on "dataflow" in Inspec then "dataflow networks" and did get some relevant records. The user then did these searches on City and only got one record which was irrelevant ("transborder dataflows...Poland"). Possibly the user could have been warned that this database was not relevant to their subject.
-Actually the user has tried "dataflow" searches in all databases.
-Log 5: User proceeded until the end of Brief list. Tried More for the first time and seemed to find it useful. Note: "GNU" has also been used by some other frequent users.
-Log 7: User spends around 10 to 20 seconds per screen until reaching a threshold point (REST OF...). After this point when going Down and back Up the user spends zero to 2 seconds per screen. When the user realises that there are quite a few more references, then exits. Usually looking for very specific topics. Hence many New searches and short sessions. Often the user does choose relevant records but does not follow it up with More.
-Log 12: User does second More in this session. Does many Prints. It might be possible to map the original terms and the merge terms and see if can make any contextual links in the areas where one might not necessarily know that much about.
-Log 14: This session contains a lot of Downs. This is possibly because the user wanted to get some up-to-date knowledge on a search previously performed. The user looked at no records in Full.
e.g. 1) "memory allocation" (then user did More) therefore the merged terms would possibly include "storage".
2) "storage allocation". This should link/strengthen these terms concerned.
-Log 16: User did search on "cloud" and did not choose anything to be relevant. Subsequently, the user followed this query up with "cloud object". It might have been possible to "sift" references on "object" and "cloud" to the top from the previous search.
-Log 20: User did search on "object operating systems" and chose references which included some terms used previously. i.e. mach. User also tried to use abbreviations such as "OS" for "operating systems".
-"Topsy" was also used by another frequent user. There were many New searches in this session. User also tries alternatives for the same word i.e. "network" and "networking". This does not however make a difference when stemming. Quite a lot of searches in this session just consisted of looking at one or two references and not finding them relevant and exiting back to the main menu.
-Log 22: Again chose quite a few references but did not do More.

GENERAL POINTS AND SUGGESTIONS

-Suggestion: When sessions are long, it might be worth having some summary information on the More and View data information before the user proceeds to do a new or edited search.

-It would seem that not more than three general subject areas emerged per user (usually it is one or two). Therefore, could have about two profiles for a user.

-Suggestion: If new terms are added to the original search and the same relevant records are given in the list then perhaps it might be possible to make some adaptations (such as changing the weights, as the weights of the original terms are probably still higher) to meet the new circumstance.

-Suggestion: If in the relevant records list, a title string matches the query string exactly (no more or no less) then should this be put at the top of the list? (This might be too much effort though). (Users: NET-B/9, NET-A/13, NET-K/10).

-Suggestion: If in the relevant records list one of the references includes a word/phrase that has been used

by the user frequently but not in this query, it might be worth pointing this reference out to the user. (User: NET-B/15 - "temporal reasoning" implied "temporal reasoning and air traffic control")

-Suggestion: Bring in co-author's work with "good" weight or build up an author profile that the user is interested in. (Users NET-B, NET-H).

-When doing a search in Inspec, some of the references could be at City. Therefore, might it be useful to either enable the user to automatically repeat this search or suggest that those references exist in the library without the user having to log in and log out again and so on.(User: NET-F/6).

-Suggestion: When repeating a search in another database it might be possible to make use of what kind of things were looked at by the user in previous searches/sessions. Hence learning might be possible not just for a particular database but across databases for that particular user. It might well be that certain databases work well for certain users and certain interest topics but it might not be possible to generalise across the board.

-Most frequent users use Inspec. The next frequently used databases are that of City and Lisa. Note: This is probably due to the nature of the network users and the way in which the system has been set up.

-Statistics on the number of relevant records out of the total number of records viewed might be useful. Also statistics for the number of relevant records chosen after a More in relation to the total number of relevant records or the total number of records might also be useful.

-Suggestion: If the search concerned can be identified to be an author search then his or her name should appear in the reference list even if he or she is a co-author and therefore is not necessarily the first author in the publication.

-Linking this point to one previously made. When users have to look through many screens which constitute references with the same weight, then the ordering needs improving. Part of the ordering is that of the alphabetical list with an author. It seems that a more contextual approach rather than an alphabetical one would be more appropriate.

-When expanding a query the weights of the merged terms in relation to the original terms have been the issue of other research.

-Suggestion: User can be prompted if he/she has chosen quite a few relevant references and has not done a More (user: NET-C/1).

-User could be prompted about Viewing facility instead of doing many Ups and Downs (user: NET-C/1).

-Example of the importance of context:
could have foreseen that the context of "speaker" was in "hi-fis" and re-ordered the list presented to the user wasting less time and space (NET-D/13). Other examples include "transmission speaker" (NET-D/18), "transputer and object tracking" (NET-E/2), "target tracking and dynamics scene analysis" (NET-E/8).

-Suggestion: It might be useful to start identifying when a user is becoming a frequent user and perhaps add this to list automatically or start doing "learning" for user automatically.

-Users do not always realise that adding to the original query, making it more specific when the original

query has not retrieved much, will not necessarily help their retrieval aims (NET-E/6, NET-F/7).

-Suggestion: Indicating words that co-occur or indeed are phrases more strongly in a semantic network that might be implemented (e.g. dynamic scene analysis). This should help in avoiding false drops especially on the City database.

-Example of context: it would have been useful if link could have been made between "computer-aided software engineering" and "CASE" - an abbreviation. (NET-F/3)

-Suggestion: It seems that it is difficult to tell much from the Briefs in Inspec. More information seems necessary for the decision-making of the relevance of a reference. Perhaps having two as opposed to one line of information per reference might be useful (NET-F/6).

-On the whole it is not too difficult to guess which references will definitely not be looked at or will be chosen to be non-relevant. It is more difficult to guess which references will definitely be looked at and chosen to be relevant for the user in that session or query (NET-F/6 and others).

-Suggestion: Some links might be useful not within a user's sessions but across board or groups of users. E.g. "visual programming" and "concurrent" (NET-F/7) could be useful to others in the field.

-On the whole frequent network users' sessions are not long because they return to use the system more. However, their session listings might seem very long as often they look at many screenfuls of Briefs.

-The general impression is that New and Edit facilities are used as intended to be syntactically and semantically (see previous report for more details).

-Example of context: "graphical programming" and "concurrent" again links on these issues could have been useful. There are definite vocabulary patterns emerging in users with these kind of interest areas (NET-F/9). It might be worth suggesting some of these terms to other users.

-When users choose references which are quite further down the Briefs list, it is quite possible that this is due to the reference being related to a previous query (NET-F/9).

-Suggestion: When choosing references from similar searches users tend to sometimes pick up the same relevant references. Could this not be identified and automatically catered for in order to help save time and avoid double selection. Alternatively, it could be used as an indication of a typical very good positive example for selection (in the learning mechanism).

-Suggestion: Perhaps in the learning mechanism it might be useful to use the combination of information of what is viewable from the title and context information.

-Suggestion: With frequent users when we know that they are insistent on a particular subject area, perhaps the search could be done automatically for them. This would also be useful if they are repeating the same set on another database simply to see what is available.

-There seems to have been an increase in the amount of searches in July.

-Suggestion: In a semantic network there could be various types of relationships i.e. parallel, narrow, broad, abbreviation etc.

-Some statistics on whether users tend to be very thorough in going through the Brief list, especially when are not necessarily spoilt for choice might be useful (e.g. NET-F, NET-J) when looking for very specific topics.

-Suggestion: "Learning" (machine learning) could be used to obtain either a good list of merge terms or optimise the merge terms that are produced after a More. This could be useful in the direction of the query expansion with the context/depth information from previous searches (NET-K/16). In cases where a More has not been done these terms could nevertheless be generated and some form of automatic learning be done from there. However, there might be some optimum points after which doing any More functions does not necessarily help in the direction of the search required by the user.

-Suggestion: Learn from these frequent users and apply either to these individual users or across the board to all users.

-Suggestion: If user goes through the whole or a substantial amount of the Briefs list and finds nothing then this information could/should be used somehow. This could help in identifying if they are looking for very specific references or simply that the database does not search their needs or that actually do know all the references that have been presented to them. (NET-L/10, NET-H/2).

-Suggestion: If there are any numerals in queries, then the user perhaps ought to be informed about the way in which this search would be performed (NET-G, NET-M).

-Suggestion: If user does same search in one database and does part of that search in another and if references are obviously unrelated then comment this to the user either prior or during the display of the references (NET-M/2, NET-M/3). This can happen when the databases provided were meant for specific fields and not others that might share some similar terminology.

-Suggestion: Making connections between the merged terms and the original terms is useful. However, also useful would be making links between the merged terms and previous queries and their terms.

-Suggestion: Rather than including just one word/term sometimes storing clusters might also be useful, i.e. including "a" if "b" and "c" also co-exist.

-Suggestion: When the user finds a reference to be non-relevant then the system has identified this to be one of the more likely to be relevant references then perhaps ought to deal with this. The user could be asked as to why he or she found it to be non-relevant for example if he or she has already obtained this reference (NET-M/20-TOPSY).

-New links between documents are formed when a query is repeated. Relevant documents can be increased and new terms identified whilst others are strengthened.

-Suggestion: Regarding the alphabetical ordering within the same weight of authors; it might be more useful if one of the co-authors had been identified to be relevant to a query to actually point this out in the list (even if the co-author is not necessarily the first author in the publication). Also the alphabetic list of all authors within the same weight category is not necessarily helpful for the users. A more context based re-ordering might be better.

-Suggestion: Author links in a semantic network might be useful.

-Suggestion: Links between words like "ellipse" and "elliptic" would also help in identifying context for users.

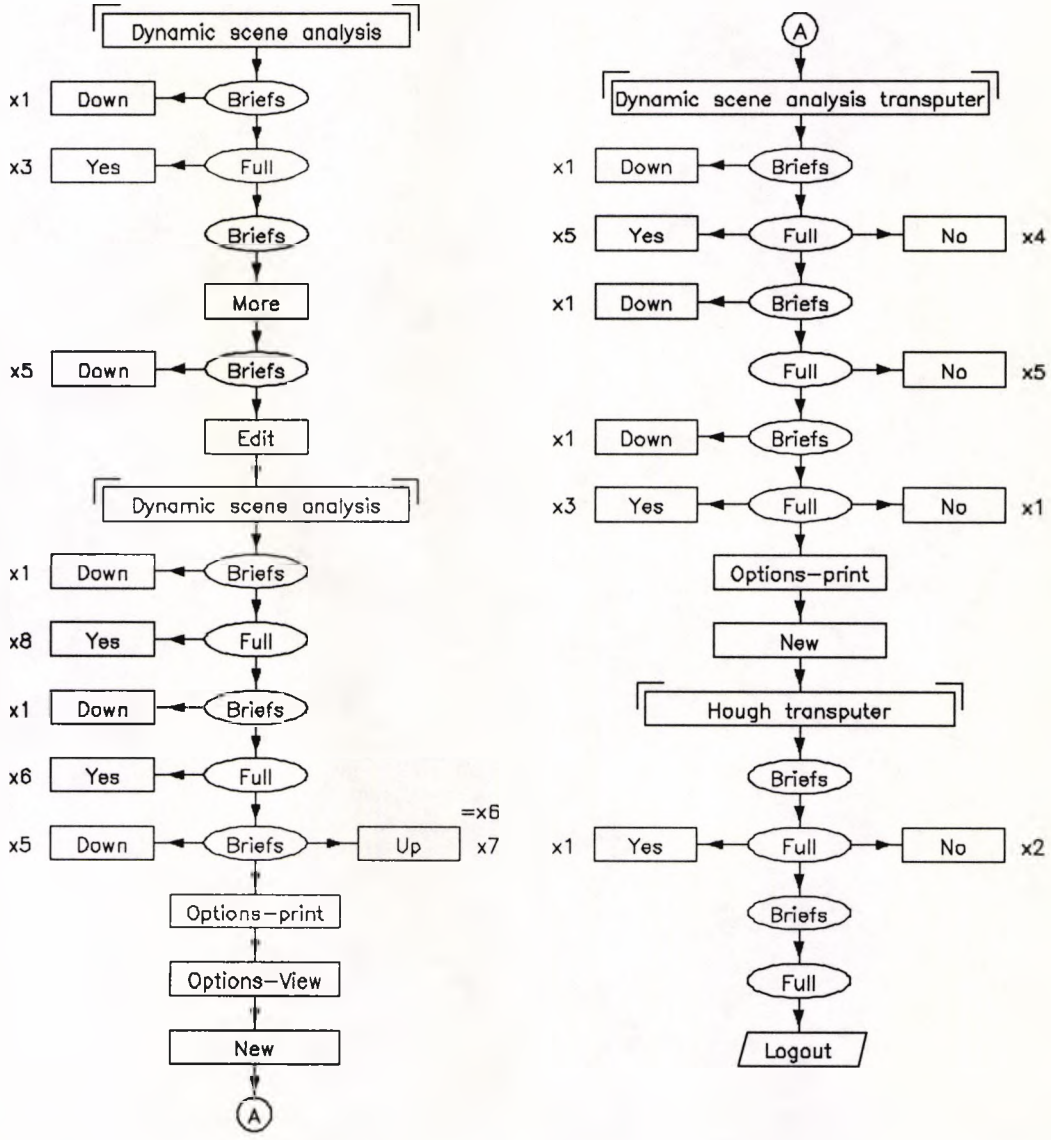
-Suggestion: If user is looking for a particular author or group of authors and repeats search on these then perhaps what he or she intends is to find more up-to-date information published by these authors. It might be worth doing some of these searches automatically or prompting the users to do searches on these authors.

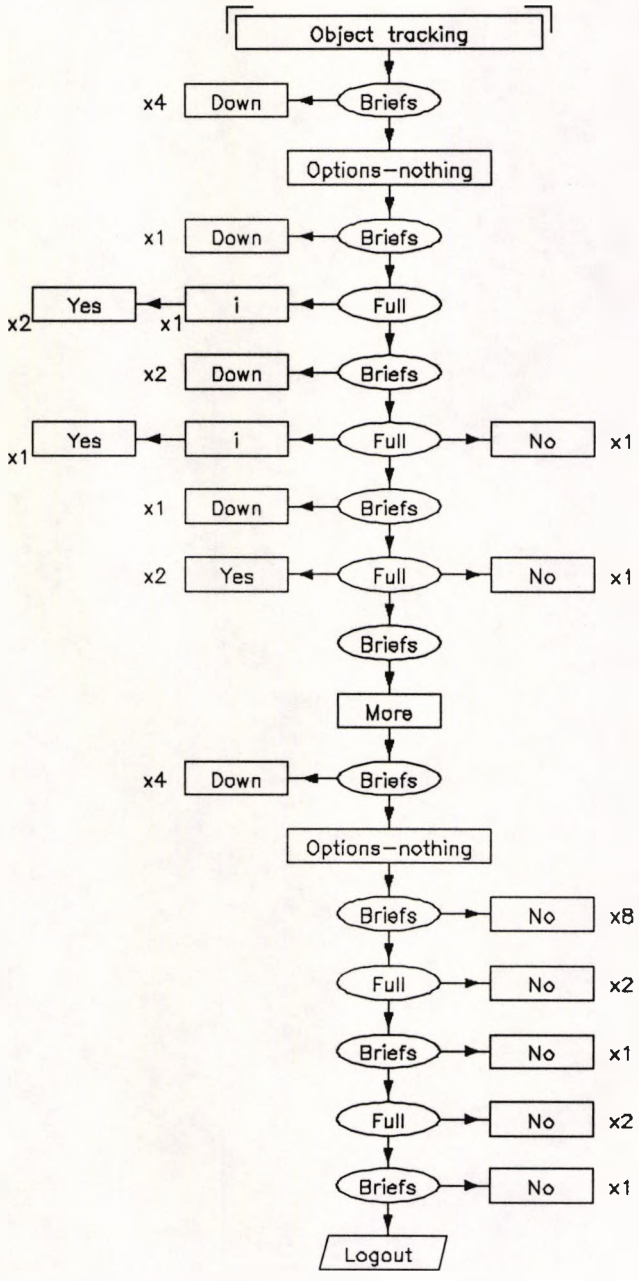
-Suggestion: If user types author surname and initials and later searches just on a surname then perhaps the system should be able to detect that it is an author that is intended in the search and not a subject. Either point or ask this to the user or sift those references to the top of the list (NET-I/7). However, before proceeding with this, it would be useful to find out exactly how many times this does actually happen.

Appendix D

Frequent User Session Summary Diagram - An Example

The following is a summary diagram for the functions performed by the user and the corresponding system feedback.


Dynamic scene analysis
x1
Down
Briefs
x3
Yes
Full
Briefs
More
x5
Down
Briefs
Edit
Dynamic scene analysis
x1
Down
Briefs
x8
Yes
Full
x1
Down
Briefs
x6
Yes
Full
=x6
x5
Down
Briefs
Up
x7
Options–print
Options–View
New
A
A
Dynamic scene analysis transputer
x1
Down
Briefs
x5
Yes
Full
No
x4
x1
Down
Briefs
Full
No
x5
x1
Down
Briefs
x3
Yes
Full
No
x1
Options–print
New
Hough transputer
Briefs
x1
Yes
Full
No
x2
Briefs
Full
Logout


Object tracking
x4
Down
Briefs
Options—nothing
x1
Down
Briefs
x2
Yes
x1
i
Full
x2
Down
Briefs
x1
Yes
i
Full
No
x1
x1
Down
Briefs
x2
Yes
Full
No
x1
Briefs
More
x4
Down
Briefs
Options—nothing
Briefs
No
x8
Full
No
x2
Briefs
No
x1
Full
No
x2
Briefs
No
x1
Logout

Appendix E

Term Details for Frequent Users - An Example

Term	Freq	Postings (No.)
@0007	1	2514
@0102	10	1584
@0125	1	450
bach	2	68
bartok	1	13
cd	2	8
celesta	1	8
chemistri	1	2541
concerto	1	322
control	4	13569
design	1	3160
earth	4	315
hifi	2	5
loudspeaker	2	3, 16
management	2	6138
matthew	2	17
motion	4	1193, 1787
natur	1	552
operat system	1	147
parallel	1	4507
passion	2	28
posthorn	1	1
robot	6	2278, 3374
scienc	4	7117
scientist	3	441
serenad	1	30
sound	2	259
speaker	2	40
st	1	247
transmission	1	343
unix	1	85

Table 1: *Table showing the terms used by a frequent user, their frequence and number of postings.*

Appendix F

Term Usage - An Example

Term	Post.	Freq	User-id
applied	1347	2	010093707, 010139039
appraisal	226	1	010136852
approach	1584	11	010013341, 010120468 010092782, 010093707
acquisition	2	2	010113059, 010116334
architectur	435, 430	4	010119759, 010169390 010195635
armi	54	1	010093566
art	1780, 1912	9	010049055, 010019116 010124890, 010195635 010096583
arthur	53	1	010098037
arthurian	5	2	010098037
artifical	8	1	010076652
artificial	321, 326	3	010195932, 010018704 010093707
as	880	2	010049055, 010099589
aspect	3856	1	010115112
assemblei	2	1	010195635
assembli	154	2	010195635
association	787, 827	3	010050376, 010115112 010138601
astigmatist	1	1	010131754
atari	9	1	010125830
athletic	9	1	010060862
atla	305	2	010092543
atomic	372	1	010101344
attitud	407	2	010049055, 010079565
attribution	15	2	010120583
audienc	72	1	010009117
audio	169	1	010117043
audiologi	32	1	010017516
auditori	42	2	010151976
author	212	3	010049055, 010094820 010122365
authoriti	334	1	010001296
autobiografi	216	4	010014349
automata	35	1	010076652
availabiliti	27	1	010076462
avionic	48	1	010094077
avoidenc	20	1	010094077

***Table 1**: Table showing terms used, their frequency of usage and the corresponding users.*

Appendix G

Explanation-based Learning - An Example

Eg1: The following is an example of explanation-based generalisation (EBG), in Prolog (Kedar-Cabelli and McCarthy, 1987). It involves learning about "suicide" given a certain number of facts and theory about what might lead to it.

Goal Concept:

kill (john, john) is a target goal but a possible generalised goal would be kill (X, Y).
i.e. "learn something about john suiciding" or "somebody killing somebody."

Domain Theory:

kill (A,B) :- hate(A,B), possess(A,C), weapon(C).
(A kills B if A hates B and A also happens to posses a weapon).
hate (W,W) :- depressed(W).
possess (U,V):- buy(U,V).
weapon (Z):- gun(Z).

Observations/Facts:

depressed (john).
buy (john,gun).
gun (gun1).

Learned Concept:

kill (X,X):- depressed (X), buy (X,C), gun(C).

The concept is formed by finding a proof for the goal from the domain theory(rules) and facts. The proof found is then generalised to contain the class of all examples having the same proof of concept membership. Hence, by applying the facts to the domain theory the following specific proof is derived:

kill (john, john) because
hates (john,john) - john hates himself because
john is depressed - depressed (john).
and
possess (john, gun1) - john possesses gun1 because
john bought gun1 - buy (john, gun1).
and
weapon (gun1) - gun1 is a weapon because gun1 is a gun
(which is a weapon) - gun (gun1).

From this proof, the above mentioned generalised proof is obtained:

- somebody will commit suicide if they are depressed and have bought a gun.

Eg2: Following is a Prolog version of an example given by Braverman (Braverman and Russell, 1988) where the goal concept is *aocmrs* meaning advertised on musical radio

Goal Concept: aocmrs(X).

Domain Theory:

aocmrs(X):- luxury_item(X), european(X).
sports_car(C):- luxury_item(C).
british(B):- european(B).

Observation/Facts/Training instances:

sports_car(Jaguar).
british(Jaguar).
aocmrs(Jaguar).

The explanation of this instance (using EBL) would be that:

A Jaguar is advertised on musical radio because
1) It is a luxury_item - as it is a sports' car and
2) It is european - as it is a British car.

Thus, ignoring issues of operationality the following five can be the learned concepts:

1) aocmrs(X):- luxury_item(X), sports_car(X), european(X), british(X).
2) aocmrs(X):- luxury_item(X), european(X), british(X).
3) aocmrs(X):- luxury_item(X), sports_car(X), european(X).
4) aocmrs(X):- luxury_item(X), european(X).
5) aocmrs(X).

So, in the fourth concept, for example the learned concept is that any european luxury item is adversited on musical radio.

Appendix H

Evaluation of Offline Prints

Relevance Assessment - Instructions

Following is a reprint of one of your latest queries in the Okapi information retrieval system. The query is followed by some 30 references. For each of these references, please answer the question at the bottom of the reference.

Question : Whether you have seen this document or not, is it
Relevant / ***Partially relevant*** / ***Non-relevant***?

Given what you were looking for at the time of this search, please judge this document's appropriateness or relevance whether or not you have read or seen it, or had done so before the search.

Thank you very much for your co-operation.

Appendix I

Number of Online Queries and Sessions: Evaluation 1

User	No. of Queries	No. of Sessions
3	5	5
4	6	3
5	2	2
6	48	18
7	8	3
8	2	2
9	3	3
10	6	3
11	4	4
12	11	10
13	13	10
Tll	108	63

Table 1: *Table showing the users, the number of their learning queries and sessions.*

Appendix J

Number of Online Queries and Sessions: Evaluation 2

User	No. of Queries	No. of Sessions
1	29	16
5	15	9
6	5	5
7	6	3
8	3	3
11	2	2
13	4	4
14	19	7
15	19	8
Ttl	102	57

***Table 1**: Table showing the users, the number of their learning queries and sessions.*

Appendix K

Queries: Evaluation 1

User	Last Query	NMPW	NGW	NAW
3	groupware	456	456	456
4	sensors and actuators	562	562	562
5	subpixel accuracy image	32	76	89
6	epipolar mass	0	0	56
7	freeman machine vision	7	33	226
8	photogrammetry	300	300	300
9	machine vision	1180	1180	1180
10	relational database in functional language	50	1018	2750
11	simulated annealing	517	517	517
12	distributed shared virtual memory fault tolerant recovery reliable	0	16	91
13	distributed office computer	137	238	4051

Table 1: The queries for which relevance judgements were obtained.

Also shown in the table are: the number of documents with maximum possible weight (NMPW), the number of documents with good weight (NGW) and average weight (NAW).

Appendix L

Queries: Evaluation 2

User	Last Query	NMPW	NGW	NAW
1	serpent	7	7	7
5	parallel concurrent systems logic	276	2021	3416
6	genetic algorithm	409	409	409
7	audio analogue	67	67	67
8	constraint logic programming	413	777	6610
11	cognitive modelling machine learning	32	481	1381
13	multimedia used medicine	1	7	2850
14	maybank	10	10	10
15	hough subpixel	2	2	2

Table 1: The queries for which relevance judgements were obtained.

Also shown in the table are: the number of documents with maximum possible weight (NMPW), the number of documents with good weight (NGW) and average weight (NAW).

Appendix M

Relevance Judgement Results: Evaluation 1

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	1	0	0	9	1	0	0	14	1	0	0	19
4	0	0	1	4	0	0	2	8	0	0	2	13	0	0	2	18
5	3	0	2	0	3	1	4	2	3	2	4	6	4	2	5	9
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	1	0	4	0	1	0	9	0	1	0	14	0	1	0	19
9	4	0	1	0	8	0	2	0	8	0	2	5	8	0	2	10
10	1	1	0	3	1	1	0	8	1	1	0	13	2	1	0	17
11	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
12	0	0	0	5	0	0	0	10	0	1	0	14	0	1	0	19
13	0	0	0	5	0	0	1	9	0	0	1	14	0	0	2	18
Ttl	8	2	4	41	13	3	9	85	13	5	9	138	15	5	11	189

Table 1: Learner - L1

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	1	0	0	9	1	0	0	14	1	0	0	19
4	0	0	1	4	0	0	1	9	0	0	2	13	0	0	2	18
5	3	0	2	0	3	1	4	2	3	2	4	6	4	2	5	9
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	1	0	4	0	1	0	9	0	1	0	14	0	1	0	19
9	4	0	1	0	8	0	2	0	8	0	2	5	8	0	2	10
10	1	1	0	3	1	1	0	8	1	1	0	13	2	1	0	17
11	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
12	0	0	0	5	0	0	0	10	0	1	0	14	0	1	0	19
13	0	0	0	5	0	0	1	9	0	0	1	14	0	0	2	18
Ttl	8	2	4	41	13	3	8	86	13	5	9	138	15	5	11	189

Table 2: Learner - L2

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
4	0	0	1	4	1	0	1	8	1	0	1	13	1	0	1	18
5	2	0	2	1	3	0	4	3	3	1	4	7	5	3	4	8
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
9	0	0	0	5	0	0	0	10	0	0	0	15	1	0	0	19
10	1	2	0	2	1	3	1	5	1	3	1	10	1	5	2	12
11	0	0	0	5	0	0	0	10	0	0	0	15	1	0	0	19
12	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
13	1	0	0	4	1	0	1	8	2	0	1	12	3	0	1	16
Tll	4	2	3	46	6	3	7	94	8	4	7	146	13	8	8	191

Table 3: Learner - L3

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
4	0	0	0	5	0	0	2	8	0	0	2	13	0	0	2	18
5	2	0	2	1	2	0	4	4	4	1	4	6	5	4	4	7
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
9	0	0	0	5	0	0	0	10	0	0	0	15	1	0	0	19
10	1	2	0	2	1	3	1	5	1	3	1	10	1	5	2	12
11	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
12	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
13	1	0	0	4	1	0	1	8	2	0	1	12	3	0	1	16
Tll	4	2	2	47	4	3	8	95	9	4	8	144	12	9	9	190

Table 4: Learner - L4

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
4	1	0	1	3	1	0	1	8	1	0	1	13	1	0	4	15
5	3	2	0	0	5	2	3	0	5	3	4	3	5	6	5	4
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	1	0	4	0	1	0	9	0	1	0	14	0	1	0	19
9	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
10	1	3	1	0	2	3	2	3	2	3	3	7	2	3	3	12
11	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
12	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
13	1	1	2	1	1	5	3	1	4	7	3	1	5	9	3	3
Tll	8	7	4	36	11	11	9	79	14	14	11	126	15	19	15	171

Table 5: Learner - L5

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
4	1	0	3	1	1	0	3	6	1	0	3	11	1	0	3	16
5	3	2	0	0	4	4	2	0	5	6	4	0	5	8	7	0
6	1	0	3	1	1	0	3	6	1	0	3	11	5	0	4	11
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	1	0	4	0	1	0	9	0	1	0	14	0	1	0	19
9	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
10	1	3	1	0	2	3	2	3	2	3	3	7	2	3	3	12
11	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
12	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
13	1	1	2	1	1	5	3	1	4	7	3	1	5	9	3	3
Tll	10	7	9	29	12	13	13	72	16	17	16	116	21	21	20	158

Table 6: Learner - L6

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	2	0	1	7	7	0	1	7	9	0	4	7
4	1	0	3	1	1	0	3	6	1	0	3	11	1	0	3	16
5	3	2	0	0	4	4	2	0	5	6	4	0	5	8	7	0
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	1	2	2	0	1	2	7	0	1	2	12	0	1	2	17
9	4	0	1	0	8	0	2	0	8	0	2	5	8	0	2	10
10	1	3	1	0	2	6	2	0	2	6	2	5	2	6	3	9
11	0	0	0	5	2	0	0	8	6	0	0	9	6	0	0	14
12	1	1	0	3	1	1	0	8	1	1	0	13	1	1	0	18
13	0	1	0	4	0	1	0	9	1	2	1	11	1	5	3	11
Til	10	8	7	30	20	13	12	65	31	16	15	103	33	21	24	142

Table 7: Learner - L12

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
4	1	0	2	2	1	0	2	7	1	0	2	12	1	0	2	17
5	3	0	2	0	3	3	3	1	4	4	5	2	4	4	6	6
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	1	2	2	0	1	2	7	0	1	2	12	0	1	2	17
9	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
10	2	0	0	3	2	0	1	7	2	0	2	11	2	1	2	15
11	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
12	0	2	0	3	0	2	0	8	0	2	0	13	0	2	0	18
13	1	0	1	3	1	0	1	8	1	0	1	13	1	0	1	18
Til	10	3	7	35	10	6	9	85	11	7	12	135	11	8	13	188

Table 8: Learner - L13

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	2	0	1	7	7	0	1	7	9	0	4	7
4	1	0	3	1	1	0	3	6	1	0	3	11	1	0	3	16
5	3	2	0	0	4	4	2	0	5	6	4	0	5	8	7	0
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	1	2	2	0	1	2	7	0	1	2	12	0	1	2	17
9	4	0	1	0	8	0	2	0	8	0	2	5	8	0	2	10
10	1	3	1	0	2	6	2	0	2	6	2	5	2	6	3	9
11	0	0	0	5	2	0	0	8	6	0	0	9	6	0	0	14
12	1	1	0	3	1	1	0	8	1	1	0	13	1	1	0	18
13	0	1	0	4	0	1	0	9	1	2	1	11	1	5	3	11
Ttl	10	8	7	30	20	13	12	65	31	16	15	103	33	21	24	142

Table 9: Learner - L14

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
4	1	0	0	4	1	0	0	9	1	0	0	14	2	0	0	18
5	2	0	2	1	4	1	4	1	5	3	4	3	5	5	5	5
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	0	0	5	0	0	0	10	0	0	0	15	0	1	0	19
9	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
10	1	3	0	1	1	3	1	5	1	3	2	9	2	5	2	11
11	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
12	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
13	2	1	0	2	2	1	0	7	4	2	0	9	4	2	0	14
Ttl	7	4	2	42	9	5	5	91	13	8	6	138	15	13	7	185

Table 10: Learner - L15

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
4	0	0	1	4	0	0	1	9	0	0	2	13	0	0	2	18
5	3	0	2	0	3	1	4	2	3	2	4	6	4	2	5	9
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
9	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
10	1	1	0	3	1	1	0	8	1	1	0	13	1	1	0	18
11	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
12	0	0	0	5	0	1	0	9	0	2	0	13	0	2	0	18
13	0	0	1	4	1	0	1	8	1	0	2	12	1	0	2	17
Tll	5	1	4	45	6	3	6	95	7	5	8	145	8	5	9	198

Table 11: Learner - L16

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
4	0	0	2	3	0	0	2	8	0	0	2	13	0	0	2	18
5	3	0	2	0	3	1	4	2	3	2	4	6	4	2	5	9
6	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
9	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
10	1	1	0	3	1	1	0	8	1	1	0	13	1	1	0	18
11	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
12	0	0	0	5	0	0	0	10	0	0	0	15	0	1	0	19
13	0	0	0	5	1	0	0	9	1	0	0	14	1	1	0	18
Tll	5	1	4	45	6	2	6	96	8	3	6	148	9	5	7	199

Table 12: Learner - L17

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	1	0	0	4	1	0	0	9	1	0	0	14	3	1	1	15
4	0	0	1	4	0	0	1	9	0	0	1	14	0	0	1	19
5	3	0	2	0	4	2	4	0	4	3	4	4	5	5	6	4
6	3	0	2	0	4	0	2	4	4	0	2	9	5	0	3	12
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	0	0	0	5	0	0	0	10	1	0	0	14	1	0	0	19
9	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
10	2	0	0	3	2	0	2	6	2	3	3	7	2	3	4	11
11	0	0	0	5	0	0	0	10	0	0	0	15	1	0	0	19
12	0	1	0	4	0	1	0	9	0	1	0	14	0	1	0	19
13	0	1	0	4	1	1	0	8	1	1	0	13	1	1	0	18
Ttl	10	2	5	38	13	4	9	84	14	8	10	133	19	11	15	175

Table 13: Learner - L18

User	R	N	P	O	R	N	P	O	R	N	P	O	R	N	P	O
3	2	0	0	3	7	0	0	3	7	0	0	8	7	0	0	13
4	0	0	0	5	0	0	0	10	1	0	2	12	1	0	2	17
5	2	2	1	0	5	2	3	0	5	3	4	3	5	6	5	4
7	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
8	2	3	0	0	4	4	2	0	4	8	3	0	6	10	4	0
9	0	0	0	5	0	0	0	10	2	0	0	13	2	0	0	18
10	1	0	2	2	1	4	2	3	2	4	3	6	2	4	4	10
11	1	0	0	4	1	0	0	9	1	0	0	14	1	0	0	19
12	0	0	0	5	0	0	0	10	0	0	0	15	0	0	0	20
13	0	3	1	1	0	3	1	6	0	3	1	11	0	3	1	16
Ttl	8	8	4	30	18	13	8	61	22	18	13	97	24	23	16	137

Table 14: Learner - L19
(Note: User 6 missing as files were too large to run on the system.)

Appendix N

Relevance Judgement Results: Evaluation 2

User	R	N	P	O	R	N	P	O
1	0	3	2	0	0	7	3	0
5	1	1	3	0	3	3	4	0
6	0	5	0	0	0	10	0	0
7	4	1	0	0	4	2	4	0
8	5	0	0	0	6	2	2	0
11	1	1	3	0	1	2	7	0
13	0	5	0	0	0	9	1	0
14	2	2	1	0	4	2	4	0
15	4	0	1	0	3	1	1	0
Ttl	17	18	10	0	25	38	27	0

Table I: Learner - L2

User	R	N	P	O	R	N	P	O
1	3	2	0	0	4	3	3	0
5	1	0	4	0	3	2	5	0
6	0	5	0	0	0	10	0	0
7	4	1	0	0	4	2	4	0
8	3	1	1	0	7	1	2	0
11	3	0	2	0	3	1	6	0
13	0	4	1	0	0	9	1	0
14	0	0	5	0	2	0	8	0
15	4	0	1	0	5	1	4	0
Ttl	18	13	14	0	28	29	33	0

Table II: Learner - L15

User	R	N	P	O	R	N	P	O
1	4	1	0	0	4	6	0	0
5	1	1	3	0	3	3	4	0
6	0	2	3	0	0	5	5	0
7	1	3	1	0	4	4	2	0
8	5	0	0	0	8	1	1	0
11	1	1	3	0	1	2	7	0
13	0	5	0	0	0	9	1	0
14	3	0	2	0	4	2	4	0
15	2	0	3	0	6	1	3	0
Ttl	17	13	15	0	30	33	27	0

Table III: Learner - L17

User	R	N	P	O	R	N	P	O
1	3	2	0	0	4	6	0	0
5	2	0	3	0	5	1	4	0
6	1	4	0	0	3	7	0	0
7	2	2	1	0	5	3	2	0
8	5	0	0	0	6	2	2	0
11	1	1	3	0	1	2	7	0
13	4	0	1	0	7	0	3	0
14	0	0	5	0	1	0	9	0
15	3	0	2	0	8	0	2	0
Ttl	21	9	15	0	40	21	29	0

Table IV: Learner - L18

User	R	N	P	O	R	N	P	O
1	4	1	0	0	4	2	0	0
5	0	5	0	0	0	7	3	0
6	5	0	0	0	9	1	0	0
7	1	3	1	0	1	7	2	0
8	1	4	0	0	1	7	2	0
11	0	0	5	0	0	1	9	0
13	3	0	2	0	4	2	4	0
14	0	0	5	0	1	0	8	0
15	5	0	0	0	7	0	3	0
Ttl	19	13	13	0	27	27	31	0

Table V: Okapi-Normal (sv)

Appendix O

Precision Values at Various Cut-off Points: Evaluation 1

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	14	57	86
10	25	52	88
15	27	48	81
20	31	48	84

Table 1: Learner - L1

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	14	57	86
10	25	52	88
15	27	48	81
20	31	48	84

Table 2: Learner - L2

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	9	44	77
10	16	38	81
15	19	42	79
20	29	45	72

Table 3: Learner - L3

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	8	50	75
10	15	27	80
15	21	43	80
20	30	40	70

Table 4: *Learner - L4*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	19	42	63
10	31	35	65
15	39	36	64
20	49	30	61

Table 5: *Learner - L5*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	26	38	73
10	38	31	66
15	49	32	65
20	62	34	66

Table 6: *Learner - L6*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	25	40	68
10	45	44	71
15	62	50	74
20	78	42	73

Table 7: *Learner - L12*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	20	50	85
10	25	40	76
15	30	37	77
20	32	34	75

Table 8: Learner - L13

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	25	40	44
10	45	44	71
15	62	50	74
20	78	42	73

Table 9: Learner - L14

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	13	54	70
10	19	47	74
15	27	48	70
20	35	43	62

Table 10: Learner - L15

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	10	50	90
10	15	40	80
15	20	35	75
20	22	36	77

Table 11: Learner - L16

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	10	50	90
10	14	43	86
15	17	47	82
20	19	47	84

***Table 12**: Learner - L17*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	17	59	88
10	26	50	85
15	32	44	75
20	45	42	75

***Table 13**: Learner - L18*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	20	40	60
10	39	46	66
15	53	42	66
20	63	38	63

***Table 14**: Learner - L19*

A 'slot' refers to all precision values of one type (either R/A or (R+P)/A) for a particular cut-off point. After identifying a maximum value (m) for a particular slot, all values between (and including) m and m-5% have been marked.

Appendix P

Precision Values at Various Cut-off Points: Evaluation 2

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	45	38	60
10	90	28	58

Table 1: Learner - L2

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	45	40	71
10	90	31	68

Table 2: Learner - L15

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	45	38	71
10	90	33	63

Table 3: *Learner - L17*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	45	47	80
10	90	44	77

Table 4: *Learner - L18*

Rank Position	Total Assessed (A)	R/A (%)	(R+P)/A (%)
5	45	42	71
10	85	32	68

Table 5: *Okapi Normal (sv)*

A 'slot' refers to all precision values of one type (either R/A or (R+P)/A) for a particular cut-off point. The shaded % values indicate those above the Okapi Normal system for that slot.

Appendix Q

Example of Relevant Documents for Queries

The queries for a particular user were as follows:

q(1) = *"graph grammars programming"*

q(2) = *"graphical programming concurrent"*

q(3) = *"software tools parallel"*

Query	Doc-id	Index terms
q(1)	3601331	support, grafic, languag, structur, specification, design, model, representation, basic, pictur, element, line, segment, rectangl, geg, object, oriented, generator, technologi, chang, dependent, group, need, computer, directed, graf, grammar, high, level, program, softwar, tool, edit
	3512638	program, interconnection, structur, aggregat, rewrit, graf, grammar, parallel, 0105, logical, abstraction, specification, script, derivation, sequenc, support, element, level
	3557048	graf, grammar, paradigm, implement, visual, languag, specification, rewrit, data, structur, diagram, syntax, directed, attribut, arrow, box, color, representation, formal, description, programmed, attributed, computer, grafic, edit, user, interfac
	3513448	graf, sup, ed, interactiv, grammar, undirected, multipl, edg, arbitrari, label, computer, grafic, design, diagram, representation, visual, support, directed
	3624845	1989, ieee, workshop, visual, languag, cat, 0012, rome, 0169, 4-6, oct, reason, 0163, larg, 0091, intelligent, grafic, interfac, directed, graf, program, iconic, queri, pictorial, data, gloto, syntax, qbd, unix, live, micro, tool, 0105, form, manipulation, type, browser, symbolic, expression, spatial, algebra, xviqu, expert_system, signor, computer, 0092, grammar, user, model, formal, specification
q(2)	3526965	compar, barrier, algorithm, time, 0110, concurrent, computer, synchronisation, mechanist, linear, logarithmic, depth, grafic, model, parallel, program, performenc
	3533602	model, based, program, simulation, 0293, behavior, concurrent, multiprocessor, machin, simula, sun, workstation, 0079, grafic, interfac, multiprocess, softwar, tool, user
	3451813	specifi, concurrent, delta, grammar, specification, graf, based, notation, distributed, formal, algebraic, basis, dynamic, dine, filosofer, multipl, server, 0220, arbitrari, topologi, multiprocess, program, grafic
	3625304	prototyp, discret, part, manufactur, specification, languag, high, level, petri, net, object, oriented, 0144, decomposition, concurrent, topdown, bottom, formal, 0097, program, softwar, engineer, based, grafic, simulation
	3514044	visual, program, parallel, softwar, grafic, tool, gilt, languag, transputer, based, multiprocessor, 0144, graf, interprocess, communication, path, high, resolution, communicat, arrai, concurrent, information, transfer, bandwidth, natural, representation, parallelist, conventional, textual, computer, user, interfac, level, machin
	3548183	declarativ, visualis, concurrent, 0079, grafic, representation, object, monitor, debug, larg, scale, program, visualisation, shared, data, space, parallel, algorithm, visual, abstraction, region, label, computer, structur, notation
	3494662	intelligent, tutor, concurrent, program, 0059, parallel, computer, specialist, knowledg, based, assistenc, prototyp, sun3, grafic, workstation, 0080, 0099, teach
	3471082	tool, concurrent, program, suprenum, grafic, representation, numerical, 250, vector, node, 0012, gflop, peak, performenc, parallel, 0080, verification, simulator, communication, generation, interprocess, data, exchang, grid, based, visualisation, test, time, softwar, prototyp, oriented, 0220, machin, larg
	3450410	specification, verification, concurrent, program, automata, nondeterministic, finit, state, infinit, sequenc, run, formalist, temporal, properti, extended, logic, etl, structured, diagram, notation, grafic, representation, sound, proof, rule, automaton, formal, parallel, conventional, 0079

Query	Doc-id	Index terms
q(3)	3570654	mixed, case, cecad, tool, eas, designer, headach, embedded, design, cycl, level, architectural, specification, hardwar, softwar, trade, integration, ada, assessment, statemat, matrixx, grafic, representation, codelink, instruction, set, simulator, structur, map, code, debugger, 0058, computer, architectur, logic, parallel
	3643259	softwar, tool, real, time, parallel, 0105, esp, visual, program, recognition, granulariti, simulation, sequential, hardwar, level, computer, grafic
	3533290	softwar, parallel, computer, exploit, parallelist, 0105, tool, algorithm, hammersmith, 0314, 12-15, june, 1989, program, architectur, intel, ipsc, 0009, concurrent, file, support, portabiliti, partool, equus, operat_system, cs, prolog, transputer, automatic, vectorisation, parallelisation, supercomputer
	3666992	von, neumann, parallel, 0080, era, tool, build, fase, machin, capabiliti, solv, histori, technological, forecast, computer, supercomputer, hardwar, softwar
	3558435	program, 0105, parallel, 0079, performenc, debug, algorithm, interactiv, tool, faust, data, flow, structur, design, set, 0009, real
	3558429	trend, parallel, algorithm, design, supercomputer, systolic, program, 0105, tool

REFERENCES

ACKLEY, D.H., HINTON, G.E., AND SEJNOWSKI, T.J. (1985).
A learning algorithm for Boltzmann machines.
Cognitive Science 9.

AGOSTI, M. (1990).
Hypermedia and information retrieval. *In: EUROPEAN SUMMER SCHOOL IN INFORMATION RETRIEVAL, BRESSANONE 9-12 JULY 1990. Proceedings.* Italy : Associazione Italiana per l'informatica ed il calcolo automatico, pp. 431-498.

ANDERSON, J.R. (1983).
Acquisitions of proof skills in geometry. *In:* Machine learning: An artifical intelligence approach, ed. by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Palo Alto, CA: Tioga Press).

ANDERSON, J.R. AND KLINE, P.J. (1979).
A learning system and its psychological implications. *In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 6TH, TOKYO, JAPAN 1979. Proceedings.* Morgan Kauffman, pp.16-21.

BARKER, P.H., VEAL, D.C., AND WYATT, B.K. (1972).
Towards automatic profile construction.
Journal of Documentation 28(1): 44-55.

BARTSCHI, M. (1985).
An overview of information retrieval subjects.
Computer (May): 67-84.

BAWDEN, D. (1990).
User-orientated evaluation of information systems and services. (Hants, England: Gower Publishing Company).

BAWDEN, D. (1992).
What kind of resource.
The Computer Bulletin (April/May), 4(2): 6-7.

BELEW, R.K. (1991).
Issues raised by machine learning in information retrieval. *In: ANNUAL INTERNATIONAL ACM/SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL 14th, CHICAGO, U.S.A, 13-16 OCTOBER 1991. Proceedings. U.S.A.:* ACM/SIGIR, pp.87-90.

BELKIN, N.J. (1978).
Information concepts for information science.
Journal of Documentation 34: 55-85.

BELKIN, N.J. (1990).
Interfaces for information retrieval systems, user modelling in information retrieval Systems. *In: EUROPEAN SUMMER SCHOOL IN INFORMATION RETRIEVAL, BRESSANONE, 9-12 JULY 1990. Proceedings.* Italy : Associazione Italiana per l'informatica ed il calcolo automatico, pp. 274-334.

BELKIN, N.J. AND ROBERTSON, S.E. (1976).
Information science and the phenomenon of information.
Journal of the American Society for Information Science (July-August): 197-204.

BHANDARKAR, A., CHANDRASEKAR, R., RAMANI, S., AND BHATNAGAR, A. (1989).
Intelligent categorization, archival and retrieval of information. *In*: Lecture notes in artifical intelligence: Knowledge based computer systems. International Conference KBCS'89, Bombay, India. ed. by S. Ramani, R. Chandrasekar, K.S.R. Anjaneyulu. (Berlin: Springer-Verlag).

BHATIA, S.K. AND DEOGUN, J.S. (1991).
Methodologies for intelligent systems. *In*: Lecture notes in artificial intelligence: 6th International Symposium, ISMIS'91, Charlotte, U.S.A. ed. by Z.W. Ras, M. Zemankova. (Berlin: Springer-Verlag).

BOOKSTEIN, A., KLEIN, S.T., AND RAITA, T. (1993).
Is Huffman coding dead? *In*: *ANNUAL INTERNATIONAL ACM SICIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL , 16TH, PITTSBURGH, USA, 1993. Proceedings.* New York: Association for computing machinery, pp. 80-87.

BOOKSTEIN, A. AND SWANSON, D. (1974).
Probabilistic models for automatic indexing.
Journal of the American Society for Information Science 25: 312-318.

BRAVERMAN, M.S. AND RUSSELL, S.J. (1988).
Boundaries of operationality. *In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 5TH, THE UNIVERSITY OF MICHIGAN, ANN ARBOR, JUNE 12-14, 1988. Proceedings.* San Mateo CA: Morgan Kaufmann Publishers Inc., pp. 221-234.

BROOKS, H.M., DANIELS, P.J. AND BELKIN N.J. (1985).
Problem descriptions and user models: developing an intelligent interface for document retrieval systems. Advances in information retrieval. *In: INFORMATICS CONFERENCE, 8TH, LONDON 1985. Proceedings.* London: ASLIB, pp. 191-214.

BUCHANAN, B.G. AND FEINGENBAUM, E.A. (1978).
DENDRAL and Meta-DENDRAL: Their applications dimension.
Artificial Intelligence 11: 5-24.

BUCHANAN, B.G. AND MITCHELL, T.M. (1978).
Model directed learning of production rules. *In*: Pattern directed inference systems, ed. by D. Waterman and F. Hayes-Roth. (New York: Academic Press).

CARBONELL, J.G., MICHALSKI, R.S., MITCHELL, T.M. (1983).
Chapter 1 An overview of machine learning. *In*: Machine learning: An artificial intelligence Approach, Volume 1, ed. by R.S. Michalski, J.G. Carbonell and T.M. Mitchell. (Los Altos, CA: Morgan Kaufmann).

CLEVERDON, C.W. (1962).
Report on the testing and analysis of an investigation into the comparative efficiency of indexing systems. (Cranfield, England: College of Aeronautics).

CLEVERDON, C.W. (1967).
The Cranfield tests on index language devices. *ASLIB, JUNE 1967. Proceedings.* 19(6).

CLEVERDON, C.W. (1990).
Chapter 9 Questions of relevance. *In*: User-orientated evaluation of information systems and services, D. Bawden. (Hants, England: Gower Publishing Company).

CLEVERDON, C.W., MILLS, J. AND KEEN, E.M. (1966).
Factors determining the performance of indexing systems. Volume 1: Design, Volume 2: Test Results. (Cranfield, England: Aslib Cranfield Research Project).

COATES-STEPHENS, S. (1992).
The analysis and acquisition of proper names for robust text understanding. PhD thesis: City University, Department of Computer Science.

CONKLIN, J. (1987).
Hypertext: An introduction and survey.
Computer (September): 17-41.

CROFT, W.B. (1986).
Boolean queries and term dependencies in probabilistic retrieval models.
Journal of the American Society for Information Science 37(2): 71-77.

CROFT, W.B. AND HARPER, D.J. (1979).
Using probabilistic models of document retrieval without relevance information.
Journal of Documentation 35(4): 285-295.

CROFT, W.B. AND THOMPSON, R.H. (1985).
An expert assistant for document retrieval systems. COINS Technical Report 85-05 : University of Massachusetts, Computer and Information Science Department.

DANIELS, P.J. (1986).
Progress in documentation: Cognitive models in information retrieval - an evaluative review. *Journal of Documentation 42(4)*: 272-305.

DARPA (1991).
Proceedings of the 3rd Message Understanding Conference. Morgan Kaufmann.

DAVIS, R. AND LENAT, D.B. (1982).
Knowledge-based systems in artificial intelligence. (New York: McGraw-Hill).

DEJONG, G. (1981).
Generalizations based on explanations. *In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 7TH, VANCOUVER, B.C., CANADA. Proceedings.* Morgan Kaufmann, pp.67-69.

DEJONG, G. AND MOONEY, R. (1986).
Explanation based learning: An alternative view. *In:* Machine learning: An artificial intelligence approach, Volume II, ed. by R.S. Michalski, J.G. Carbonell and T.M. Mitchell. (Los Altos, CA: Morgan Kaufmann).

DIEDERICH, J. (1989).
"Learning by instruction" in connectionist system. *In: INTERNATIONAL WORKSHOP ON MACHINE LEARNING, 6TH, CORNELL UNIVERSITY, ITHACA, NEW YORK 1989. Proceedings.* San Mateo CA: Morgan Kaufmann, pp. 66-68.

DOSZKOCS, T.E. (1986).
Natural language processing in information retrieval.
Journal of the American Society for Information Science 37(4): 191-196.

DOSZKOCS, T.E. AND RAPP, B.A. (1979).
Searching MEDLINE in English: a prototype user interface with natural language query, ranked output, and relevance feedback. *In: INFORMATION CHOICES AND POLICIES, ASIS ANNUAL MEETING, 42ND, MINNEAPOLIS, MINNESOTA, OCT 14-18 1979. Proceedings.* New York: Knowledge Industry Publications Inc., pp. 131-139.

DUMAIS, S.T. (1988).
Chapter 30 Textual information retrieval. In : Handbook of human computer interactions, ed. by M Helander. (North Holland: Elservier Science Publishers B.V.).

DURHAM, T. (1989).
Analysing neural know-how.
Computing (March): 30-31.

DURHAM, T. (1989).
The gathering of an information harvest.
Computing (April): 31-32

DURHAM, T. (1989).
Having the last word on information retrieval.
Computing (May): 32-33.

EFTHIMIADIS, E.N. (1992).
Interactive query expansion and relevance feedback for document retrieval systems. PhD Thesis: City University, Department of Information Science.

ELLIS, D. (1992).
The physical and cognitive paradigms in information retrieval research.
Journal of Documentation 48(1): 45-64.

ERMAN, L.D., HAYES-ROTH, F., LESSER, V.R., AND REDDY, D. R. (1980).
The Hearsay II speech understanding system: integrating knowledge to resolve uncertainty.
ACM Computing Surveys 12: 213-253.

FAIRCHILD, K.F., POLTROCK, S.E., AND FURNAS, G.W. (1987).
SemNet: Three-dimensional graphic representations of large knowledge bases. *In*: Cognitive science and its applications for human-computer interaction, ed. by R Guindon. Lawrence Erlbaum.

FAIRTHORNE, R.A. (1965).
Some basic Comments on retrieval testing.
Journal of documentation 21(4): 603-605.

FEIGENBAUM, E. AND MCCORDUCK, P. (1983).
The fifth generation: Artificial intelligence and Japan's computer challenge to the world. (Reading, MA: Addison-Wesley).

FININ, T.W. (1983).
Providing help and advice in task-oriented systems. *In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 8TH, KARRRRRLSRUHE, WEST GERMANY, 1983.* Proceedings. Morgan Kauffmann, pp.176-178.

FISHER, D. AND LANGLEY, P. (1985).
Approaches to conceptual clustering. *In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 9TH, LOS ANGELES, CA. 1985. Proceedings.* pp. 691-697.

FISHER, D.H. (1987).
Conceptual clustering, learning from examples and inference. *In: INTERNATIONAL WORKSHOP ON MACHINE LEARNING, 9TH, LOS ALTOS, CA 1987. Proceedings.* California: Morgan Kauffmann Publishers, Inc., pp. 38-49.

FOX, E.A. (1989).
Knowledge-based information retrieval. Tutorial notes given on the topic of knowledge-based information retrieval. *In: INTERNATIONAL ACM SIGIR CONFERENCE OF RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 12TH, CAMBRIDGE M.A., USA, 25-28 JUNE 1989. Proceedings.*

FREI, H.P. AND SCHAUBLE, P. (1991).
Determining the effectiveness of retrieval algorithms.
Information Processing and Management 27(2/3): 153-164.

FREIHERR, G.T. (1979).
The problems and promises of artificial intelligence.
Research Resources Reporter 3(9): 1-6.

FRIEDBERG, R.M. (1958).
A learning machine: Part I.
IBM Journal 2: 2-13.

FRIEDBERG, R.M. (1959).
A learning machine: Part II.
IBM Journal of Research and Development 3(3): 282-287

FRISSE, M.F. AND COUSINS, S.B. (1992).
Interfaces for end-user information seeking.
Journal of the American Society for Information Science 43(2): 181-191.

FUHR, N. AND BUCKLEY, C. (1989)
Probabilistic indexing from relevance feedback data. *In: INTERNATIONAL CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, ACM, CAMBRIDGE, M.A., USA, June 1989. Proceedings. Association for Computing Machinery.*

FUHR, N. AND BUCKLEY, C. (1993).
Optimizing document indexing and search term weighting based on probabilistic models. *In: THE FIRST TEXT RETRIEVAL CONFERENCE (TREC-1), GAITHERSBURG, MARYLAND 1993. Proceedings.* Washington D.C.: National Institute of Standards and Technology (Special Publication 500-207), U.S. Government Printing Office, pp. 89-99.

FUHR, N. AND ROBERTSON, S. (1993).
Workshop on: Machine learning and relevance feedback. *In: THE FIRST TEXT RETRIEVAL CONFERENCE (TREC-1), GAITHERSBURG, MARYLAND, 1993. Proceedings.* Washington D.C.: National Institute of Standards and Technology (Special Publication 500-207), U.S. Government Printing Office, pp. 369-370.

FURNAS, G.W. (1986).
Generalized fisheye views. *In*: Proceedings of CHI'86: Human Factors in Computing Systems, 1986.

FURNAS G.W., LANDAUER T.K., GOMEZ L.M., AND DUMAIS S.T. (1987).
The vocabulary problem in human-system communication.
Communications of the ACM 30(11): 964-971.

GAY, L.S. AND CROFT, W.B. (1990).
Interpreting nominal compounds for information retrieval.
Information Processing and Management 26(1): 21-38.

GORDON, M. (1990).
Evaluating the effectiveness of information retrival systems using simulated queries.
Journal of the American Society for Information Science 41(5): 313-323.

GORDON, M. AND KOCHEN, M. (1989).
Recall-precision trade-off: A derivation.
Journal of the American Society for Information Science 40(3): 145-151.

GOTLIEB, C.C. AND KUMAR, S. (1968).
Semantic clustering of index terms.
Journal of the Association for Computing Machinery 15(4).

GRANT, A.S. (1990).
Modelling cognitive aspects of complex control tasks. PhD thesis: University of Strathclyde, Department of Computer Science.

GUIDA, G. AND TASSO, C. (1983).
IR-NLI: An expert natural language interface to online databases. *In: CONFERENCE ON APPLIED NATURAL LANGUAGE PROCESSING, SANTA MONICA, CA, FEB. 1-3, 1983. Proceedings.* Menlo Park, CA. USA: Association Computer Linguistics, pp. 31-38.

GÜNTZER, U., JÜTTNER, G., SEEGMÜLLER, G., AND SARRE, F. (1989).
Automatic thesaurus construction by machine learning from retrieval sessions.
Information Processing & Management 25(3): 265-273.

HAINES, D. AND CROFT, W.B. (1993).
Relevance feedback and inference networks. *In: ANNUAL INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 16TH, PITTSBURGH, USA. Proceedings.* New York : The association of computing machinery, pp.2-11.

HALASZ, F.G. (1987).
Reflections on notecards: Seven issues for the next generation of hypermedia systems. Communications of the ACM 31(7): 36-52.

HALASZ, F.G., MORAN T. AND TRIGG, R. (1987).
Notecards in a nutshell. *In: CHI+GI: HUMAN FACTORS IN COMPUTING SYSTEMS AND GRAPHICS INTERFACE, TORONTO, CANADA, 5-9 APRIL 1987. Proceedings.* U.S.A.: ACM, pp. 45-52.

HANCOCK-BEAULIEU, M. AND WALKER, S. (1992).
An evaluation of automatic query expansion in an online library catalogue.
Journal of documentation 48(4): 406-421.

HARTER, S.P. (1975).
A probabilistic approach to automatic keyword indexing, Part II: An algorithm for probabilistic indexing.
Journal of the American Society for Information Science 26: 280-289.

HAYES, P.J., KNECHT, L.E., AND CELLIO, M.J. (1988).
A news story categorization system. *In: CONFERENCE ON APPLIED NATURAL LANGUAGE PROCESSING, 2ND, Austin, Texas, USA, 9-12 Feb. 1988. Proceedings.* Morriston, New Jersey, U.S.A.: Association Computing Linguistics, pp. 9-17.

HEBB, J.O. (1949).
The organization of behaviour. (New York: John Wiley and Sons).

HENRY, G. AND DIODATO (1991).
The rates of assignment of thesaurus terms in the ERIC information retrieval system: An analysis of hierarchies and levels.
Journal of Documentation 47(3): 276-283.

HJORLAND, B. (1992).
The concept of 'subject' in information science.
Journal of Documentation 48(2): 172-200.

HOLLAND, J.H. (1980).
Adaptive algorithms for discovering and using general patterns in growing knowledge bases.
International Journal Policy Analysis and Information Systems 4(3): 245-268.

HOLLAND, J.H., HOLYOAK, K.J., NISBETT, R.E., AND THAGARD, P.R. (1986).
Induction. (Cambridge, MA: MIT Press).

HOPFIELD, J.J. (1982).
Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences 79.

IDE, E. (1971).
New experiments in relevance feedback. *In*: The smart retrieval system, ed. by G. Salton. (New Jersey: Prentice-Hall).

INGWERSEN, P. (1982).
Search procedures in the library - analysed from the cognitive point of view.
Journal of Documentation 38(3): 165-191.

INGWERSEN, P. (1987).
Towards a new research paradigm in information retrieval in knowledge engineering : Expert systems and information retrieval. (London: I. Wormell, Taylor Graham).

JONES, L.P., GASSIE, E.W.Jr., AND RADHAKRISHNAN, S. (1990).
INDEX: The statistical basis for an automatic conceptual phrase indexing system.
Journal of the American Society for Information Science 41(2): 87-97.

JONES, R.M. (1988).
A comparitive evaluation of two online public access catalogues. (British Library Research Paper 39).

KEDAR-CABELLI, S.T. AND MCCARTHY, L.T. (1987).
Explanation based generalization as resolution theorem proving. *In: INTERNATIONAL WORKSHOP ON MACHINE LEARNING, 4TH, UNIVERSITY OF CALIFORNIA, IRVINE JUNE 22-25 1987. Proceedings.* Los Altos: Morgan Kaufmann, pp. 383-389.

KEEN, E.M. (1991).
The use of term position devices in ranked output experiments.
Journal of Documentation 47(1): 1-22.

KEEN, E.M. (1992).
Term position ranking: Some new test results. *In: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 15TH, COPENHAGEN, JUNE 1992. Proceedings.* New York: Association Computing Machinery, pp. 66-76.

KELLER, R.M. (1985).
Development of a framework for contextual concept learning. *In: INTERNATIONAL MACHINE LEARNING WORKSHOP, SKYTOP, 3RD, PA USA, JUNE 1985. Proceedings.*

KELLER, R.M. (1987).
Concept learning in context. *In: INTERNATIONAL WORKSHOP ON MACHINE LEARNING, 4TH, IRVINE USA JUNE 22-25 1987. Proceedings.* Los Altos: Morgan Kauffmann Publishers Inc., pp. 91-102.

KOBSA, A. AND WAHLSTER, W. (Eds.). (1989).
User models in dialog systems. (Berlin: Springer-Verlag).

KOHONEN, T. (1984).
Self-organization and associative memory. (Berlin: Springer-Verlag)

KOLODNOR, J.L. (1987).
Extending problem solver capabilities through case based inference. *In: INTERNATIONAL WORKSHOP ON MACHINE LEARNING, 4TH, IRVINE USA JUNE 2-25 1987. Proceedings.* Los Altos: Morgan Kauffmann Publishers, Inc., pp. 167-178.

KOLOKOURIS, A.T. (1986).
Machine learning.
BYTE (November): 225-231.

LAIRD J.E., ROSENBLOOM P.S., AND NEWELL A. (1986).
Chunking in soar: The anatomy of a general learning mechanism.
Machine Learning 1(1): 11-46.

LANCASTER, F.W. (1978).
Information retrieval systems. (London: Wiley Interscience).

LANCASTER, F.W. (1979).
Information retrieval systems: Characteristics, testing and evaluation. (Second edition, New York: John Wiley and Sons).

LANGEFORS, B. AND SUNDGREN, B. (1975).
Information systems architecture. (New York: Petrocelli/Charter).

LANGLEY, P. (1986).
On machine learning.
Machine Learning 1(1): 5-10.

LANGLEY, P. (1993).
Invited talk. *In: INTERNATIONAL WORKSHOP ON MACHINE LEARNING, 10th. Proceedings.*

LANGLEY, P. AND CARBONELL, J.G. (1984).
Approaches to machine learning.
Journal of the American Society for Information Science 35(5): 306-316.

LANGLEY, P.W., SIMON, H.A., AND BRADSHAW, G.L. (1983).
Rediscovering chemistry with the BACON System. *In*: Machine learning: An artifical intelligence approach, ed. by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Palo Alto, CA: Tioga Press).

LANGLEY, P., ZYTKOW, J., SIMON, H.A., AND BRADSHAW, G.L. (1986).
The search for regularity: Four aspects of scientific discovery. *In*: Machine learning: An artificial Intelligence approach,Volume II, ed. by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Los Altos CA: Morgan Kaufmann).

LEBOWITZ, M. (1985).
Concept learning in a rich input domain: Generalization-based memory. *In*: Machine learning: An artifical intelligence approach, ed. by R.S. Michalski, J.G. Carbonell and T.M. Mitchell. (Los Altos CA: Morgan Kaufmann).

LEBOWITZ, M. (1987).
Intelligent Information Systems: Or How to Avoid Information Overload, Electro/87 and Mini/Micro Northeast: Focusing on the OEM Conference Record, New York, USA, Electron. Conventions Manage. Los Angeles, CA, 1987.

LENAT, D.B. (1977).
On automated scientific theory formulation: A case study using the AM program.
Machine Intelligence 9: 251-256.

LENAT, D.B. (1983).
The role of heuristics in learning by discovery: Three case studies. *In:* Machine Learning: An Artifical Intelligence Approach, ed. by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Palo Alto CA: Tioga Press).

LENAT, D.B. AND BROWN, J.S., (1984).
Why AM and Eurisko appear to work.
Artificial Intelligence 23(3): 269-294.

LESK, M.E. (1969).
Word-word associations in document retrieval systems.
American Documentation 20(1): 27-38.

LEWIS, D.D. (1990).
Representation and learning in information retrieval. Dissertation proposal: University of Massachusetts, Computer and Information Science Department.

LEWIS, D.D. (1992).
Representation and learning in information retrieval. PhD thesis: Graduate School of the University of Massachusetts, Department of Computer and Information Science.

LICKLIDER, J.C.R. (1965).
Libraries of the future. (Cambridge: MIT Press).

LIDDY, E. D. (1990).
Anaphora in natural language processing and information retrieval.
Information Processing and Management 26(1): 39-52.

LINOFF, G. AND STANFILL, C. (1993).
Compression of indexes with full positional information in very large text databases. *In: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 16TH, PITTSBURGH USA 1993. Proceedings.* New York: Association Computer Machinery, pp. 88-95.

LUGER, G.F. AND STUBBLEFIELD, W.A. (1989).
Artificial intelligence and the design of expert systems. (California: The Benjamin/Cummings Publishing Company, Inc.).

MACHLUP, F. AND MANSFIELD, U. (1983).
The study of information: Interdisiciplinary messages. (New York: John Wiley & Sons, Inc.).

MARCHIONINI, G. (1992).
Interfaces for end-user information seeking.
Journal of the American Society for information science 43(2): 156-163.

MARCUS, R.S. (1986).
Intermediary systems for information retrieval. *In*: Selected Papers from the 1st Conference on Computer Interfaces and Intermediaries for Information Retrieval, Williamsburg Virginia Oct 3-6 1984, ed. by M.E. Powell. (Alexandria, VA: Defence Technical Information Center)

MARON, M.E. AND KUHNS, J.L. (1960).
On relevance, probabilistic indexing and information retrieval.
Journal of the Association for Computing Machinery 7(3): 216-244.

McCARTHY, J. (1968).
Programs with Common Sense. *In:* Semantic Information Processing, ed. by M. Minsk. (Cambridge: MIT Press).

McCLUSKEY, T.L. (1990).
Explanation-based learning. *In:* Intelligent Systems - state of the art and future directions, ed. by Z.W. Ras and M. Zemankova. (New York: Ellis Horwood).

McCULLOCH, W.S. AND PITTS, W. (1943).
A logical calculus of ideas imminent in nervous activity.
Bulletin of Mathematical Biophysics 5: 115-133.

McDERMOTT, D. (1986).
A critique of pure reason. Research Memorandum: Yale University. New Haven, CT, Computer Science Department.

MICHALSKI, R.S. (1980).
Pattern Recognition as Rule-Guided Inductive Inference.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2 (4): 349-361.

MICHALSKI, R.S. (1986).
Chapter 1 Understanding the Nature of Learning: Issues and Research Directions. *In*: Machine Learning : An Artificial Intelligence Approach, Volume 2, ed. by R.S. Michalski, J.G. Carbonell and T.M. Mitchell. (Los Altos, CA : Morgan Kaufmann).

MICHALSKI, R.S., CARBONELL, J.G. AND MITCHELL, T.M. (1983).
Machine Learning: An Artificial Intelligence Approach, (Palo Alto, CA: Tioga Press).

MICHALSKI R.S., CARBONELL J.G. AND MITCHELL T.M. (1986).
Machine Learning: An Artificial Intelligence Approach, Volume II. (Los Altos, CA: Morgan Kaufmann).

MICHALSKI, R.S. AND CHILAUSKY. (1980).
Learning by being told and learning from examples: An experimental comparison of two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis.
International Journal of Policy and Information systems 4(2): 125-161.

MICHALSKI, R.S. AND STEPP, R.E. (1983).
Learning from observation: Conceptual clustering. *In:* Machine learning: An artifical intelligence approach, ed. by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Palo Alto, CA: Tioga Press).

MICHIE, D. (1980).
Knowledge-based systems. Report No. UIUCDCS-R-80-1001: University of Illinois (Urbana, Illinois), Department of Computer Science.

MINSKY, M. (1985).
The society of the mind. (Cambridge: MIT Press).

MINSKY, M. AND PAPERT, S. (1969).
Perceptrons. (Cambridge MA : MIT Press).

MITCHELL, T.M. (1979).
An analysis of generalization as a search problem. *In: INTERNATIONAL JOINT CONFERENCE IN ARTIFICIAL INTELLIGENCE 6. Proceedings.*

MITCHELL, T.M., KELLER, R.M. AND KEDAR-CABELLI, S.T. (1986).
Explanation based generalization: A unifying view.
Machine Learning (Netherlands) 1(1): 47-80.

MITCHELL, T.M., UTGOFF, P.E., AND BANERJI, R. (1983).
In: Machine learning : An artificial intelligence approach, ed. R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Palo Alto, CA: Tioga Press).

MITEV, N.N and WALKER, S. (1985).
Designing an online public access catalogue: OKAPI, a catalogue on a local area network. (London : British Library Information Reasearch Report 39).

MONDSCHEIN, L.G. (1990).
Documentation Note: "Selective dissemination of information (SDI): Relationship to productivity in the corporate R & D Environment".
Journal of Documentation 46(2): 137-145.

MONTGOMERY C.A. (1981)
Where do we go from here? *In*: Information retrieval research.ed.R.N. Oddy, S.E. Robertson, C.J. van Rijsbergen, P.W. Williams.(London:Butterworths).

MOOERS, C.N. (1960).
The next twenty years in information retrieval.
American Documentation II (3): 229-236.

NEWELL, A. AND SIMON, H. (1976).
Computer Science as empirical enquiry: Symbols and search.
Communications of the ACM 19(3) : 113-126.

ODDY, R.N. (1977).
Information retrieval through man-machine dialogue.
Journal of Documentation 33 : 1-14.

PAICE, C.D. (1990).
Constructing literature abstracts by computer: Techniques and prospects.
Information Processing and Management 26(1): 171-186.

PAZZANI, M.J. (1989).
Explanation-based learning with weak domain theories. *In: INTERNATIONAL WORKSHOP ON MACHINE LEARNING, 6TH, CORNELL UNIVERSITY, ITHACA, NEW YORK, JUNE 26-27 1989. Proceedings.* San Mateo, CA : Morgan Kaufmann, pp. 72-74.

POLLACK, S.M. (1968).
Measures for the comparison of information retrieval systems.
American Documentation 19(4): 592-602.

POLLITT, A.S. (1986).
A rule-based system as an intermediary for searching cancer therapy literature on Medline. *In:* Intelligent Information Systems: progress and prospects, ed. by Davis. (London: ASLIB).

POLLITT, A.S. (1989).
Demonstrating MenUSE (Menu-based User Search Engine) A front-end system for searching online databases. Technical Report TR89/8: The Polytechnic of Huddersfield, School of Computing and Mathematics.

POPKIN, R.H., STROLL, A., AND KELLY, A.V. (1969).
Philosophy made simple. (London: W.H. Allen & Company, Ltd).

PORTEOUS, J. (to be published).
PhD thesis: City University, Department of Computer Science.

QUINLAN, J.R. (1983).
Learning efficient classification procedures and their application to chess and games. *In:* Machine learning: An artificial intelligence approach, ed. by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Palo Alto CA: Tioga Publishing Co).

QUINLAN, J.R. (1986).
The effect of noise on concept learning. *In*: Machine learning: An artifical intelligence Approach Volume II, ed. by R.S. Michalski, J.G. Carbonell and T.M. Mitchell. (Los Angeles CA : Morgan Kaufmann).

QUINLAN, J.R. (1986).
Induction of decision trees.
Machine Learning 1(1) : 81-106.

RADA, R. (1987).
Knowledge-sparse and knowledge-rich learning in information retrieval.
Information Processing & Management 23(3): 195-210.

RADA, R. AND BARLOW, J. (1991).
Document ranking using an enriched thesaurus.
Journal of Documentation 47(3): 240-253.

RADA, R. AND BICKNELL, E. (1989).
Ranking documents with a thesaurus.
Journal of the American Society for Information Science 40(5): 304-310.

RAGHAVAN, V.V., BOLLMAN, P. AND JUNG, G.S. (1989).
Retrieval system evaluation using recall and precision: Problems and answers (Extended Abstract). *In: INTERNATIONAL CONFERENCE OF ACM SIGIR, 12TH, CAMBRIDGE, MASS. USA, JUNE 22-28 1989. Proceedings.* New York : Association Computing Machinery, pp. 59-68.

RAGHAVAN, V.V. AND JUNG, G.S. (1989).
A machine learning approach to automatic pseudo-thesaurus construction. *In: INTERNATIONAL SYMPOSIUM ON METHODOLOGIES FOR INTELLIGENT SYSTEMS, 4TH, CHARLOTTE NORTH CAROLINA OCTOBER 12, 1989. Proceedings.* New York : Elsevier Science Publishing Co.

RANDALL, J.H. AND BUCHLER, J. (1957).
Philosophy, an introduction. (New York : Barnes & Noble, Inc.).

RASMUSSEN, J. (1980).
The Human as a systems component. In: Human interaction with computers, H.T. Smith and T.R.G. Green (eds.). (London: Academic Press).

RICH, E. (1979).
User modelling via stereotypes.
Cognitive Science 3: 329-354.

RICH, E. (1983)
Artificial intelligence. (Singapore: McGraw-Hill International Editions).

RICH, E. (1989).
Steretypes and user modelling. *In:* User Models in Dialog Systems, ed. by A. Kobsa and W. Wahlster. (Berlin : Springer-Verlag).

ROBERTSON, S.E. (1977).
Theories and models in information retrieval.
Journal of Documentation 33(2): 126-148.

ROBERTSON, S.E. (1977).
The probability ranking principle in information retrieval.
Journal of Documentation 33(4): 294-304.

ROBERTSON, S.E. (1979).
Indexing theory and retrieval effectiveness.
pp. 40-56.

ROBERTSON, S.E. (1981).
The methodology of information retrieval experiment. *In*: Information retrieval experiment, ed. K. S. Jones. (London: Butterworth & Co. (Publishers) Ltd).

ROBERTSON, S.E. (1990).
Basic information research: An essay in contribution to information UK 2000. (London : British Library Research and Development Department).

ROBERTSON, S.E. AND BOVEY, J.D. (1983).
A front-end for information retrieval experiments. Final Report to the British Library Research and Development Department. Report No. BLRDD Report No. 5807.

ROBERTSON, S.E., BOVEY, J.D., THOMPSON, C.L., AND MACASKILL, J.M. (1986).
Weighting, ranking and relevance feedback in a front-end system.
Journal of Information Science 12: 71-75.

ROBERTSON, S.E., MARON, M.E. AND COOPER, W.S. (1982).
Probability of relevance: A unification of two competing models for document retrieval.
Information Technologhy Research and Development, 1(1-21). (London: Butterworths).

ROBERTSON, S.E. AND K. SPARCK JONES (1976).
Relevance weighting of search terms.
Journal of the American Society for Information Science 27(3): 129-146

ROBERTSON, S.E. AND THOMPSON, C.L. (1989).
Weighted searching: The CIRT experiment. Prospects for intelligent retrieval.
Informatics 10: 153-165.

ROBERTSON, S.E., WALKER, S., HANCOCK-BEAULIEU, M., GULL, A., AND LAU, M. (1983).
Okapi at TREC. *In: THE FIRST TEXT RETRIEVAL CONFERENCE (TREC-1), GAITHERSBURG, MARYLAND 1993. Proceedings.* Washington D.C.: National Institute of Standards and Technology, Special Publication 500-207, U.S. Government Printing Office, pp. 21-30.

ROBERTSON, S.E., WALKER, S., JONES, S., HANCOCK-BEAULIEU, M., GULL, A. AND GATFORD, M.
Okapi at TREC-2. *In: THE SECOND TEXT RETRIEVAL CONFERENCE (TREC-2)*. To be published.

ROCCHIO, J.J. JR. (1971).
Relevance feedback in information retrieval. *In:* The smart retrieval system - Experiments in automatic document processing, ed. by G. Salton. (New Jersey: Prentice-Hall, Inc.).

RODERER, N.K. (1982).
Levels of measurement in evaluating information systems. THE AMERICAN SOCIETY FOR INFORMATION SCIENCE ANNUAL MEETING, Proceedings.

ROSENBLATT, F. (1958).
The Perceptron: A probabilistic model for information storage and organisation in the brain. *Psychological Review 65:* 386-407

ROSENBLOOM, P.S. AND NEWELL, A. (1986).
The chunking of goal hierarchies: A generalized model of practice. *In*: Machine Learning : An Artificial Intelligence Approach Volume II, ed. by R.S. Michalski, J.G. Carbonell and T.M. Mitchell. (Los Altos CA: Morgan Kauffmann Publishers, Inc).

RUMELHART, D.E., HINTON, G.E., AND WILLIAMS, R.J. (1986).
Learning internal representations by error propagation. *In:* Parallel Distributed Processing, Volumes 1 and 2, ed. by J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. (Cambridge, MA: MIT Press).

RUSSELL, B. (1946).
History of western philosophy. (London: George Allen & Unwin Ltd.).

SALTON, G. (1987).
Historical note: The past thirty years in information retrieval.
Journal of the American Society for Information Science 38(5): 375-380.

SALTON, G., BUCKLEY, C., AND SMITH, M. (1990).
On the application of syntactic methodologies in automatic text analysis.
Information Processing and Management 26(1): 73-92.

SALTON, G. AND LESK, M.E. (1968).
Computer Evalution of indexing and Test Processing.
Journal of the association for Computing Machinery 15(1): 621-640.

SALTON, G. AND MCGILL, M.J. (1983).
Introduction to modern information retrieval. (Singapore: McGraw-Hill International Book Company).

SAMMUT, C., AND BANERJI, R.B. (1986).
Learning concepts by asking questions. *In*: Machine learning: An artificial intelligence approach Volume II, ed. by R.S. Michalski, J.G. Carbonell and T.M. Mitchell. (Los Altos CA: Morgan Kauffmann Publishers, Inc).

SAMUEL, A.L. (1959).
Some studies in machine learning using the game of checkers.
IBM Journal of Research and Development 3: 221-229.

SARACEVIC, T. (1971).
Selected Results from an inquiry into testing of information retrieval systems.
Journal of the American Society For Information Sciences (USA) 23(2): 126-139.

SARACEVIC, T., KANTOR, P., CHAMIS, A.Y., TRIVISION, D. (1988A). A Study in Information seeking and retrieving. *In*: Background and methodology.
Journal of the American Society for Information Science 39(3): 165-175.

SARACEVIC, T., KANTOR, P. (1988B).
A study in information seeking and retrieving. II. Users, questions and effectiveness.
Journal of the American Society for Information Science 39(3): 176-195.

SARACEVIC, T., KANTOR, P. (1988C).
A study in information seeking and retrieving. III. Searchers, searches, and overlap.
Journal of the American Society for Information Science 39(3): 197-216.

SCHANK, R.C., KOLODNER, J.L., AND DEJONG, G. (1981).
Chapter 7 Conceptual information retrieval. *In*: Information retrieval research, ed. by R.N. Oddy, S.E. Robertson, C.J. van Rijsbergen, P.W. Williams. (London: Butterworth).

SHNEIDERMAN, B. (1987).
Designing the user interface. (Massachusetts: Addison Wesley).

SHOVAL, P. (1985).
Principles, procedures and rules in an expert system for information retrieval.
Information Processing and Management 21 (6): 475-487.

SIMON, H.A. (1983).
Why Should machines learn?, *In*: Machine learning: An artifical intelligence approach, ed. by R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. (Palo Alto, CA: Tioga Press).

SMADJA, F. (1993).
Retrieving collocations from text: Xtract.
Computational Linguistics 19(1): 143-177.

SMEATON, A.F. (1990).
Natural language processing and information retrieval.
Information Processing and Management 26(1): 19-20.

SMITHSON, S. (1989).
The evaluation of information retrieval systems : A case study approach. *In: PROSPECTS FOR INTELLIGENT RETRIEVAL, KINGS COLLEGE CAMBRIDGE 21-23 MARCH. Proceedings.* London: ASLIB.

SONNENWALD, D.H. (1992).
Developing a theory to guide the process of designing information retrieval systems. *In: International ACM SIGIR Conference on Research and Development in Information Retrieval, 15th, June 21-24, 1992. Proceedings.* pp. 310-317.

SPARCK JONES, K. AND WEBSTER (1979).
Research relevance weighting. (Computer Laboratory, University of Cambridge: British Library Research and Development Report 5553).

SPARCK JONES, K. (1981a).
Chapter 12 Retrieval systems tests, 1958-1978. *In*: Information retrieval experiment. (London: Butterworths & Co. (Publishers) Ltd).

SPARCK JONES, K. (1981b)
Chapter 13 The Cranfield tests. *In*: Information retrieval experiment. (London: Butterworths & Co. (Publishers) Ltd).

SPARCK JONES, K. (1981c).
Information retrieval experiment. (London: Butterworths & Co. (Publishers) Ltd).

SPARCK JONES, K. (1989).
Realism about user modeling. *In:* Symbolic computation: User models in dialog systems, ed. by A. Kobsa, W. Wahlster. (Berlin: Springer-Verlag).

SPARCK JONES, K. (1990).
Retrieving information or answering questions? (London: The Eighth British Library Annual Research Lecture 1989. The British Library Research & Development Department).

SU, L. (1989).
An investigation to find appropriate measures for evaluating interactive information retrieval. In: Annual meeting of the ASIS, 52nd, Washington D.C. 1989. Proceedings. Katzer J and Newby G.B.

SULLIVAN, M.V., BORGMAN, C.L. AND WIPPERN, D. (1990).
End-users, medicated searches, and front-end assistance programs on dialog: A comparison of learning, performance, and satisfaction.
Journal Of The American Society For Information Science 41(1): 27-42.

SWETS, J.A. (1963).
Information retrieval systems.
Science (141): 245-250.

TAGUE, J.M. (1981).
The pragmatics of information retrieval experiment. *In*: Information retrieval experiment, ed. by K. Spark Jones. (London: Butterworths & Co. (Publishers) Ltd).

TAGUE, S.J. (1992).
The pragmatics of information retrieval experiment, revisted.
Info-Processing-Management 1(28): 467-490.

TITUS, H.H. (1953).
Living issues in philosophy: An introductory textbook. (Second Edition, New York: American Book Company).

TOME ASSOCIATES LTD, (1988).
Publication on TOME SEARCHER and TOME SELECTOR, TOME Associates Ltd., IMO House, Northfield Ave., London W13 9SJ, England.

TURTLE, H.R. (1991).
Inference networks for document retrieval. PhD thesis: University of Massachusetts, Computer and Information Science Department.

VAN RIJSBERGEN, C.J. (1979).
Information retrieval. (Second Edition, London: Butterworth).

VAN RIJSBERGEN, C.J. (1986).
A new theoretical framework for information retrieval. *In*: F. Rabitti (ed.), 1986-ACM Conference on Research and Development in Information Retrieval. Pisa: IEI. p 194-200.

VAN RIJSBERGEN, C.J., ROBERTSON, S.E., AND PORTER, M.F. (1980).
Models in probabilistic information retrieval. British Library R&D Report no. 5587. 1980.

WALKER, S. AND DE VERE, R. (1990).
Improving subject Retrieval in online catalogues: 2. Relevance feedback and query expansion. (London : British Library Research Paper 72).

WALKER, S. AND HANCOCK-BEAULIEU, M. (1991).
OKAPI at City: An evaluation facility for interactive IR. (London: British Library Research Report 6056).

WADE, S.J. AND WILLETT, P. (1988).
INSTRUCT: a teaching package for experimental methods in information retrieval. Part III. Browsing, clustering and query expansion.
Program 22(1): 44-61.

WALKER, S. AND JONES, R.M. (1987).
Improving subject retrieval in online catalogues: 1. Stemming, automatic query spelling correction and cross reference tables. (London: British Library Research Paper 24).

WERSIG, G. AND NEVELIG, U. (1975).
The phenomena of interest to information science.
The Information Scientist 9(4): 127-140.

WIDROW, G. AND HOFF, M.E. (1960).
Adaptive switching circuits. Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record Part 4. pp. 96-104.

WILLETT, P. (1990).
Chapter 11 Laboratory experimentation: An experimentalist's viewpoint. *In*: User-orientated evaluation of information systems and services, D. Bawden. (Hants, England: Gower Publishing Company).

WILLIAMS, R.S. (1988).
Learning to program by examining and modifying cases. *In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 5TH, ANN ARBOR,USA, JUNE 12-14 1988. Proceedings.* San Mateo: Morgan Kauffmann Publishers, Inc. pp. 318-324.

WINSTON, P.H. (1975).
Learning structural descriptions from examples. *In:* The psychology of computer vision, ed. by P.H. Winston. (New York: McGraw-Hill).

WINSTON, P.H. (1984).
Artificial intelligence. (Massachusetts: Addison Wesley Publishing House).

WITTEN, I., BELL, T., AND NEVILL, C. (1993).
Models for compression in full-text retrieval systems.*In: DATA COMPRESSION CONFERENCE, SNOWBIRD, UTAH, APRIL 1991, Proceedings.* pp.23-32.

WONG, S.K.M., CAI, Y.J., AND YAO, Y.Y. (1993).
Computation of term associations by a neural network. *In: INTERNATIONAL ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 16TH, PITTSBURGH, USA, JULY 1993. Proceedings. ACM,* pp.107-115.

YIP, M.K. (1981).
An expert system for document retrieval. M.S.thesis: MIT, Cambridge, Mass..

ZYTKOW, J.M. AND SIMON, H.A (1983).
A theory of historical discovery: The construction of computational models.

ZYTKOW, J.M. AND SIMON, H.A. (1986).
A theory of historical discovery: The construction of componential models. *In*: Machine learning, Volume 1 No. 1, 107-137. (Boston: Kluwer Acadeic Publishers).

The Design and Analysis of Computer Experiments

Robert John Buck

as fulfillment for a Ph. D. at
City University, London, U. K.

Department of Statistics

May 1993

Table of Contents

Table of Contents 2

List of Tables 5

List of Figures 6

Acknowledgements 7

Declaration of Power of Discretion 8

Abstract 9

Key to Symbols and Abbreviations 10

Chapter 1: Robust Engineering Design and Computer Experiments 11

Section 1.1: Introduction 11

Section 1.2: Computer Experiments - An Introduction 12

Section 1.3: Robust Engineering Design - An Overview 13

Section 1.4: Loss Model Approach ("Taguchi" Method) 17

Section 1.5: Response Model Approach 20

Section 1.5.1: Response Surface Methodology 20

Section 1.5.2: DACE 21

Section 1.6: References 23

Chapter 2: Spatial Statistics 26

Section 2.1: Introduction 26

Section 2.2: Statistical Model used in Thesis 27

Section 2.2.1: Best Linear Unbiased Predictor 28

Section 2.2.2: Bayes Predictor 29

Section 2.2.3: Maximum Likelihood Estimation 31

Section 2.3: Correlation Function 32

Section 2.4: Estimating Main Effects and Interactions 33

Section 2.5: Robustness 36

Section 2.6: References 37

Chapter 3: Parameter Estimation and Model Building 38

Section 3.1: Introduction 38

Section 3.2: Computing Costs for the MLE 39

Section 3.3: FORWARD Algorithm 41
Section 3.4: Examples 44
Section 3.4.1: A Known Function 44
Section 3.4.2: Circuit Simulation 51
Section 3.5: ONETIME Algorithm 57
Section 3.6: Comparison of FORWARD and ONETIME Algorithms 58
Section 3.6.1: Description of Examples 59
Section 3.6.2: Comparison Results 60
Section 3.7: Conclusion 63
Section 3.8: References 68
Chapter 4: Latin Hypercube Sampling 70
Section 4.1: Introduction 70
Section 4.1.1: Experimental Design and Computer Experiments 70
Section 4.1.2: Latin Hypercube Sampling - A Review 72
Section 4.2: Asymptotic Space Filling Property 74
Section 4.2.1: Application 74
Section 4.3: Discrepency Functions 76
Section 4.3.1: Variance as Discrepency Function 76
Section 4.3.2: Behavior of Latin Hypercube Sampling 80
Section 4.4: Conclusion 82
Section 4.5: References 82
Chapter 5: Numerical Optimization Algorithms 85
Section 5.1: Introduction 85
Section 5.2: Nelder-Mead Simplex 86
Section 5.3: Adaptive Random Search 87
Section 5.4: NPSOL 88
Section 5.5: Conclusion 90
Section 5.6: References 90
Chapter 6: Case Studies in Robust Engineering Design 92

Section 6.1: Introduction 92
Section 6.2: Modeling and Optimization 94
Section 6.3: Voltage-Shifter Circuit Example 97
Section 6.4: Output Buffer Example 106
Section 6.4.1: Results 107
Section 6.5: Discussion 126
Section 6.5.1: Polynomial Models 126
Section 6.5.2: Taguchi Approach 127
Section 6.5.3: Region Reduction 128
Section 6.5.4: Latin Hypercube Sampling 129
Section 6.5.5: Plots of Factor Effects 130
Section 6.5.6: Computation Time 130
Section 6.6: Conclusion 131
Section 6.7: References 131
Chapter 7: Discussion of Statistical Methods of RED 133
Section 7.1: Introduction 133
Section 7.2: Estimation and Noise Parameter Distribution 133
Section 7.2.1: Distribution of Noise Parameter 134
Section 7.2.2: Model Fitting Assumptions 135
Section 7.2.3: Further Topics 137
Section 7.3: Loss Functions and Optimization 138
Section 7.4: Sample Size 140
Section 7.5: Ease of Use 141
Section 7.6: System and Tolerance Design 142
Section 7.7: Physical RED vs. Computer RED 143
Section 7.8: Conclusion 144
Section 7.9: References 145
Chapter 8: Conclusions and Further Research 146
Appendix: DACE Code and User's Guide 149
Bibliography 216

List of Tables

Table 3.1: Known Function Example 45
Table 3.2: Circuit-Simulation Example 51
Table 3.3: Computation Time 64
Table 3.4: Maximum Likelihood Results 64
Table 3.5: Cross-Validation RMSE 65
Table 3.6: Change in -2 ln L for Intel Example 65
Table 3.7: Change in -2 ln L for Cubic Toy and Tat Toy Examples 66
Table 3.8: Change in -2 ln L for ATT Example 67
Table 4.1: Variance of Combined LHS 82
Table 6.1: Input factors and their ranges: Voltage-shifter circuit 99
Table 6.2: Performances for the first five experimental-design points ... 101
Table 6.3 Nominal performances and variabilities at optimal **c**: 106
Table 6.4: Influential Variables in Stage 1 111
Table 6.5: Regions of Inputs Modeled at 4 Different Stages 123
Table 6.6: Influential Variables in Stage 2 124
Table 6.7: Influential Variables in Stage 3 126

List of Figures

Figure 3.1: Known Function: Estimated Main Effects 47
Figure 3.2: Known Function: Contour Plot of Estimated Interaction ... 48
Figure 3.3: Known Function: Predicted Response vs. True Response .. 49
Figure 3.4: Circuit Simulation: Estimated Main Effects 53
Figure 3.5: Circuit Simulation: Estimated Interaction, x_4 and x_6 54
Figure 3.6: Circuit Simulation: Estimated Joint Effect, x_3 and x_6 55
Figure 3.7: Circuit Simulation: Estimated Joint Effect, x_4 and x_6 56
Figure 4.1: Standard Deviation vs. Sample Size 81
Figure 4.2: Standard Deviation vs. Volume 81
Figure 4.3: Efficiency vs. Sample Size 81
Figure 4.4: Efficiency vs. Volume 81
Figure 6.1(a): GaAs voltage-shifter circuit 98
Figure 6.1(b): Response of GaAs voltage-shifter circuit 98
Figure 6.2: Estimated main effects for Bandwidth 103
Figure 6.3: Estimated main effects for Gain 104
Figure 6.4: Estimated main effects for Voltage 105
Figure 6.5(a): Plot of Responses, T1 vs. VDN 110
Figure 6.5(b): Plot of Responses, T2 vs. VUP 110
Figure 6.5(c): Plot of Responses, T1 vs. IOL 110
Figure 6.5(d): Plot of Responses, T2 vs. IOH 110
Figure 6.6: Estimated Main Effects for ISS 112
Figure 6.7: Joint Effect Plot of T1 and P3 for ISS 113
Figure 6.8: Scatterplot for T1 vs. TF 114
Figure 6.9: Scatterplot for T2 vs. TR 115
Figure 6.10: Estimated Main Effects for ICC 116
Figure 6.11: Estimated Main Effects for TDL 117
Figure 6.12: Estimated Main Effects for VSTDL 118
Figure 6.13: Estimated Main Effects for TDH 119
Figure 6.14: Estimated Main Effects for VCTDH 120
Figure 6.15: Estimated Main Effects for VSTDH 121
Figure 6.16: Estimated Main Effects for VCTDL 122

Acknowledgements

I have received support at many levels and from many people during my studies all of which has been greatly appreciated. I would like to thank my supervisor Professor Henry Wynn for encouraging me to turn my research into a Ph.D. thesis and for his support and advice. I would also like to thank Professor Jerry Sacks for getting me started in an interesting field of research and teaching me how research in statistics gets done. I would also like to thank Professor Will Welch for his help in keeping me organized, truly an undervalued skill, and his valued instruction in statistics and computing.

Financially I was supported by a number of organizations during my work and would like to acknowledge them as well. I have been supported in the U.S. by the AT&T Affiliates Program, by Intel Corp., by AFOSR and NSF through DMS 86-09819, by NSA MDA 904-89-H-2011 and by Cray Research, Inc. through the National Center for Supercomputing Applications, University of Illinois. I have also been supported in the U.K. by the Science and Engineering Research Council and Mentor Graphics.

On a personal level, I would like to again thank Professor Henry Wynn, Professor Jerry Sacks, and Professor Will Welch for many enlightening conversations about a wide variety of topics both in and out of statistics. I would like to thank the many graduate students of the Biochemistry Department at the University of Illinois for making me an unofficial member of the department during my stay at the University. I would also like to thank Ron Bates, my colleague at City University, for being my mate and explaining the game of cricket.

Finally, I would like to thank my family for their love and total support in everything I have done. I would like to specifically thank my parents for continuing to give me the best of themselves. Most importantly I want to thank my wife, Cheryl, for her love and support and who is my closest and most cherished friend.

Declaration of Power of Discretion

I grant powers of discretion to the University Librarian to allow this thesis to be copied in whole or in part without further reference to me. This permission covers only single copies made for study purposes, subject to normal conditions of acknowledgement.

ABSTRACT

Computer simulation modeling is an important part of engineering design. The process of creating quality products under variable conditions is called robust engineering design. Frequently these simulation models are expensive to run and it can take many runs to find the appropriate parameter settings of the engineering design. To reduce the cost of robust engineering design, it has been proposed that statistical models be used to predict the results of the simulator at unobserved values of the inputs. This involves running experiments on the computer simulation models, or computer experiments. Computer experiments have no random error and frequently involve a large number of experimental factors; because of this, there are many reasons why standard prediction and design methods may not work well.

Methods from spatial statistics, frequently refered to as kriging, are used to predict new observations of the simulation model. A generalized linear model with unknown covariance parameters is used on several examples of high dimensions. The model requires estimation of the covariance function parameters and methods are described for parameter estimation and model building.

Latin hypercube sampling is used for the experimental design. Latin hypercube sampling is as easy to use as Monte Carlo sampling and has been shown to have better estimation properties. The space filling properties of Latin hypercube sampling are investigated here and shown to fill the design space more uniformly than Monte Carlo sampling.

These statistical methods are applied to two circuit simulation models. The results show that these methods work well on computer experiments and can form the basis for a methodology in robust engineering design. Although these methods have been applied to circuit design problems, the methods are applicable to a wide variety of computer simulation models.

Robust engineering design received a great deal of attention when Taguchi's ideas were introduced. An investigation into the methods that Taguchi introduced revealed some shortcomings from a statistical view. General conclusions are drawn about the efficacy of traditional Taguchi methods compared with the prefered model-based approach of the thesis.

Key to Symbols and Abbreviations

DACE - Design and Analysis of Computer Experiments. Used here as the name for a robust engineering design methodology.

FORWARD - Name of algorithm for computing maximum likelihood estimates

LHS - Latin Hypercube Sampling

LM - Loss Method

MC&B - McKay, Conover, and Beckman

ONETIME - Name of algorithm for computing maximum likelihood estimates

RED - Robust Engineering Design

RSM - Response Surface Methodology

RM - Response Method

SN - Signal - Noise

SRS - simple random sample

STW - Shoemaker, Tsui and Wu

SWMW - Sacks, Welch, Mitchell, and Wynn

$\mathbf{s}_i, S$ - i^{th} row of experimental design S

$\mathbf{x}, \mathbf{X}$ - input value for prediction, deterministic and random.

$y(\mathbf{x}), Y(\mathbf{x})$ - response variable, deterministic and random.

$\mathbf{y}_S, \mathbf{Y}_S$ - data from experiment with design S.

$\hat{y}(\mathbf{x})$ - estimate of $y(\mathbf{x})$.

n, n_D, n_N, n_r - sample size: general, inner array, outer array, random sample.

$V(\mathbf{x}, \mathbf{w}), R(\mathbf{x}, \mathbf{w})$ - Variance and Correlation function of stochastic process.

R_S - Correlation matrix for design S

$(\theta, \mathbf{p})$ - Parameters for correlation function $R(\mathbf{x}, \mathbf{w})$

β_i - i^{th} coefficient of linear model.

$E_\theta(X)$ - Expectation of X over distribution with parameter θ.

$Var_\theta(X)$ - Variance of X over distribution with parameter θ.

-Chapter 1 -

Robust Engineering Design and Computer Experiments

1.1 Introduction

The power of computers has grown exponentially over the last decade. With the increase in power, the uses for computers has expanded rapidly. This is true particularly in the area of simulation modeling, where everything from the weather to airplanes are now being modeled on computers. Many problems being modeled have features that are not well known in reality. This leads to tinkering with the code or experimentation to get a better understanding of how the system being modeled works. Although computer power has been increasing, the expense of running many of these simulation models has increased even more rapidly and running the simulators can still be an expensive exercise. Experimental design and statistical prediction models can be used to minimize the number of runs of the simulation model, just as they do in physical or "real" experiments.

Another field that has developed rapidly at the same time is research into methods for building quality products, one aspect of which is robust engineering design. Research into methods for robust engineering design delve into a wide range of statistical topics from experimental design to data exploration, estimation and prediction. This thesis investigates the application of statistics to research strategies for robust engineering design for computer simulation models. Robust engineering design is a natural application for statistical research on computer experiments because of the widespread use of computer simulation in engineering design and the importance of robust engineering design in industry today.

The thesis has been organized so that those whose main interests are in either robust engineering design or design and analysis of computer experiments but not necessarily both may look at relevant sections without missing much. The reader most interested in robust engineering design is directed to Sections 1.3-1.5 and Chapters 5-7. Readers interested more in the statistical aspects of

the design and analysis of computer experiments should look to Section 1.2 and Chapters 2-4, and 6.

Sections 1.3-1.5 give an introduction to the problem of robust engineering design and have a brief overview of the main strategies for robust engineering design. Chapter 6 gives a detailed description of a strategy for robust engineering design developed from work on computer experiments and applies it to several examples. Chapter 7 looks more closely at the underlying problems in robust engineering design and discusses how the main strategies attempt to approximate the system and find solutions.

Section 1.2 gives an introduction to computer experiments and why new methods of analysis are useful for this field of research. Chapter 2 gives an overview of an area of spatial statistics known as kriging and defines the statistical model used throughout this thesis. Chapter 3 describes some methods in parameter estimation and applies the statistical models described in Chapter 2 to some examples. Chapter 4 discusses computer experiments and experimental design, in particular the use of Latin hypercube sampling, and develops some new theory on the space filling properties of Latin hypercube sampling. Chapter 6 as mentioned applies the methods of design and analysis of computer experiments to the problem of robust engineering design.

1.2 Computer Experiments - An Introduction

An excellent discussion of statistics and computer experiments is given in Sacks, Welch, Mitchell, and Wynn[1] and a summary of their comments would be useful. Interest in computer experiments is growing and numerous examples have begun to reach the literature. Naturally, traditional statistical methods have typically been applied to these early examples, but it is useful to restate the objectives of computer experiments and examine how computer experiments may differ from physical experiments.

The single factor which differentiates computer experiments from physical experiments is random error. Computer experiments, unlike physical experiments, do not have a random error component unless it has been explicitly added to the code via a random number generator. Despite similar goals, the absence of random error creates some important differences with physical experiments.

• The classic ideas in experimental design on blocking, replication and randomization are not applicable.

- The full complexity of the computer model is measurable.
- The error of prediction is due solely to systematic bias.
- The distributional assumptions of least squares models are irrelevant.

There are a large number of objectives for computer experiments, however the following could be considered the primary ones:

1. Prediction at untried inputs.
2. Optimize the response or a function of the response.
3. Tune the code to physical data.

Prediction can be thought of as a primary objective, since if one is able to predict the simulation model accurately and inexpensively, the predictor can be used as a cheap substitute in further studies such as optimization.

If prediction is taken as the main objective, the primary statistical questions are:

1. For what values of the inputs should data be collected, and how many?
2. How should the data be used to most accurately estimate or predict new observations of the simulation model?

As mentioned classic statistical methods have been applied to computer experiments, most notably least squares fitting. Since there is no random error to mask the complexity of the simulation model and simulation models are unlikely to have simple low order polynomial response surfaces, it is not surprising that Iman and Helton[2] found many situations where the response surface was not estimated adequately by least squares models. The suggested alternative for prediction models and experimental design are discussed in Chapter 2 and 4 respectively.

1.3 Robust Engineering Design - An Overview

The problem of designing and building high quality products has been highly publicized in the last 10 to 15 years. A perceived dominance in quality products by Japanese manufacturers and the introduction of Taguchi methods during the late 1970's and early 1980's led to a flurry of activity by manufacturers to incorporate Taguchi's philosophy. It has also led to a critical review of Taguchi's methods and efforts to develop better optimization strategies.

Robust Engineering Design (RED) is the process of designing a product that will perform well under variable conditions. The variability may be due to any

of a large number of possible sources: manufacturing variability, environmental variability, or product degradation over time. The design process is increasingly carried out on computers, using computer-aided design/computer-aided engineering (CAD/CAE) tools. The ideas of RED remain the same, but many of the differences between physical and computer experiments are reflected in differences between computer design and physical design problems.

The goal of researchers studying the process of RED is to develop a strategy that will make RED simple and efficient. The rest of this chapter describes the work that has been done to achieve this goal and reviews several currently popular methods. A discussion of some of the difficulties in implementing RED and how these methods attempt to resolve them can be found in Chapter 7.

All strategies in RED share many of the same ideas and definitions. First, the designer needs to define an appropriate measure of performance and create a list of variables or factors they feel will influence the performance of the product. Engineers commonly refer to these factors as parameters, but this will be avoided here because of possible confusion with the use of the word parameter as used in statistics. After observing the product's response at preselected conditions the designer can use estimates of the product's performance to select factor values to improve the product's performance under variable conditions.

Let $\mathbf{X}=(X_1,...,X_d)$ denote the d-dimensional vector of input factors the designer wishes to vary in the computer simulation model. Once the factor list, represented by $\mathbf{X}$, is obtained they need to be split into two categories, design and noise factors. *Design factors* are used by the designer to develop the product. *Noise factors* are variables that the designer has control of only during the design stage, e.g. manufacturing variability, or environmental variability. Once past the design stage these factors are not controllable and should be considered random variables. Some design factors may have variations from noise factors superimposed. We write $X_i=c_i+U_i$ to differentiate between the design factors, c_i, and noise factors, U_i, that may make up X_i. If an input factor is not a design factor, then c_i has a fixed value (make it 0) and is ignored. Similarly, if there is no noise factor component of X_i, then $U_i=0$. The performance y is, therefore, a function of $\mathbf{X}=\mathbf{c}+\mathbf{U}$, where $\mathbf{c}=(c_1,...,c_d)$ and $\mathbf{U}=(U_1,...,U_d)$.

Generally, non-additive U's can be dealt with by treating them as factors with no designable adjustment. In specific cases other routes may be available.

For example, to accommodate multiplicative variation, write $X_i = c_i U_i$, or work on a logarithmic scale to produce additive variation.

After some investigation the designer may want to subdivide the design factors into two further categories: location factors and dispersion factors. *Location factors* are design factors that have little influence on the amount of variability produced by the noise factors but affect the mean response. *Dispersion factors* are design factors that do influence product variability. One of the crucial aspects of all strategies in RED is how to identitify and investigate the interaction between dispersion factors and noise factors.

Let us investigate the mathematical formulation of the RED problem in more detail. In a physical system let

$$\mathbf{Y} = f(\mathbf{X}) + e(\mathbf{c}), \tag{1.3.1}$$

represent the outputs of the product or process under study. The output or response, $\mathbf{Y}$, is a random vector and the variability comes from two sources: $e(\mathbf{c})$ which is the random or measurement error and $\mathbf{U}$, the noise factors. Also note that the random error could be a function of the design factors and modeling $e(\mathbf{c})$ could help to improve product robustness. When using simulation models to emulate the physical system there is no measurement error and $e(\mathbf{c})$ does not need to be included in (1.3.1) and all variability is due to the noise factors.

The goal of RED is to find input factor settings so the response attains a stated target. In reality, the response is not an individual item, but a population of manufactured items operating under a range of possible conditions. Let Ω_D be the sample space for the design factors and Ω_N be the sample space for the noise factors. Then $\Omega = \Omega_D \times \Omega_N$ is the sample space for $\mathbf{X}$. RED studies are concerned with trying to find values of $\mathbf{c}$ so that all events in Ω_N attain the stated target. This is an unrealistic goal, but we can try to minimize the variation of this population around the target.

The variability of the population around the target needs to be measured so the appropriate levels for the design factors can be chosen. This variability is measured and summarized by the loss function and risk function. A function that measures the performance of a product under specific conditions is usually referred to as a *loss function*. Many features of a product's performance may make up the loss function, e.g. customer satisfaction and cost. A common

example of a loss function, $l(y)$, is the quadratic loss function, $(y-t)^2$, where y is the product response and t is the target the designer is trying to meet. The designer typically is not interested in results under specific conditions but in the product's performance under varying conditions. A function that defines how well the product performs, as measured by the loss function, under varying conditions is usually known as a *risk function* or performance measure. The goal of the designer is to minimize the risk function by proper selection of **c**. Two common forms of risk function are expected loss, $E_U(l(y))$, and maximum loss, $\max_U l(y)$. Computing the risk function requires knowing both $f(\mathbf{X})$ in (1.3.1) and the distribution of **U**. Since one or both are usually unknown the risk function is estimated and referred to as *estimated risk* or a performance statistic. When **y** is a vector, loss functions and risk functions are needed for each component of **y** and minimizing risk becomes a multivariate decision making problem.

The quadratic loss function is a commonly used loss function. When expected loss is used as the risk function in conjunction with the quadratic loss function, there are two ways to try to minimize the risk function. The risk function is the mean squared error of Y, $MSE(Y)$, and can be written as:

$$MSE(Y)=E_U(Y-t)^2=Var_U(Y)+(E_U(Y)-t)^2,$$

where t is the target and Y is a random variable because it is a function of U. One approach is to use the dispersion factors to minimize $Var_U(Y)$ and then use the location factors to adjust to target, i.e. eliminate bias. This approach in the form described is called the *dual response approach*[3] because it usually involves modeling both the mean and variance of Y. The Taguchi method could be considered a variation of the dual response approach because the approach to minimizing risk is the same: minimize variability using dispersion factors and then adjust to target with the design factors. The second approach is to minimize $MSE(Y)$ directly, in which case identification of location and dispersion factors is not essential. This approach can not be used unless the response Y is known or estimated. The implementation and benefits of these two optimization methods will be discussed briefly later in this chapter and in Chapter 7.

To summarize, the goal of the designer is to minimize the variability of the population around a target. A robust product design is found by choosing dispersion factor settings that minimize this variability and using the location

factors to keep the product response on target. All strategies in RED follow this general philosophy either explicitly or implicitly.

Strategies for RED can be classified in two general categories. Shoemaker, Tsui, and Wu (STW)[4] refer to these general approaches as the loss model (LM) approach and the response model (RM) approach. The LM approach uses an experimental plan that allows risk to be estimated directly from experimental observations. The "Taguchi" method is the classic example of the LM approach. The RM approach does not attempt to directly estimate the risk, but instead concentrates on accurate prediction of the response using statistical models, or predictors. The RM approach uses *estimates* of the response rather than observed response values at a specific product factor setting to estimate risk.

1.4 Loss Model Approach ("Taguchi" Method)

The "Taguchi" method [5] [6] is a commonly used tool in many areas where quality improvement is an issue. Kacker[7] gives a good overview of the method, while Phadke and Dehnad [8] and Box, *et al.* [9] give some good additional comments and criticisms and Kacker and Shoemaker [10] and Phadke [11] provide more examples. The "Taguchi" method is an easily implemented procedure, which helped the spread of RED ideas in industry.

The Taguchi method can be briefly summarized by a few basic ideas. An experimental plan is developed so product variability can be measured at each setting of the design factors in the experimental plan. The settings of the design factors are usually deviations from a nominal or "working" set of design factor values. The designer can determine which design factors influence product variability by identifying which design factors have a significant effect on the estimated variance over the range of input values. The settings of design factors that influence product variability are chosen to minimize the product variability. Those design factors that do not influence product variability are used to either adjust the mean product response to its performance target or to save on costs. Note that in the Taguchi method it is assumed that the location factors can be used to adjust to the performance target and this allows the performance target to be separated from the risk function.

To estimate risk directly from the experiment it is necessary to have multiple observations at each setting of the design factors. These observations are intended to mimic the variability that occurs in the real world. If noise factors

can be controlled by the designer an experimental plan can be used to determine specific settings for these noise factors. An experimental plan for noise factors increases the separation of noise factor input values allowing maximum dispersion of the noise factor settings, which implies maximum dispersion of the product response, in a minimal number of runs. If the noise factors can not be incorporated into an experimental design then replication is the only option. Putting the noise factors in an experimental plan helps to produce larger variability in the response than random sampling, but it does not lend itself to producing true estimates of product variability.

The experimental plan used to simulate "natural" variability is a product array formed of two subplans, the inner and outer arrays in Taguchi's terminology. The *inner array* is for the design factors and the *outer array* is for the noise factors. All interaction effects between design factors and noise factors are estimable when a product array is used for the experimental plan. Taguchi has published a series of orthogonal arrays for use in constructing product arrays and further work has been carried out by a large number of researchers to find more arrays. The two subplans form a product array by combining all outer array runs with each inner array run. The resulting experimental plan has a total of $n_D n_N$ runs, where n_D and n_N are the number of runs for the inner and outer arrays respectively.

Product arrays increase rapidly in size with an increase in the number of factors involved. To keep the number of runs as small as possible some design concessions are usually made. The most important concession is that the number of interaction effects between design factors or between noise factors that are estimable in the experimental design is kept to a minimum. Frequently the designs are Plackett-Burman[12] type designs, designs that are only able to estimate main effects. Transformations to produce additive models can be used to help eliminate interactions that may exist. Box [13] and Logothetis[14] discuss possible strategies.

Besides reducing the number of interactions that are estimable, replication and the number of factor levels are reduced to help keep the size of the experimental design small. This leads to the use of saturated or almost saturated designs which create special problems in uncovering important factors. There has been a considerable amount of work in this area because of the importance

of the Taguchi method and the desire to make the product array as small as possible. Berk and Picard[15] give a good overview to this area of research. The experimental plan is usually limited to 2-level orthogonal arrays since 3-level orthogonal arrays double the degrees of freedom per factor hence almost doubling the size of the experimental plan. The use of 2-level experimental plans limits the class of models that may be used to estimate the response.

Taguchi uses the Signal-Noise (SN) ratio to measure product variability. As the term implies the function is a ratio of the mean (the signal) and variance (the noise). The goal of the Taguchi method is to find settings of the dispersion factors that maximize the SN ratio, which is equivalent to minimizing the risk function. Once the experiment has been run, SN ratios are calculated for each run in the inner array. The designer can use these results to identify the dispersion factors and location factors. The dispersion factor settings are chosen to maximize the SN ratio. The location factor settings are chosen to adjust the product response to target. Usually the dispersion factor settings are selected from the finite set of experimental plan values. Discussion of the actual process of choosing location factor settings is curiously ignored in the literature. The new settings should give an immediate improvement in product robustness. Further experimentation, with the new settings as the new nominal points, can be done to make further improvements in product robustness.

Vining and Myers[3] state that Taguchi has developed over 60 variations of the Signal-Noise ratio. This is due to the need for different SN ratios for different combinations of objectives and model assumptions. León, Shoemaker, and Kacker[16] and Box[13] give a detailed review of Taguchi's SN ratio. León, Shoemaker, and Kacker[16] introduce PerMIAs, a generalized version of Taguchi's SN ratio. The conclusion that comes from this work is that one must be careful in considering which SN ratio to use otherwise the results will be of dubious value.

Vining and Myers[3] suggested the dual response approach, estimation of the mean and variance separately for each point of the inner array, to overcome many of the difficulties in using the SN ratio. This method also introduces more complicated models for estimating the response moving from ANOVA to regression techniques. They use standard linear regression models for estimating the mean and variance. Nelder and Lee[17] extend this idea by using generalized linear models to simultaneously model the mean and variance. The strategy

remains the same, identify location and dispersion factors, minimize variability and adjust to target.

1.5 Response Model Approach

Response model approaches use an estimate of the response to get an estimate of product variability. Two methods, Response Surface Methodology (RSM) and the strategy reviewed in Welch and Sacks[18] referred to here as DACE, are now currently in use.

The approaches are similar in that they both use estimates of the response to estimate product variability and they both use combined arrays for their experimental plans. Since the response rather than risk is being modeled, it is less important to have replication at the design factor settings and eliminates the need for the outer array used in the LM approach. Instead all factors, design and noise, can be put into a single experimental plan, the *combined array*. The use of combined arrays generates large savings in experimental runs.

The RM approach fits well with the CAD/CAE RED problem since there is no real product variability. Any product variability is controlled by the designer through the CAD/CAE software. Instead of using experimental runs to model variability directly, the RM approach strives to build a good predictor of the response surface then apply the variability to the predictor rather than the simulator. Since the predictor is cheaper to run than the simulator, more "replications" can be used to get an estimate of product variability when using the predictor.

The two methods differ most notably in the choice of model for estimating the response. RSM uses classic regression models while DACE uses a Gaussian stochastic process. This difference leads to different strategies in searching for optimal solutions. These differences will be discussed in detail in the overview of the two methods and in the discussion section. As described in the literature, they also differ in the approach to minimizing risk. RSM carries over the ideas of minimizing variance and adjusting to target as used in the LM approach while DACE tries to minimize $MSE(Y)$ directly. In practice either optimization approach could be used with either RSM or DACE.

1.5.1 Response Surface Methodology

Response Surface Methodology introduced by Box and Wilson[19] has a long statistical history. RSM predates the ideas of Taguchi and has thus not covered noise factors in Taguchi's sense. RSM is discussed in many texts.[20 21 22] The

extensive theory on RSM can readily be applied to RED. Myers[23] extends it to include noise factors and applies it to RED. An early application of RSM can be found in Alvarez, *et al.*[24]

RSM can be split into three stages: screening and identification, region seeking, and optimization. Small experiments are carried out to identify location and dispersion factors. The estimated linear models developed from these experiments are used to indicate in which direction the designer should search in the design factor space for design factor settings that produce a more robust product design. When a region is found which indicates no further searching is necessary, a final experiment is carried out to locate the optimal solution.

The design and analysis of the experiments are drawn from the work in linear regression models. The experimental plans typically suggested are two or three-level orthogonal arrays, frequently fractional factorials or central composite designs. The same ideas on transformations used in the LM approach are applicable here, especially to separate location and dispersion factors. The models used are classic linear regression models typically of first or second order. By the nature of the division of product factors into design and noise factors, it seems essential that three way interactions involving design and noise factors also need to be considered. STW[4] give a real example where three way interactions are significant.

If separate models are proposed for the mean response and response variance and a combined array is used for the experimental design, identifying dispersion factors can be difficult. Box and Meyer [25] and Nair and Pregibon[26] discuss methods for finding and estimating dispersion factors.

The model that is generated from screening experiments can be used to estimate product variability. This can be done easily from the the model if the noise factors are assumed to be independent random variables with mean 0 and variance σ_i^2. If the noise factors are not independently distributed it becomes much more difficult to model the variance from the regression model. Monte Carlo estimation can be used to estimate variance using the assumed noise factor distribution. The models for mean response and product variability can then be used to find improved product factor settings. This also can lead to new regions to investigate for further improvements.

1.5.2 DACE

The DACE method described in Bernardo, *et al.*[27] and outlined in Welch and Sacks[18] was developed from previous work on prediction and computer experiments which is reviewed in Sacks, Welch, Mitchell, and Wynn.[1] Examples can be found in the literature[27] and in Chapter 6.

The sequential experimentation plan can be summarized in six steps.

Step 1. Model the performance y and develop the loss function.

Step 2. Design an experiment and collect the data from the simulation runs.

Step 3. Use the data to estimate the parameters of the statistical model chosen in Step 1.

Step 4. Decompose $\hat{y}$ into effects due to individual factors and interaction effects.

Step 5. If the predictor is not accurate enough then select a smaller region where the optimal response is most likely to occur. Repeat Steps 2-5.

Step 6. When the predictor reaches a satisfactory level of accuracy optimize the chosen performance measure. Do a confirmatory run. Return to Step 5 if necessary.

The aim is to scan a large region of the input space for likely solutions and focus on these regions during subsequent experimentation to produce more accurate models of the response surface.

The experimental plan is generated using Latin Hypercube Sampling (LHS) which was introduced by McKay, Conover, and Beckman[28] for use in computer experiments. LHS is a readily implemented experimental plan with good space filling properties. Chapter 4 contains a discussion of some theoretical aspects of LHS.

For the statistical model we use a stochastic process of the form

$$Y(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x})\beta_i + Z(\mathbf{x}).$$

Where $Z(\mathbf{x})$ is a stochastic process with mean zero and correlation

$$Corr(Z(\mathbf{x}),Z(\mathbf{w})) = R(\mathbf{x},\mathbf{w})$$

between the responses at two inputs $\mathbf{x}$ and $\mathbf{w}$ and variance

$$Var(Z(\mathbf{x})) = \sigma^2.$$

A discussion of this model is given in Sacks, Welch, Mitchell, Wynn[1] (SWMW) and an overview is given in Chapter 2. There are many possible choices for R, some are discussed in Chapter 2. The correlation function used in the above work is

$$R(\mathbf{x},\mathbf{w}) = \prod_i exp(-\theta_i |x_i - w_i|^{p_i})$$

The stochastic process interpolates between design points and computes estimates of the response according to the correlation matrix R and the observations. No prior assumptions about interactions and nonlinearities need to be made.

The initial stages may not generate models that can predict the response accurately enough for the optimization procedure to pinpoint product factor settings. The predictor should be accurate enough to eliminate regions where the solution will not be found and subsequent regions will be selected to eliminate these areas from consideration. The plots in Step 3 listed above give visual information about the relationship between product factors and the response and can help guide the selection of a new subregion. A few reductions in the size of the region under study will produce an accurate predictor and optimization algorithms can then be used to locate the product factor settings that meet design specifications.

It is feasible to estimate product variability directly from the statistical model used in DACE by integrating over the noise factors using the relevant distribution. In most cases it will be more practical to use a Monte Carlo estimate of the variance or MSE to estimate product variability. An estimate of the distribution of the noise factors needs to be fully described to compute the estimate in either case. The cost of prediction is inexpensive, so an estimate of product variability can use hundreds of predictions.

1.6 References

1. Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., (1989) "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409-435.
2. Iman, R. L. and Helton, J. C., (1988) "An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models," *Risk Analysis*, 8, 71-90.

3. Vining, G. G. and R. H. Myers, (1990) "Combining Taguchi and Response Surface Philosophies: A Dual Response Approach," *Journal of Quality Technology*, 22, 38-45.

4. Shoemaker, A. C., Tsui, K.-L., and Wu, C. F. J., (1991) "Economical Experimentation Methods for Robust Design," *Technometrics*, 33, 415-428.

5. Taguchi, G. and Y. I. Wu, (1979) *Introduction to Off-Line Quality Control*, Central Japan Quality Control Association.

6. Taguchi, G., (1986) *Introduction to Quality Engineering*, Asian Productivity Organization, Tokyo.

7. Kacker, R. N., (1985) "Off-line Quality Control, Parameter Design and the Taguchi Method (with discussion)," *Journal of Quality Technology*, 17, 176-209.

8. Phadke, M. S. and Dehnad, K., (1988) "Optimization of Product and Process Design for Quality and Cost," *Quality and Reliability Engineering International*, 4, 105-112.

9. Box, G. E. P., Bisgaard, S., and Fung, C., (1988) "An Explanation and Critique of Taguchi's Contributions to Quality Engineering," *Quality and Reliability Engineering International*, 4, 123-131.

10. Kacker, R. N. and Shoemaker, A. C., (1986) "Robust Design: A Cost-Effective Method for Improving Maunfacturing Processes," *AT&T Technical Journal*, 65, 39-50.

11. Phadke, M. S., (1986) "Design Optimization Case Studies," *AT&T Technical Journal*, 65, 51-68.

12. Plackett, R. L. and Burman, J. P., (1946) "The Design of Optimum Multifactorial Experiments," *Biometrika*, 33, 305-325.

13. Box, G. E. P., (1988) "Signal to Noise Ratios, Performance Criteria and Transformations (with discussion)," *Technometrics*, 30, 1-40.

14. Logothetis, N., (1990) "Box-Cox Transformations and the Taguchi Method," *Applied Statistics*, 39, 31-48.

15. Berk, K. N. and Picard, R. R. , (1991) "Significance Tests for Saturated Orthogonal Arrays," *Journal of Quality Technology*, 23, 79-89.

16. León, R. V., Shoemaker, A. C., and Kacker, R. N., (1987) "Performance Measures Independent of Adjustment," *Technometrics*, 29, 253-265.

17. Nelder, J. A. and Lee, Y., (1991) "Generalized Linear Models for the Analysis of Taguchi-type Experiments," *Applied Stochastic Models and Data Analysis*, 7, 107-120.

18. Welch, W. J. and Sacks, J., (1991) "A System for Quality Improvement Via Computer Experiments," *Communications in Statistics-Theory and Methods*, 20, 477-496.

19. Box, G. E. P. and Wilson, K. B., (1951) "On the Experimental Attainment of Optimum Conditions (with discussion)," *Journal of the Royal Statistical Society - Series B*, 13, 1-45.

20. Box, G. E. P. and Draper, N. R., (1987) *Empirical Model Building and Response Surfaces* , John Wiley & Sons , New York, N.Y. .

21. Khuri, A. I. and Cornell, J. A., (1987) *Response Surfaces: Design and Analysis*, Marcel Dekker, New York, N.Y..

22. Box, G. E. P., Hunter, W. G., and Hunter, J. S., (1978) *Statistics for Experimenters* , John Wiley , New York .

23. Myers, R. H., (1991) "Response Surface Methodology in Quality Improvement," *Communications in Statistics-Theory and Methods*, 20, 457-476.

24. Alvarez, A. R., Abdi, B. L., Young, D. L., Weed, H. D., Teplik, J., and Herald, E. R., (1988) "Application of Statistical Design and Response Surface Methods to Computer-Aided VLSI Device Design," *IEEE Transactions on Computer-Aided Design*, 7, 272-288.

25. Box, G. E. P. and Meyer, R. D., (1986) "Dispersion Effects from Fractional Designs," *Technometrics*, 28, 19-27.

26. Nair, V. N. and Pregibon, D., (1988) "Analyzing Dispersion Effects from Replicated Factorial Experiments," *Technometrics*, 30, 247-257.

27. Bernardo, M. C., Buck, R., Liu, L., Nazaret, W. A., Sacks, J., and Welch, W. J., (1992) "Integrated Circuit Design Optimization Using a Sequential Strategy," *IEEE Transactions in Computer-Aided Design*, 11, 361-372.

28. McKay, M. D., Conover, W. J., and Beckman, R. J., (1979) "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239-245.

- Chapter 2 -

Spatial Statistics

2.1 Introduction

In Section 1.2 the nature of computer simulation models and how they invite the same types of questions as experiments in the physical world was discussed. This similarity in research questions would certainly invite the use of the same techniques. There are some differences between computer experiments and physical experiments that make the use of classic statistical techniques for computer experiments, such as linear models, suspect. The most important of these differences is that the output of a computer experiment is not affected by random error, the same input will always give the same output. These differences have been discussed in Section 1.2.

Since computer simulation models are typically fairly complex, nonlinear systems of equations it is unlikely that polynomials, especially low order polynomials favored in regression, will estimate the response surface accurately. There are several examples in the literature that emphasize this point.[1] [2] [3] Also, the discrepency between the true response surface and the predicted surface using polynomial models is due to bias and an increase in sample size will not be helpful in reducing this error. Another drawback of linear models is that they are not interpolators, i.e. a model where $\hat{y}(\mathbf{s}) = y(\mathbf{s})$ when $\mathbf{s}$ is a point in the experimental design, so estimates at the design points do not necessarily equal the response, which is known to be the true value.

The response surface which the simulation model produces is not random, however we can assume that y is a realization of a stochastic process. Then measures of uncertainty can be made. What is needed is a statistical model that has the following properties:

1. An ability to model complex surfaces.
2. The model is an interpolating predictor.
3. Developed theory and applications for realizations of random functions, or stochastic processes are available.

Statistical models from the field of spatial statistics have these properties, although the models have typically been studied in only two or three dimensions.

In Section 2.2 the statistical model used for this thesis is described. Section 2.3 is a discussion of some correlation functions that could be used with the model described in Section 2.2. Section 2.4 contains a description of a version of ANOVA main effects used with stochastic process and Section 2.5 describes some robustness properties of the model.

2.2 Statistical Model used in Thesis

The model treats the deterministic response $y(\mathbf{x})$ as a realization of a stochastic process, $Y(\mathbf{x})$, and has the form

$$Y(\mathbf{x}) = \sum_{i=1}^{k} f_i(\mathbf{x})\beta_i + Z(\mathbf{x}). \tag{2.2.1}$$

The stochastic process $Z(\mathbf{x})$ is assumed to have mean zero and covariance

$$V(\mathbf{w},\mathbf{x}) = \sigma^2 R(\mathbf{w},\mathbf{x})$$

between $Z(\mathbf{w})$ and $Z(\mathbf{x})$, where σ^2 is the process variance and $R(\mathbf{w},\mathbf{x})$ is the correlation.

Before deriving estimates for the parameters and $y(\mathbf{x})$, more notation needs to be defined. Let S be the experimental design with elements s_{ij}, $i = 1, \ldots, n$ and $j = 1, \ldots, d$, where n is the number of runs in the design and d is the number of factors in the experiment. Let $\mathbf{s}_i$ be the i^{th} row of S. Let $\mathbf{y}_S$ be the vector of responses for the design S. Let

$$R_S = \{R(\mathbf{s}_i, \mathbf{s}_j)\},\ 1 \le i \le n, 1 \le j \le n$$

be the correlation matrix for the stochastic process Z at the design sites and

$$r(\mathbf{x}) = [R(\mathbf{s}_1, \mathbf{x}), \ldots, R(\mathbf{s}_n, \mathbf{x})]'$$

be the correlations between the Z's at the design points and an untried input $\mathbf{x}$.

Let $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_k]'$ be the $k \times 1$ vector of coefficients for the linear model. Let the k functions in the regression at an untried input $\mathbf{x}$ be written as

$$f(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]'$$

and let the $n \times k$ regression design matrix be written as

$$F = \begin{bmatrix} f'(\mathbf{s}_1) \\ \vdots \\ f'(\mathbf{s}_n) \end{bmatrix}.$$

There are two methods that can be used to develop a predictor for this model, a frequentist approach or a Bayesian approach. If a Gaussian distribution is assumed, which is the case for the work in this thesis, the two approachs produce the same results given an improper prior distribution on the β's for the Bayesian approach.

2.2.1 Best Linear Unbiased Predictor

One method of analysis of this class of models is found in the field of spatial statistics and is known as kriging.[4] A current review of the subject can be found in Cressie.[5] Given a design S and the response $\mathbf{y}_S$ and assuming that the β's and σ^2 are unknown but $R(\mathbf{x},\mathbf{w})$ is known, consider the linear predictor

$$\hat{y}(\mathbf{x}) = c'(\mathbf{x})\,\mathbf{y}_S$$

of $y(\mathbf{x})$. If the frequentist view is held, one can replace $\mathbf{y}_S$ by the random vector $\mathbf{Y}_S$ and treat $\hat{y}(\mathbf{x})$ as random. The best linear unbiased predictor (BLUP) is obtained by finding $c(\mathbf{x})$, an $n\times 1$ vector, which minimizes

$$MSE[\hat{y}(\mathbf{x})] = E[c'(\mathbf{x})\mathbf{Y}_S - Y(\mathbf{x})]^2$$

subject to the unbiasedness condition

$$E[c'(\mathbf{x})\mathbf{Y}_S] = E[Y(\mathbf{x})].$$

which gives

$$c'(\mathbf{x})F - f(\mathbf{x}) = 0.$$

The MSE can be rewritten as

$$MSE[\hat{y}(\mathbf{x})] = Var(Y(\mathbf{x})) + Var(\hat{Y}(\mathbf{x})) - 2Cov(Y(\mathbf{x}), \hat{Y}(\mathbf{x}))$$

$$= \sigma^2[1 + c'(\mathbf{x})R_S c(\mathbf{x}) - 2c'r(\mathbf{x})].$$

Therefore the BLUP is found by minimizing

$$c'(\mathbf{x})R_S c(\mathbf{x}) - 2c'r(\mathbf{x})$$

subject to

$$F'c(\mathbf{x}) - f(\mathbf{x}) = 0.$$

Let $\lambda(\mathbf{x})$ a $k\times 1$ vector be the Lagrange multipliers needed for the constrained minimization of the MSE. The Lagrangian function is

$$L_M = c'(\mathbf{x})R_S c(\mathbf{x}) - 2c'r(\mathbf{x}) + \lambda'(\mathbf{x})(F'c(\mathbf{x}) - f'(\mathbf{x}))$$

and using matrix differentiation

$$\frac{\partial L_M}{\partial c(\mathbf{x})} = 2R_S c(\mathbf{x}) - 2r(\mathbf{x}) + F\lambda(\mathbf{x}) = 0.$$

Solving for $c(\mathbf{x})$ gives

(2.2.2) $$c(\mathbf{x}) = R_S^{-1}(r(\mathbf{x}) - 1/2F\lambda(\mathbf{x}))$$

and using $F'c(\mathbf{x}) = f(\mathbf{x})$ to solve for $\lambda(\mathbf{x})$ gives

$$\lambda(\mathbf{x}) = 2(F'R_S^{-1}F)^{-1}(F'R_S^{-1}r(\mathbf{x}) - f(\mathbf{x})).$$

Substituting for $\lambda(\mathbf{x})$ in (2.2.2)

$$c(\mathbf{x}) = R_S^{-1}(r(\mathbf{x}) - F(F'R_S^{-1}F)^{-1}(F'R_S^{-1}r(\mathbf{x}) - f(\mathbf{x}))$$

and the predictor is

(2.2.3) $$\hat{y}(\mathbf{x}) = f'(\mathbf{x})\hat{\beta} + r'(\mathbf{x})R_S^{-1}(\mathbf{y}_S - F\hat{\beta})$$

where $\hat{\beta} = (F'R_S^{-1}F)^{-1}F'R_S^{-1}\mathbf{y}_S$ is the generalized least squares estimate of β.

These results show that the predictor is made up of two parts: the generalized least squares predictor and an interpolator through the residuals from the generalized least squares regression model. The mean squared error for $\hat{y}(\mathbf{x})$ is given by

(2.2.4) $$MSE(\hat{y}(\mathbf{x})) = \sigma^2 \left(1 - [f'(\mathbf{x}), r'(\mathbf{x})] \begin{bmatrix} 0 & F' \\ F & R_S \end{bmatrix}^{-1} \begin{bmatrix} f(\mathbf{x}) \\ r(\mathbf{x}) \end{bmatrix} \right).$$

Typically, the correlation parameters, $(\theta, \mathbf{p})$, are not known either. Zimmerman and Cressie[6] show that (2.2.3) is an unbiased estimator of $E(Y(\mathbf{x}))$ if it is assumed that the distribution of $(Y_S, Y(\mathbf{x}))$ is symmetric about its mean and that $(\theta, \mathbf{p})$ is an even and translation-invariant function of Y. This is true when $Y(\mathbf{x})$ is assumed to have a Gaussian distribution and the estimates of $(\theta, \mathbf{p})$ are the maximum likelihood estimates. However, the estimates of $MSE(\hat{y}(\mathbf{x}))$ will be biased.

2.2.2 Bayes Predictor

To develop a Bayes predictor for the model (2.2.1) assume that $Z(\mathbf{x})$ has known covariance and that the prior distribution on β is Gaussian with mean μ and covariance $\sigma_M^2 I$, where σ_M^2 is known. For simplicity, also assume that the prior on β is independent of $Z(\mathbf{x})$. By standard theory the best Bayes predictor for $y(\mathbf{x})$ is

(2.2.5) $$\hat{y}_B(\mathbf{x}) = E[Y(\mathbf{x}) \mid \mathbf{y}_S],$$

where $E[Y(\mathbf{x})|\mathbf{y}_S]$ is the expectation of $Y(\mathbf{x})$ conditioned on the data $\mathbf{y}_S$. The multivariate distribution of $(Y(\mathbf{x}),\mathbf{Y}_S)$ is multivariate Normal with mean

$$E\begin{bmatrix}\mathbf{Y}_S\\ Y(\mathbf{x})\end{bmatrix} = \begin{bmatrix}F\mu\\ f'(\mathbf{x})\mu\end{bmatrix}$$

and covariance

$$Cov\begin{bmatrix}\mathbf{Y}_S\\ Y(\mathbf{x})\end{bmatrix} = \begin{bmatrix}C & D\\ D' & E\end{bmatrix}$$

where

$$C = Cov(\mathbf{Y}_S) = R_S + \sigma_M^2 FF',$$

$$D = Cov(Y(\mathbf{x}),\mathbf{Y}_S) = r(\mathbf{x}) + \sigma_M^2 Ff(\mathbf{x})$$

and

$$E = Var(Y(\mathbf{x})) = \sigma^2 + \sigma_M^2 f'(\mathbf{x}) f(\mathbf{x}).$$

Then the conditional mean of $Y(\mathbf{x})$ given $\mathbf{Y}_S = \mathbf{y}_S$ is

(2.2.6) $$E[Y(\mathbf{x})|\mathbf{y}_S] = f'(\mathbf{x})\mu + (r'(\mathbf{x}) + \sigma_M^2 f'F')(R_S + \sigma_M^2 FF')^{-1}(\mathbf{y}_S - F\mu)$$

The purpose of this section is to show that when $\sigma_M^2 \to \infty$, the Bayes predictor is the same as (2.2.3). To reduce the notation, let $f(\mathbf{x}) = f$ and $r(\mathbf{x}) = r$.

The inverted matrix in the second term on the right hand side of (2.2.6) can be rewritten as

$$(R_S + \sigma_M^2 FF')^{-1} = R_S^{-1} - \sigma_M^2 R_S^{-1} F (I + \sigma_M^2 F' R_S^{-1} F)^{-1} F' R_S^{-1}.$$

The inverted matrix in this equation can be rewritten as

(2.2.7) $$(I + \sigma_M^2 A)^{-1} = \frac{1}{\sigma_M^2}(I + \frac{1}{\sigma_M^2} A^{-1})^{-1} A^{-1},$$

where $A = F' R_S^{-1} F$. Since A is a positive definite matrix, for $\alpha = 1/\sigma_M^2 \to 0$

$$(I + \alpha A^{-1})^{-1} = I - \alpha A^{-1} + o(\alpha).$$

Rewriting (2.2.7) using this result, (2.2.6) can be rewritten as

(2.2.8) $$f'\mu + (r' + \sigma_M^2 f'F)(R_S^{-1} - CA^{-1}C' + \alpha CA^{-1}A^{-1}C' + o(\alpha))E,$$

where $C = R_S^{-1} F$, and $E = y_S - F\mu$. Now (2.2.8) can be rewritten as

(2.2.9) $$f'\mu + r'(R_S^{-1} - CA^{-1}C')E + \alpha r' CA^{-1}C'E + f'FCA^{-1}A^{1}C'E + o(\alpha),$$

since $\sigma_M^2 f'F(R_S^{-1} - CA^{-1}C')E = 0$. Taking the terms in (2.2.9) individually: $r'(R_S^{-1} - CA^{-1}C')E = r'R_S^{-1}y_S - r'R_S^{-1}F\hat{\beta}$, where $\hat{\beta} = (F'R_S^{-1}F)^{-1}F'R_S^{-1}y_S$, $f'FCA^{-1}A^{-1}C'E = f'\hat{\beta} - f\mu$, and $\alpha r'CA^{-1}E + o(\alpha) \to 0$ as $\alpha \to 0$. So simplifying (2.2.8) and returning to original notation gives

$$E(Y(\mathbf{x}) \mid \mathbf{Y}_S) = r'R_S^{-1}y_S - r'R_S^{-1}F\hat{\beta} + f'\hat{\beta}$$

$$= f'\hat{\beta} + r'R_S^{-1}(y_S - F'\hat{\beta})$$

as $\sigma_M^2 \to \infty$, $(\alpha \to 0)$, which is the same as the best linear unbiased predictor in (2.2.3).

2.2.3 Maximum Likelihood Estimation

The predictors described all involve model parameters. Typically, some or all of the parameters are not known and need to be estimated. There are many ways to estimate these parameters. If R_S is known, then the parameters β and σ^2 could be estimated by least squares. Several methods of estimating unknown covariances are described in Cressie.[5] A common method of estimation, which is used for the work in this thesis, is maximum likelihood estimation.

If the stochastic process is assumed to have a Gaussian distribution, then the density function is

$$(2\pi\sigma^2)^{-n/2} |R_S|^{-1/2} \exp\{\frac{-1}{2\sigma^2}(\mathbf{y}_S - F\beta)R_S^{-1}(\mathbf{y}_S - F\beta)\}$$

and the ln likelihood is

$$\ln L^* = -\frac{1}{2}[n \ln \sigma^2 + \ln |R_S| + (\mathbf{y}_S - F\beta)R_S^{-1}(\mathbf{y}_S - F\beta)/\sigma^2]. \tag{2.2.10}$$

The parameter estimates $\hat{\beta}$ and $\hat{\sigma}^2$ depend on the value of R_S hence on $(\theta, \mathbf{p})$. When R_S is known the maximum likelihood estimates of $\hat{\beta}$ and $\hat{\sigma}^2$ are

$$\hat{\beta} = (F'R_S^{-1}F)^{-1}F'R_S^{-1}\mathbf{y}_S$$

and

$$\hat{\sigma}^2 = \frac{1}{n}(y_S - F'\hat{\beta})'R_S^{-1}(\mathbf{y}_S - F'\hat{\beta}).$$

The estimate for β is the generalized least-squares estimate and $\hat{\sigma}^2$ is the standard MLE of σ^2 for a Gaussian distribution with a linear model and known covariance.

When R_S, or equivalently for the case here, when $(\theta, \mathbf{p})$ is not known maximum likelihood estimates are computed using an iterative procedure. First, an initial estimate of R_S is used to compute estimates of $\hat{\beta}$ and $\hat{\sigma}^2$. These estimates are substituted into (2.2.10) and the log likelihood can be rewritten as

$$\ln L = -\frac{1}{2}[n \ln \hat{\sigma}^2 + \ln |R_S|]. \tag{2.2.11}$$

Then (2.2.11) is minimized with respect to $(\boldsymbol{\theta}, \mathbf{p})$. These estimates of R_S are used to get new estimates of β and σ^2 and the procedure is repeated until $\ln L$ in (2.2.11) has reached a maximum. Further discussion on methods of computing maximum likelihood estimates is in Chapter 3.

2.3 Correlation Function

To compute estimates for the model given in the previous section a correlation function, $R(\mathbf{w}, \mathbf{x})$, needs to be specified. There is a large number of choices for $R(\mathbf{w}, \mathbf{x})$. The correlation function used here is from the stationary family of correlation functions, $R(\mathbf{w}, \mathbf{x}) = R(\mathbf{w} - \mathbf{x})$ and assumes that any non-stationary behavior can be modeled by the linear model part of the stochastic process. Also, we have chosen to restrict our choice of correlations to those which are products of one dimensional correlations, $R(\mathbf{w}, \mathbf{x}) = \prod R_j(w_j - x_j)$. One benefit of using this class of correlation functions is the simplification of some mathematical and computational problems. This is still a highly flexible family of correlation functions and is found to be adequate for predicting the response in most situations.

The correlation function used in the examples and study for this thesis is

$$R(\mathbf{w}, \mathbf{x}) = \prod_{j=1}^{d} \exp(-\theta_j |w_j - x_j|)^{p_j}, \tag{2.3.1}$$

where $\theta_j \geq 0$ and $1 \leq p_j \leq 2$. Other possible correlation functions are

$$R_l(\mathbf{w}, \mathbf{x}) = \prod_{j=1}^{d} (1 - \theta_j |w_j - x_j|)_+ \tag{2.3.2}$$

which gives a linear spline for the predicted response and

$$R_c(\mathbf{w}, \mathbf{x}) = \prod_{j=1}^{d} [1 - a_j (w_j - x_j)^2 + b_j |w_j - x_j|^3], \tag{2.3.3}$$

which for certain choices of a_j and b_j produce cubic spline predictors. Currin, *et al.*[3] compared the predictive ability for these correlation functions as well as others on several small examples. The empirical RMSE for the correlation

function (2.3.1) is consistently one of the best predictors of the correlation functions examined. Stein in his comments on SWMW[7] proposed the correlation function

$$\prod_{j=1}^{d} \frac{1}{\Gamma(\nu)2^{\nu-1}} (\alpha_j \mid w_j - x_j \mid)^{\nu} K_{\nu}(\alpha_j \mid w_j - x_j \mid), \tag{2.3.4}$$

where K_{ν} is a modified Bessel function of order ν. A comparison of this correlation function with (2.3.1) in the rejoinder to Stein in SWMW showed the predictive accuracy of the two correlation functions for the example tested was essentially the same.

Understanding the Parameters

The parameters in the correlation function (2.3.1) drive the predictor, especially when the linear model part of the stochastic process is assumed to be a constant. The shape of the prediction surface is not obvious from the values of the correlation parameters. The θ's and p's have different effects on the prediction surface. If $p = 2$ then the covariance function is infinitely differentiable and the prediction surface will be smooth which should be the case for most analytic functions. If $p < 2$ the covariance function is only once differentiable and the surface becomes rougher as $p \to 1$. For $p = 1$ the correlation function is a product of Ornstein-Uhlenbeck processes, which are continuous but not very smooth. Smaller p also has the effect of "inflating the value" of θ.

The meaning of the value for θ is more difficult to read. For $\theta = 0$ the variable is not significant; if $\theta = \infty$ then the variable is uncorrelated. For values of θ between zero and infinity the effect is somewhat relative to other θ's and the data. For "small" θ's the main effects are linear. As θ increases the effect of x on the response becomes more nonlinear. "Large" θ's also can imply that the variable is part of an interaction term. It is difficult to determine whether a "large" θ is due to nonlinear main effects or interactions without plotting the main effects.

2.4 Estimating Main Effects and Interactions

Since the parameter values themselves give only limited insight into the shape of the response surface, plotting the main effects and interactions of the input factors is strongly advocated. These effects are the continuous version of the effects in classic ANOVA, but instead of averaging over the data the models are integrated over the design space. The overall mean, the average of $y(\mathbf{x})$

over the experimental region, is defined to be:

$$\mu_0 = \int y(\mathbf{x}) \prod_{k=1}^{d} dx_k. \tag{2.4.1}$$

The main effect for x_i is defined to be:

$$\mu_i(x_i) = \int y(\mathbf{x}) \prod_{k \neq i} dx_k - \mu_0, \tag{2.4.2}$$

Second order interactions of x_i and x_j are

$$\mu_{ij}(x_i, x_j) = \int y(\mathbf{x}) \prod_{k \neq i,j} dx_k - \mu_i(x_i) - \mu_j(x_j) - \mu_0. \tag{2.4.3}$$

In such a way, interactions can be computed to any q-order interaction desired and $y(\mathbf{x})$ can be rewritten as

$$y(\mathbf{x}) = \mu_0 + \sum_{i=1}^{d} \mu_i(x_i) + \sum_{i<j} \mu_{ij}(x_i, x_j) + \cdots + \mu_{1\ldots d}(x_1, \ldots, x_d).$$

Just as in the discrete case, the sums of squares of the above decomposition can be written as

$$\int y^2(\mathbf{x}) = \mu_0^2 + \sum_{i=1}^{d} \mu_i^2(x_i) + \sum_{i<j} \mu_{ij}^2(x_i, x_j) + \cdots + \mu_{1\ldots d}^2(x_1, \ldots, x_d).$$

Plotting is difficult for more than 2-dimensions, but the information may be used to determine whether any higher-order interactions exist. To obtain estimates of these integrals, $y(\mathbf{x})$ can be replaced with $\hat{y}(\mathbf{x})$. Since

$$E(\hat{\mu}_I(\mathbf{x})) = E\left(\int \hat{y}(\mathbf{x}) \prod_{k \in I} dx_k\right) = \int E(\hat{y}(\mathbf{x})) \prod_{k \in I} dx_k = \int y(\mathbf{x}) \prod_{k \in I} dx_k = E(\mu_I),$$

where I is the index set of variables over which to integrate, the estimates of the main effects and interactions are unbiased. For numerical quadrature problems the overall mean, $\hat{\mu}_0$, can be used as an estimate of $\int y(\mathbf{x})\, d\mathbf{x}$. To see the effect of the input variables the integrals can be computed for m evenly spaced points and the values plotted.

These integrals are not difficult to compute. The only place where $\mathbf{x}$ occurs in the predictor (2.2.3) is in $r(\mathbf{x})$. Since $r(\mathbf{x})$ is a product of one dimensional correlation functions,

$$\int r(\mathbf{x}) \prod_{k \in I} dx_k = \prod_{k \in I} \int r(\mathbf{x}) dx_k.$$

This property reduces the cost and increases the numerical accuracy in

computing the estimates of main effects and interactions.

For large simulation models it is cumbersome to plot all the possible main effects and interactions, a total of $d(d+1)/2$ plots. To reduce the number of plots, ANOVA - like tables are used to determine important effects. These tables are constructed as follows:

1. Variation around the overall mean, $\int (Y-\mu_0)^2 d\mathbf{x}$, can be estimated by taking a random sample of size n_r and computing

$$SS(\hat{Y}) = \frac{1}{n_r}\sum_{k=1}^{n_r}(\hat{y}(\mathbf{x}_k)-\mu_0)^2.$$

   We have found that a random sample of size $n_r = 1000$ is a good compromise between efficiency and accuracy in obtaining an estimate of variation. We are not estimating variability in the statistical sense, but are trying to estimate the amount of fluctuation of the response surface about μ_0 and in that sense it is similar to the total sums of squares in ANOVA.

2. Let m equal the number of points for which $\mu_i(\mathbf{x}_k)$ are computed. For all $i = 1, \ldots, d$, estimate the corresponding squared integral by

$$SS(\mu_i) = \frac{1}{m}\sum_{k=1}^{m}[\mu_i(\mathbf{x}_k)]^2.$$

3. Repeat Step 2. for as many q-order interactions as desired.

4. For all the sums of squares computed in Steps 2 and 3 compute the ratio of $SS(\mu_i)/SS(\hat{Y})$.

Like discrete ANOVA sums of squares, the sums of squares can be decomposed for the continuous case, but

$$SS(\hat{Y}) \neq \sum_{i=1}^{d} SS(\mu_i) + \sum_{i<j} SS(\mu_{ij}) + SS(\mu_{1,\ldots,d})$$

because $SS(\hat{Y})$ is only an estimate of $\int (Y-\mu_0)^2 d\mathbf{x}$, however the decomposition should be a reasonable estimate. The sums of squares for main effects and interactions will allow the variables to be ranked in importance for their effect on the response. The ratios $SS(\mu_I)/SS(\hat{Y})$ provide a measure of importance of the effect compared to the variation in the response surface. These ratios are not the equivalent of F-tests in ANOVA, since the decomposition here is comparing the effects to the variation of the response over the input space and not the error

variance around the response. The choice of the cut-off value is a subjective decision. Besides choosing which effects to plot, the ratios can be used to search for higher-order interactions if so desired.

2.5 Robustness

Since the model parameters are unknown and finding a good predictor is the goal, we would like the prediction error to be robust to misspecified model parameters. In general, the linear model part of the stochastic process is reduced to a constant, β_0. This in turn is estimated by the mean of the data, $\bar{y}$, the robustness properties of which are well studied. When the linear model term is β_0 the robustness of the covariance function parameters, $(\theta,\mathbf{p})$ is important. There are two small scale studies that provide insight into the robust properties of the covariance parameters.

Sacks, Schiller, and Welch[8] carried out a small robustness study where $\theta_1 = \cdots = \theta_d = \theta$ and $p_1 = \cdots = p_d = 2$ to find optimal experimental designs using integrated mean squared error (IMSE)

$$J_\theta(S,\hat{Y}) = \frac{1}{\sigma^2}\int E_\theta(\hat{Y}(\mathbf{x}) - Y(\mathbf{x}))^2\,dx.$$

as the optimization criterion. The study also contains evidence for the robustness of the correlation function parameters. Two studies are carried out, for d=2 and d=7. In both cases $Y(\mathbf{x})$ is the realization of a Gaussian stochastic process. Both studies show that the *IMSE* is reasonably robust to misspecified θ, especially if θ is underestimated.

A small empirical study in two dimensions is described in Welch, *et al.*[9] In this study a deterministic function was used as the example and the maximum likelihood estimates were computed. The MLE values of θ and $\mathbf{p}$ were perturbed separately to see how the changes affected the empirical root mean squared error (ERMSE). Changing $\mathbf{p}$ from its MLE value of 2.0 to 1.0 only increased the ERMSE from 5.5% to 8% of the range of Y and even at $\mathbf{p}=1.8$ the ERMSE increased only marginally. The changes to θ were similar to those from the previous study which showed that underestimation (even up to an order of magnitude) had limited effect on the ERMSE while overestimation of θ by an order of magnitude increased ERMSE from 5.5% to 12% of the range of Y.

2.6 References

1. Iman, R. L. and Helton, J. C., (1988) "An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models," *Risk Analysis*, 8, 71-90.

2. Yu, T. K., Kang, S. M., Sacks, J., and Welch, W. J., (1991) "Parametric Yield Optimization of CMOS Analogue Circuits by Quadratic Statistical Circuit Performance Models," *International Journal of Circuit Theory and Applications*, 19, 579-592.

3. Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D., (1991) "Bayesian Prediction of Deterministic Functions, with Application to the Design and Analysis of Computer Experiments," *Journal of the American Statistical Association*, 86, 953-963.

4. Matheron, G., (1963) "Principles of Geostatistics," *Economic Geology*, 58, 1246-1266.

5. Cressie, N., (1991) *Statistics for Spatial Data*, John Wiley & Sons , New York, N.Y. .

6. Zimmerman, D. L. and Cressie, N., (1991) "Mean Squared Prediction Error in the Spatial Linear Model with Estimated Covariance Parameters," *Annals of the Institute of Statistical Mathematics*, 43, 27-43.

7. Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., (1989) "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409-435.

8. Sacks, J., Schiller, S. B., and Welch, W. J., (1989) "Design for Computer Experiments," *Technometrics*, 31, 41-47.

9. Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D., (1992) "Screening, Predicting, and Computer Experiments," *Technometrics*, 34, 15-25.

- Chapter 3 -

Parameter Estimation and Model Building

3.1 Introduction

In Section 2.2 we outline a model that treats the deterministic output of a computer code as the realization of a stochastic process, following SWMW.[1] A discussion of this statistical model can be found in Chapter 2. The model automatically adapts to nonlinear and interaction effects in the data. This reduces the problem of statistical model building to a problem of screening for important factors. The approaches discussed in this chapter are used to identify important variables (model building) and build a predictor (parameter estimation) without making any assumptions of linearity or additivity.

The model parameter estimates are typically computed by maximum likelihood methods with the assumption that the response is a realization of a Gaussian stochastic process. Given the correlation parameters $(\theta,\mathbf{p})$ of correlation function (2.3.1) the MLE of β is the generalized least squares estimate and the MLE of σ^2 is $\hat{\sigma}^2 = 1/n\,(\mathbf{y} - F\hat{\beta})^T R_S^{-1}(\mathbf{y} - F\hat{\beta})$. Substituting $\hat{\beta}$ and $\hat{\sigma}^2$ into the likelihood (2.2.4), the problem is to maximize

$$(3.1.1) \qquad l(\theta,\mathbf{p}) = -\frac{1}{2}(n \ln \hat{\sigma}^2 + \ln \det R_S),$$

which is a function of only the correlation parameters, $(\theta,\mathbf{p})$, and the data. Full maximum likelihood estimation of all the correlation parameters, followed by plotting of estimated main effects and interactions, could be used to identify important effects. However, if the dimension d of $\mathbf{x}$ is large, there will be many correlation parameters, and maximum likelihood is intractable or at least numerically costly.

Initially, the maximum likelihood estimates are computed for unconstrained $(\theta,\mathbf{p})$; even for small problems this is too expensive to be feasible. For example, as mentioned in Welch, *et al.*,[2] to compute the MLE for a problem with 20 input variables and an experimental plan of 50 runs it took 2 CPU hours on a Cray X-MP. One option is to use a single parameter, p, by setting $p_i = p$

for $i = 1, \ldots, d$. This reduces the number of parameters to be estimated approximately by half.

Two strategies for using the likelihood equation to get parameter estimates are described in this chapter. One strategy, which will be referred to as FORWARD, limits the number of parameters that are optimized by initially assigning all the variables to a single (θ_0, p_0), i.e. $\theta_i = \theta_0$ and $p_i = p_0$ for $i = 1, \ldots, d$, and then during a number of stages allows significant variables to have their own (θ, p) when computing the MLE. The algorithm is essentially a forward selection technique for the correlation parameters, θ_i. This method reduces the number of parameters to be estimated, but requires several, possibly many, MLE optimizations of increasing costs. The method gives good results at a reasonable cost for sets of input variables with a low proportion of important factors. The second strategy, which will be referred to as ONETIME, continually updates the parameter estimates by computing MLE for (θ_i, p_i) given the rest of the parameters are fixed.

Before describing the two algorithms some notation needs to be developed. Let $D = \{1, \ldots, d\}$ be the set of indices for the input variables and let C be a subset of D and C' the complement of C. Let $\theta_C = \{\theta_i = \theta_0 \text{ for } i \in C\}$, i.e. those variables in C will have the same parameter estimate $\hat{\theta}_c$. Also, let $\theta_{C'}$ be the set of unconstrained parameters for those variables in C'. Both algorithms are sequential in nature; let $\hat{\theta}_{C_k}$ be the parameter estimate at the k^{th} stage of the algorithm. This notation is applied to the power parameters, **p**, as well.

The FORWARD algorithm is described in Section 3.3 and two examples are presented in Section 3.4 to show how the algorithm, and the statistical model in general, performs. Section 3.5 describes the second algorithm, ONETIME, and Section 3.6 compares the performance of the two algorithms. Section 3.7 provides some further remarks. First, in Section 3.2 a brief overview of the costs of computing the maximum likelihood estimates.

3.2 Computing Costs for the MLE

Optimization for large numbers of parameters can be expensive. Computer costs can be divided into two types, time and memory. The memory costs vary little in comparison to time costs for the different algorithms so when referring to costs, time costs will be implied. The cost of computing the maximum likelihood estimate is a function of the cost due to the number of calls to the

objective function, the likelihood equation, and the length of time required to compute the objective function. The total cost of the algorithm is approximately the product of these two costs.

The cost due to the number of function calls will not be considered here. There is an extensive literature on the subject of optimization algorithms. A recent text by Zhigjavsky[3] gives a broad survey of the field. The optimization algorithm used is the AMOEBA subroutine as given in Numerical Recipes.[4] This routine is a variation of the Nelder-Mead simplex algorithm.[5] There are some overhead costs when running the optimization algorithm, but these costs are greatly outweighed by the cost of computing the likelihood.

There are several specialized algorithms for computing MLE. Marshall and Mardia[6] and Kitanidis[7] propose methods for covariance functions which are linear in their parameters. Zimmerman[8] discusses methods for making computations easier for regularly spaced data. Dietrich and Osborne[9] develop algorithms for a restricted set of covariance functions with the assumption that the range parameters are known. Vecchia[10] shows that the likelihood can be approximated by the product of likelihoods from subsets of the data. This technique requires anisotropic parameters to be known or non-existant for the algorithm to be effective. None of these methods are applicable given our choice of design and covariance function, so we have continued to use the more general optimization approach already mentioned to compute maximum likelihood estimates.

We discovered that for the correlation function (2.3.1), the generation of the covariance matrix, R, is responsible for well over half the cost of computing an individual likelihood value which means that it is the cause of over half the cost of computing the MLE. The reason for this is twofold. First, the correlation matrix requires on $O(dn^2)$ arithmetic operations. Secondly, when computing unconstrained estimates of the parameters the correlation matrix needs to be computed at each step of the optimization. It is clear that the computation of MLE's for this model quickly becomes prohibitively expensive, especially since the sample size, n, increases as d increases. The methods described in the next section attempt to reduce computational costs by looking at near-optimal estimates of model parameters.

3.3 FORWARD Algorithm

The basic ideas of the FORWARD algorithm are as follows. Initially, in the correlation function (2.3.1) we set $\theta_1 = \cdots = \theta_d = \theta_0$ and $p_1 = \cdots = p_d = p_0$. so that numerical maximization of the likelihood is only over θ_0 and p_0. As the factors have the same scales, this reflects prior belief that all factors are on the same footing. (Alternatively, knowledge that particular factors are active could be used to shortcut the first few of the stages to be described.)

At each stage, let C denote the set of indices of factors constrained to share *common* values of θ_j and of p_j, while the remaining factors are allowed their own values of θ_j and p_j. Starting with $C = \{1, \ldots, d\}$, the algorithm iterates in the following way. For each j in C in turn, we remove the constraint $\theta_j = \theta_0$ and $p_j = p_0$ and maximize the log likelihood (3.1.1) subject to $\theta_i = \theta_0$ and $p_i = p_0$ for all i in $C - \{j\}$. The x_j that leads to the largest likelihood is removed from C. The procedure continues until none of the factors in C makes a large improvement in the likelihood relative to the previous stage. We now give a formal definition of the algorithm, then we discuss some variants to reduce computing time.

1. Maximize the log likelihood (3.1.1) subject to $\theta_1 = \cdots = \theta_d = \theta_0$ and $p_1 = \cdots = p_d = p_0$. and denote the maximum by l_0.
2. Set $C = \{1, \ldots, d\}$.
3. Repeat Steps 4, . . . , 7 until termination.
    4. For each j in C do:
        5. Maximize the log likelihood in equation (3.1.1) subject to $\theta_i = \theta_0$ and $p_i = p_0$ for all i in $C - \{j\}$, and denote the maximum by l_j.
    6. Let j^* denote the factor producing the largest increase, $l_j - l_0$, in the log likelihood at Step 5.
    7. IF: $l_{j^*} - l_0$ is sufficiently large then set $C = C - \{j^*\}$ and $l_0 = l_{j^*}$ (remove factor j^* from C)
       ELSE: stop, taking the estimates associated with l_0 from the previous stage.

The optimization algorithm used at Steps 1 and 5 is the Numerical Recipies routine AMOEBA; see Chapter 5 for details. This algorithm relies on choosing

several starting points to help avoid settling for only a local optimum. This is a characteristic of many optimization routines. We make several, usually five, tries from different starting points. Even when these tries drive to the same optimum it does not guarantee that a global optimum has been found, a common problem with maximum likelihood.

The algorithm is similar in spirit to forward selection of regression variables, but the relationship between the factor effects and the introduced parameters is more subtle. In experiments with factor sparsity, we have typically found that the few 'strong' factors are the first to demand their own values of θ_j and p_j. These strong factors usually have a large estimated θ_j, and removing them from C tends to drive down the estimated common θ_0 for factors in C. If there are relatively few runs, the estimated common θ_0 can eventually become zero, suggesting that factors in C are completely inactive. On the other hand, the estimated common θ_0 may be nonzero at termination, suggesting that factors in C have a minor effect. These minor effects need not be identical just because the factors have the same correlation parameters. We have also noticed in problems where factor sparsity is absent that factors can be removed from C because they demand a zero value of θ_j and are presumably inactive.

Thus, the algorithm tends to terminate early by identifying exceptions: factors that are either exceptionally active or exceptionally inactive. In all cases, the final maximum likelihood estimates can be used to construct a predictor of $y(\mathbf{x})$. Then, estimated effects can be plotted, as outlined in Section 2.4, to identify the important factors, interactions, etc.

Discussion about the stopping criterion has been deliberately vague. From one stage to the next, two correlation parameters are introduced. A standard asymptotic likelihood ratio test would suggest that twice the improvement in the log likelihood is distributed χ^2_2, with a critical value of about 6 for 5% significance. We have found a cutoff of 5--6 reasonable in a number of applications, although there are many reasons why this should not be regarded as a proper statistical test. The common θ_0 provides another indication of termination. If its estimate becomes zero, the factors sharing the common θ_0 are apparently inactive. If $\theta_0 \neq 0$ it is not necessarily true that all variables in C are active. Main effects plots or further testing using 2 ln L can be used to indicate which variables in C are truly active. This occurs because the difference between θ_0 and

θ_i for some i in C may be too small to be significant. Finally, when a stage produces only a modest change in the likelihood, not meeting the above criterion, it is prudent to run the algorithm for at least one further stage. We have sometimes found that several small changes in the likelihood can be followed by a larger change, presumably because the model does not fit well until several factors all receive their own correlation parameters.

The algorithm as described may still require excessive computer time. At each stage, a maximum likelihood computation is performed in Step 5 for each of the (many) factors in C. An adaptation dramatically reduces computational cost further.

Steps 4 and 5 find the factor that gives maximum increase in the likelihood when given its own values of θ_j and p_j. To get an inexpensive indication of the change in the likelihood, we replace Step 5 above by a maximization of the likelihood only over θ_j, keeping all other parameters (θ_i for $i \neq j$ and p_i for all i) fixed at the values that produced l_0 at the previous iteration:

5′. Maximize the log likelihood (3.1.1) over θ_j, keeping all other parameters fixed at the values estimated at the previous iteration, and denote the maximum by l'_j.

This one-dimensional line search, or some other adaptation, is forced by practical necessity. It is obviously much cheaper than the optimization over many parameters. In a number of test examples, including those reported in Section 3.4, it introduces the same factors as the full optimization. Even if it failed to find the factor giving the greatest change in the log likelihood, the factor could still emerge at a later stage. One could also try optimizing over both θ_j and p_j, but θ_j seems to be the more important parameter.

Letting j^* denote the factor index in C that produces the largest increase $l'_j - l_0$ in Step 5′, we now perform the stage's single maximum likelihood computation involving more than one parameter. Thus, Step 6 is replaced by:

6′a. Let j^* denote the factor producing the largest increase, $l'_j - l_0$, in the log likelihood at Step 5′.

6′b. Maximize the log likelihood (3.1.1) subject to $\theta_i = \theta_0$ and $p_i = p_0$ for all i in $C - \{j^*\}$, and denote the maximum by l_{j^*}.

Note that the stopping criterion is based on the likelihood calculated at Step 6′b and not on the line search approximation in Step 5′.

Other variants which were tried include giving several factors their own θ_j and p_j values simultaneously, if more than one factor is indicated by the line searches.

At each stage of this algorithm two optimizations take place. The first is the MLE optimization for $(\theta_{C'}, p_{C'}, \theta_{C'}, p_{C'})$. The number of parameters estimated at each stage is $2k+2$ for $k \leq d$, where k is the number of parameters in C', rather than the $2d+2$ parameters for the full parameterization. The second optimization is for finding the input variable which maximizes the reduction of the likelihood (step 5′). This optimization is extremely cheap. Not only is it a one dimensional optimization problem on θ_i which usually requires no more than 10-20 function calls, the calculation of R_S is reduced to $O(n^2)$.

3.4 Examples

It is important to test the algorithms on known functions of suitable complexity so that a true measure of the algorithms success can be taken. The first example therefore takes a completely known function. In the second example we embed a real code, involving six inputs whose effects are fairly well understood, in a function of 20 inputs, where the 14 further inputs are designed to be almost inactive.

3.4.1 A Known Function

For the first example, $y(\mathbf{x})$ is a known function defined on the 20-dimensional input space $[-1/2,+1/2]^{20}$. The most important part of $y(\mathbf{x})$ is

$$5x_{12}/(1+x_1) + 5(x_4-x_{20})^2 + x_5 + 40x_{19}^3 - 5x_{19},$$

and there are very small effects from most of the remaining factors:

$$0.05x_2 + 0.08x_3 - 0.03x_6 + 0.03x_7 - 0.09x_9 - 0.01x_{10} - 0.07x_{11}$$
$$+ 0.25x_{13}^2 - 0.04x_{14} + 0.06x_{15} - 0.01x_{17} - 0.03x_{18}.$$

This function is designed to be challenging, with strong nonlinear effects and two interactions.

We will describe analyses of data from Latin hypercube sampling designs[11] of 30, 40, and 50 runs. For a discussion of Latin hypercube designs refer to Chapter 4.

We first describe the analysis of the data from the 50-run design. Taking the predictor (2.2.3) with correlation function (2.3.1) and the linear model part just a constant β, we used the algorithm of Section 3.3 as modified for the cheap line searches (Step 5′). Table 3.1 shows that the initial maximum log likelihood subject to $\theta_1 = \cdots = \theta_{20} = \theta$ and $p_1 = \cdots = p_{20} = p$ is -33.5 (-2 ln L = 67.0). Line searches over θ_j for $j = 1, \ldots, 20$ in turn, lead to a best maximum log likelihood of -27.6 when θ_{12} is unconstrained. This is similar to the value of -26.0 (-2 ln L = 52.0) given in Table 3.1 for the full maximization over θ_{12}, p_{12}, and the other 19 factors' common θ_c and p (Step 6′b). At the next stage, the line searches identify θ_{19}, and so on.

Factors With Own θ_j and p_j	$\hat{\theta}$ for Factors in C	-2 Log Likelihood	Change
--	.15	67.0	--
12	.051	52.0	15.0
12, 19	.039	43.6	8.4
12, 19, 20	.035	33.0	10.6
12, 19, 20, 4	.00032	5.4	27.6
12, 19, 20, 4, 1	.000015	-20.6	26.0
12, 19, 20, 4, 1, 5	0	-42.4	21.8

Table 3.1 Known-Function Example

The first six factors to receive their own θ_j and p_j are x_{12}, x_{19}, x_{20}, x_4, x_1, and x_5. All of these stages lead to large changes in the likelihood, satisfying our benchmark criterion of about 5--6 for twice the increase in the log likelihood. With these six factors having their own θ_j's and p_j's, the estimated common θ_c is zero, and the line searches show approximately zero change in the log likelihood for the remaining factors. Thus, having correctly identified the six important factors, the algorithm terminates. Running time is about 5 minutes on a Cray X-MP.

Clearly, the cheap line searches (Step 5′) of the algorithm in Section 3.3 correctly identify the important factors. There is also a considerable saving in

computing time: running the algorithm with the fuller search in Step 5 takes a total of nearly 2 hours of Cray X-MP CPU time.

At termination, the estimated θ_j's and p_j's are:

j	1	4	5	12	19	20
$\hat{\theta}_j$	0.021	0.036	0.000085	0.011	0.0030	0.030
$\hat{p}_j$	2.00	2.00	2.00	2.00	1.70	2.00

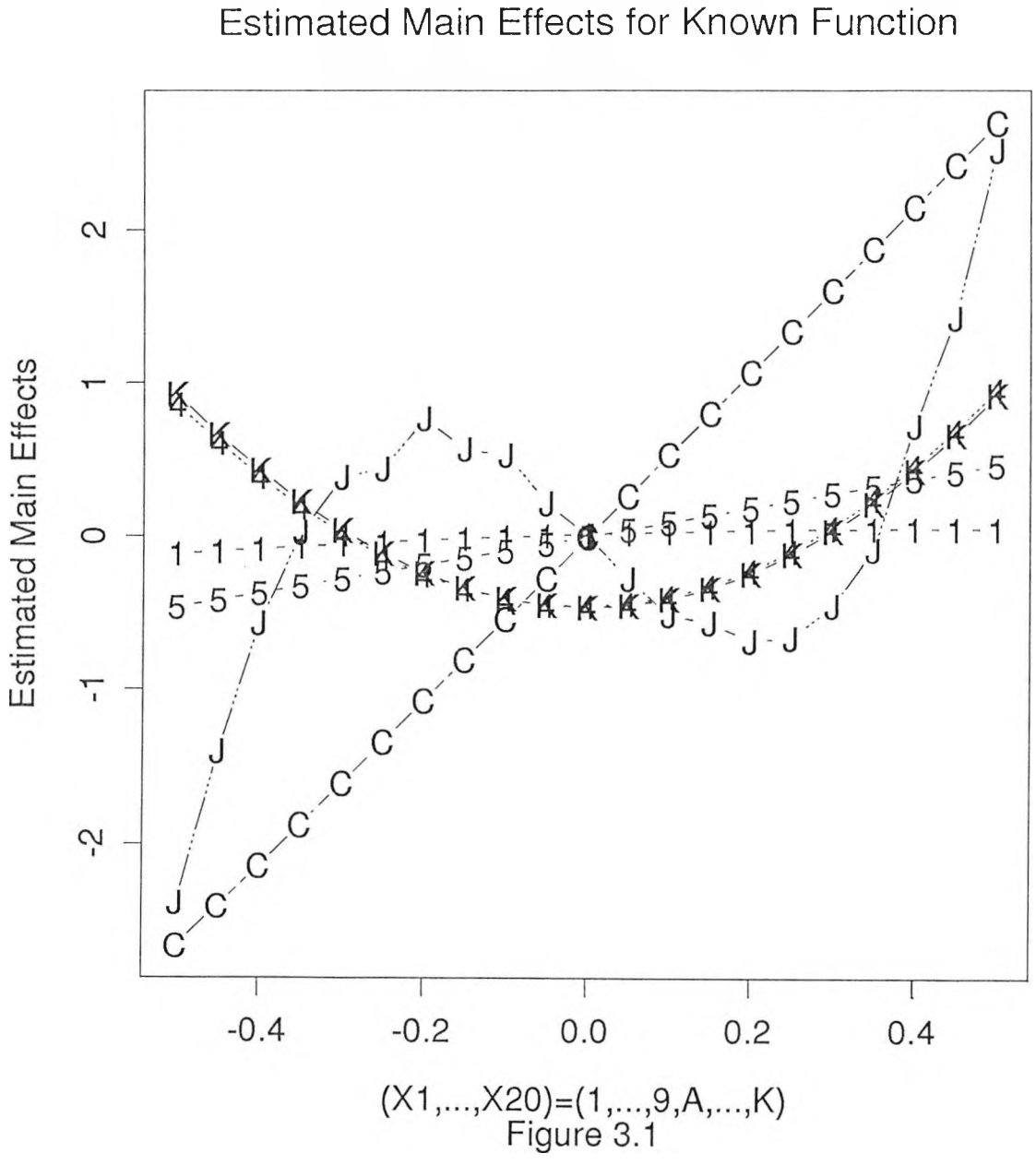
Rather than attempt to interpret these numbers it is usually more productive to plot the estimated main effects and interactions as outlined in Section 2.4.

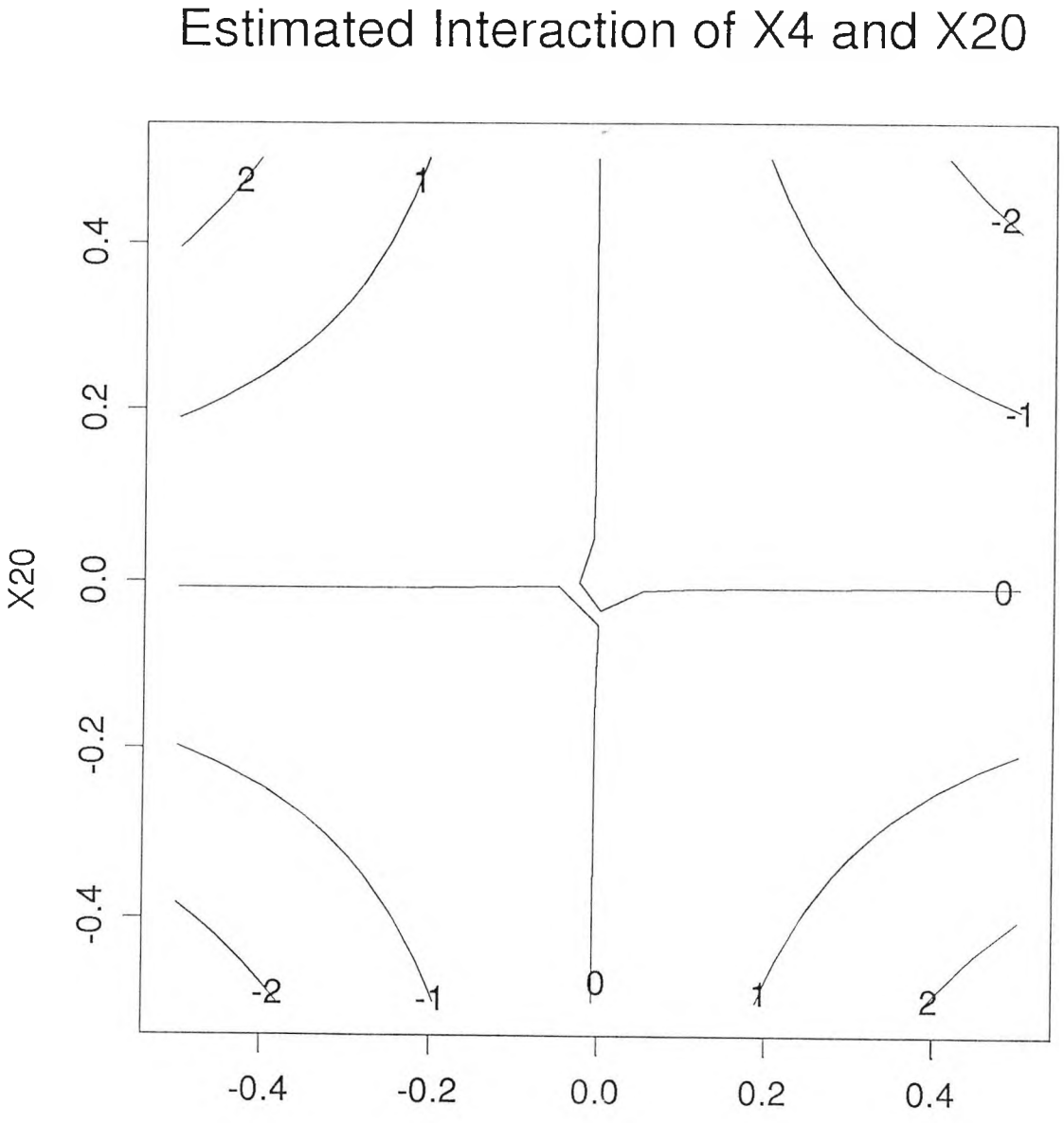
Figure 3.1 shows the estimated main effects of x_1, x_4, x_5, x_{12}, x_{19}, and x_{20}. [The remaining factors have an estimated common θ of zero, so the fitted predictor $\hat{y}(\mathbf{x})$ is constant with respect to these variables.] The quadratic effects of x_4 and x_{20} and the cubic effect of x_{19} are immediately apparent. Inspection of the estimated two-factor interaction effects for each pair of identified factors suggests large interactions between x_1 and x_{12} and between x_4 and x_{20}. For example, Figure 3.2 is the contour plot of the estimated interaction effect of x_4 and x_{20}, which agrees well with the true interaction, $-10x_4x_{20}$. The contour plot of the estimated interaction of x_1 and x_{12} has contours ranging from about -1 to 1, and is therefore also non-trivial relative to the estimated main effects in Figure 3.1. Thus, although x_1 has no main effect, it is picked up by the algorithm, and its purely interaction effect is revealed. The remaining estimated two-factor interactions are negligible: none of these plots has contours of magnitude much larger than about ± 0.1. Thus, only the two real interactions are identified.

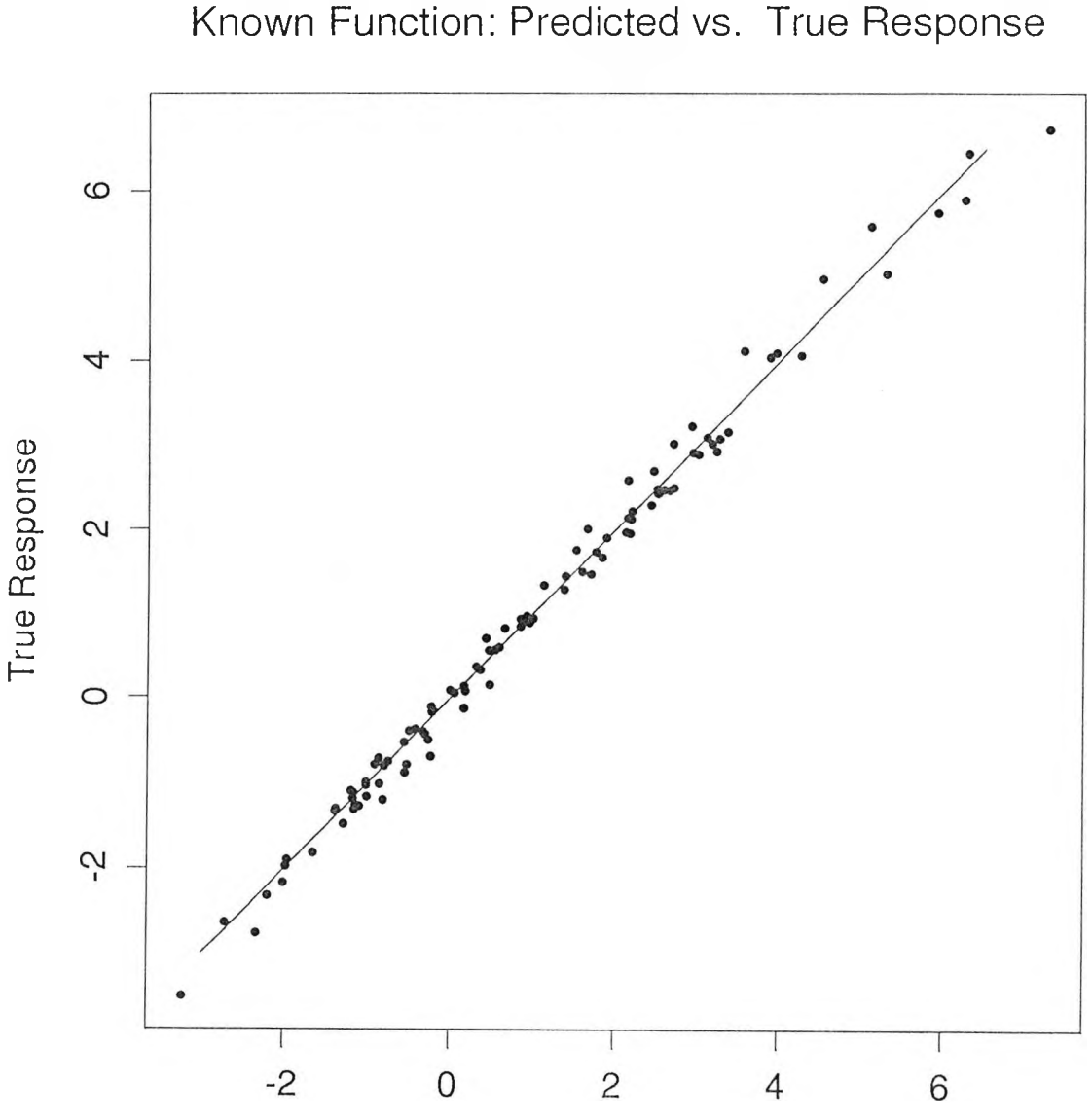
To assess the accuracy of a predictor, we now typically perform a cross validation. Let $\hat{y}_{-i}(\mathbf{x}_i)$ denote the best linear unbiased predictor (2.2.3) of $y(\mathbf{x}_i)$ based on all the data *except* the observation $y(\mathbf{x}_i)$. A cross-validation version of the empirical root mean square error (ERMSE) is then

$$\left\{\frac{1}{n}\sum [\hat{y}_{-i}(\mathbf{x}_i)-y(\mathbf{x}_i)]^2\right\}^{1/2}. \tag{3.4.1}$$

Here, the cross-validation ERMSE is 0.201, relative to a data range of about -4.4 to 8.1. To minimize computation, the MLEs of the correlation parameters are not re-computed for each prediction; they are still based on the complete data set. Nonetheless, the cross-validation ERMSE is a good measure of prediction uncertainty in this example. To show this, we generated y at 100 random points


Estimated Main Effects for Known Function
Estimated Main Effects
2
1
0
-1
-2
-0.4
-0.2
0.0
0.2
0.4
(X1,...,X20)=(1,...,9,A,...,K)
Figure 3.1


Estimated Interaction of X4 and X20
X20
0.4
0.2
0.0
-0.2
-0.4
2
1
-2
-1
0
-2
-1
0
1
2
-0.4
-0.2
0.0
0.2
0.4
X4
Figure 3.2


Known Function: Predicted vs. True Response
True Response
-2
0
2
4
6
-2
0
2
4
6
Predicted Response
Figure 3.3

in the 20-dimensional region. At these new points, the ERMSE is 0.198, very close to the cross-validation ERMSE. The plot in Figure 3.3 of $\hat{y}(\mathbf{x})$ against $y(\mathbf{x})$ at the 100 random points demonstrates the accuracy of the predictor.

To reduce computing time for the likelihood maximizations, we performed another analysis, fixing $p_1 = \cdots = p_{20} = p$ (regardless of the status of C). This roughly halves the number of correlation parameters to optimize, and reduces computing time by about 25%. The six important factors are still identified, but the accuracy of the final predictor is compromised. The ERMSE at the same 100 random points, is now .247 (versus .198 with different p_j's).

We now describe some standard screening methods applied to the data from the 50-run Latin hypercube. Fitting a first-order model in $x_1, \ldots, x_{20}$ to the ranks of y by least squares (a minor departure from the stepwise method advocated by Iman and Conover[12]), and arbitrarily taking $|t|>2$ to indicate significance, identifies only x_1 and x_{12}. The relatively unimportant x_{13} has $t=1.98$, so might also be included in practice. Repeating this with the raw y's rather than the ranks gives $|t|>2$ for the unimportant variables x_{17} and x_{18} as well. Residual analysis is not very revealing because there are several inadequacies masking each other; though, armed with foreknowledge, there is a suggestion of the cubic effect of x_{19}. Using residual analysis to cope with possible interactions would be tedious when there are many x_j's.

Without identifying the important factors, any predictor is likely to be inaccurate. Even if a quadratic model is fitted by least squares to the six important factors, the predictor remains relatively inaccurate. Fitting such a model, followed by backward elimination removing the term with the smallest $|t|$ value until $|t|>2$ for all terms, gives an ERMSE at the 100 random points of about 0.91, nearly five times as large as that for our predictor.

Alternative screening methods based on other designs might also be considered. For example, two-level, Plackett-Burman designs[13] are often used for screening, at least in physical experiments with random error. A 28-run, first-stage design (plus center point) would allow further runs to estimate interactions and quadratic effects for the important factors and still probably stay within a total of 50 runs. As with the above linear main-effects analyses, the Plackett-Burman design cannot be expected to do well, and it only finds x_{12} and x_{19}.

Applying our algorithm to the data from the n=30 or n=40 Latin hypercubes is less successful. With n=30 only x_1 and x_{12} are detected, and with n=40 only x_{12} is found. In both cases, several of the remaining important factors would be the next to enter, but they lead to small changes in the likelihood. Thus, it appears that our challenging example is too challenging without at least 50 runs. Repeating the n=40 and n=50 analyses with data from new Latin hypercubes confirms that 40 runs are inadequate but 50 runs is successful.

To summarize the first example, with 50 runs from a Latin hypercube our algorithm correctly identifies the six important factors, and the fitted model leads to a fairly accurate predictor. Fitting first-order models to the raw responses or their ranks by least squares fails to identify some of the important factors and may even find unimportant inputs to be significant. Similarly, a first-stage, Plackett-Burman design fails here. The complexity of the response relationship necessitates 50 runs for our algorithm to be successful.

3.4.2 Circuit Simulation

The second example is based on the circuit-simulation code analyzed by SWMW.[1] In their treatment, six inputs (transistor widths) are varied, and with 30 runs of the code a reasonably accurate predictor was found. Thus, the nature of the relationship between y and $x_1, \ldots, x_6$ is fairly well understood. To the output of this real code, a clock skew, a small contribution from $x_7, \ldots, x_{20}$ was added. In this way a function with a 20-dimensional input is created, where it is known which variables are important and their effects, yet the function is realistic.

A 50-run Latin-hypercube design was again tried for the 20 inputs. All inputs are normalized to [−1/2,+1/2]. To generate the data, factors $x_1, \ldots, x_6$ from the design are fed into the circuit-simulation code, and the 50 resulting clock skews are augmented with small, linear effects due to $x_7, \ldots, x_{20}$.

Table 3.2 shows that the algorithm gives factors x_5, x_3, x_6, x_2, and x_4, in that order, their own θ_j's and p_j's. Thereafter, the change in the log likelihood is rather smaller, and the algorithm terminates with these factors. Thus, all of the "real" inputs except x_1 are identified; the original SWMW[1] analysis also found x_1 to be irrelevant (to the surprise of the engineer), so the algorithm correctly identifies the five important factors.

Factors With Own θ_j and p_j	$\hat{\theta}$ for Factors in C	-2 Log Likelihood	Change
--	.014	-139.0	---
5	.0032	-147.0	8.0
5, 3	.0027	-153.8	6.8
5, 3, 6	.0067	-161.4	7.6
5, 3, 6, 2	.0018	-169.8	8.4
5, 3, 6, 2, 4	.00043	-196.6	26.8
5, 3, 6, 2, 4, 18	.00014	-201.0	4.4
5, 3, 6, 2, 4, 18, 9	.0000014	-204.6	3.6

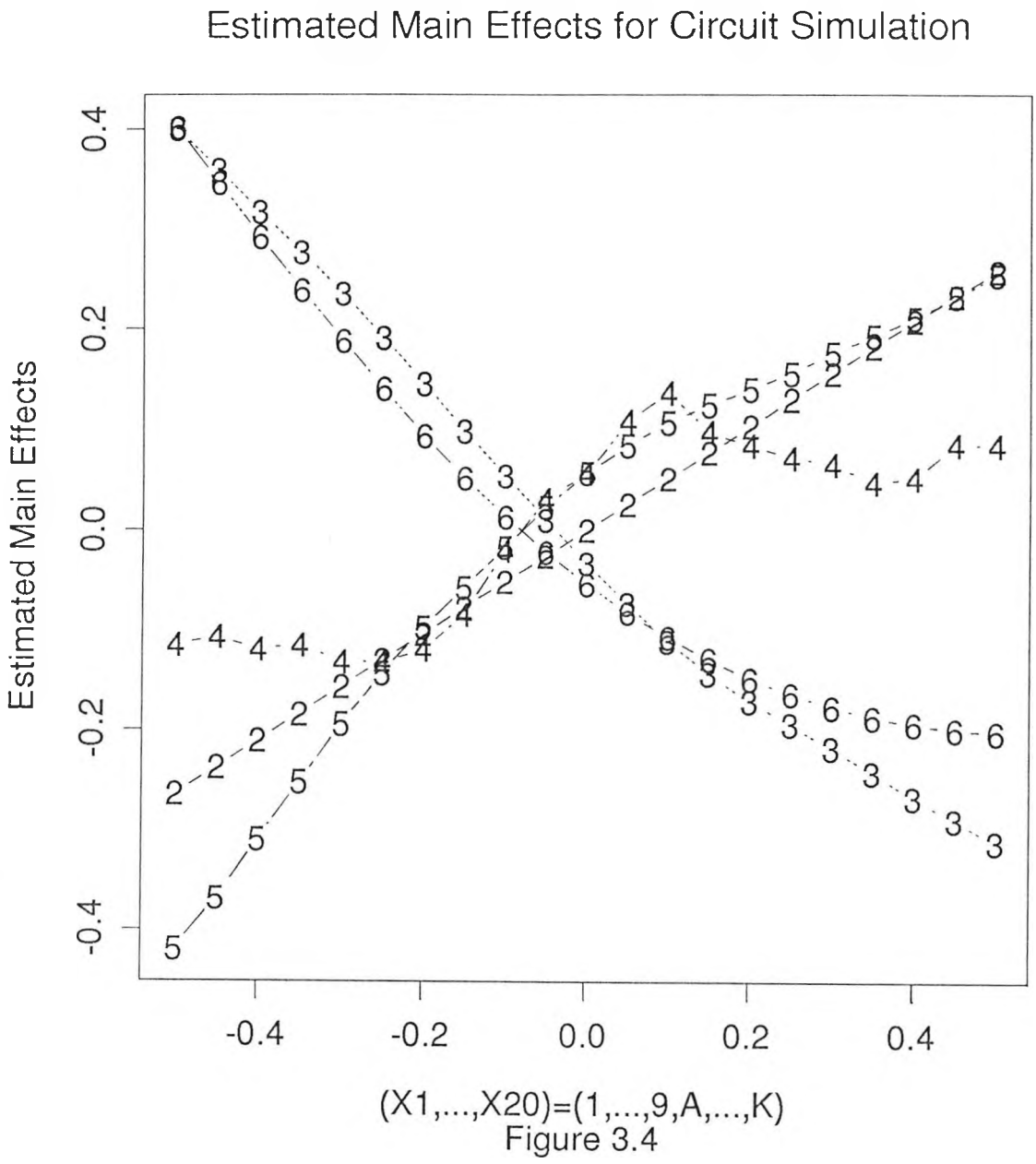
Table 3.2 Circuit-Simulation Example

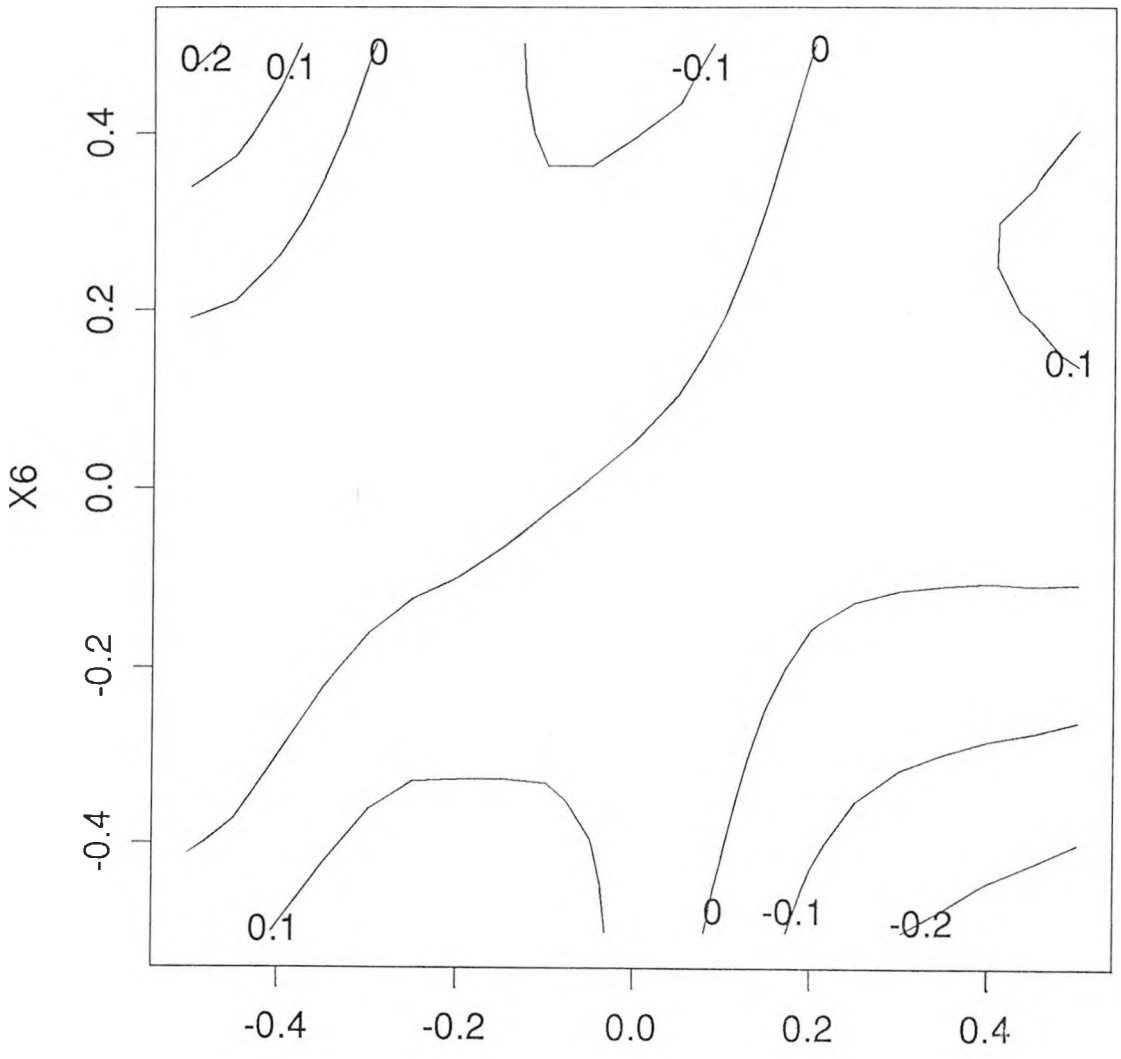
At termination, the estimated θ_j's and p_j's are:

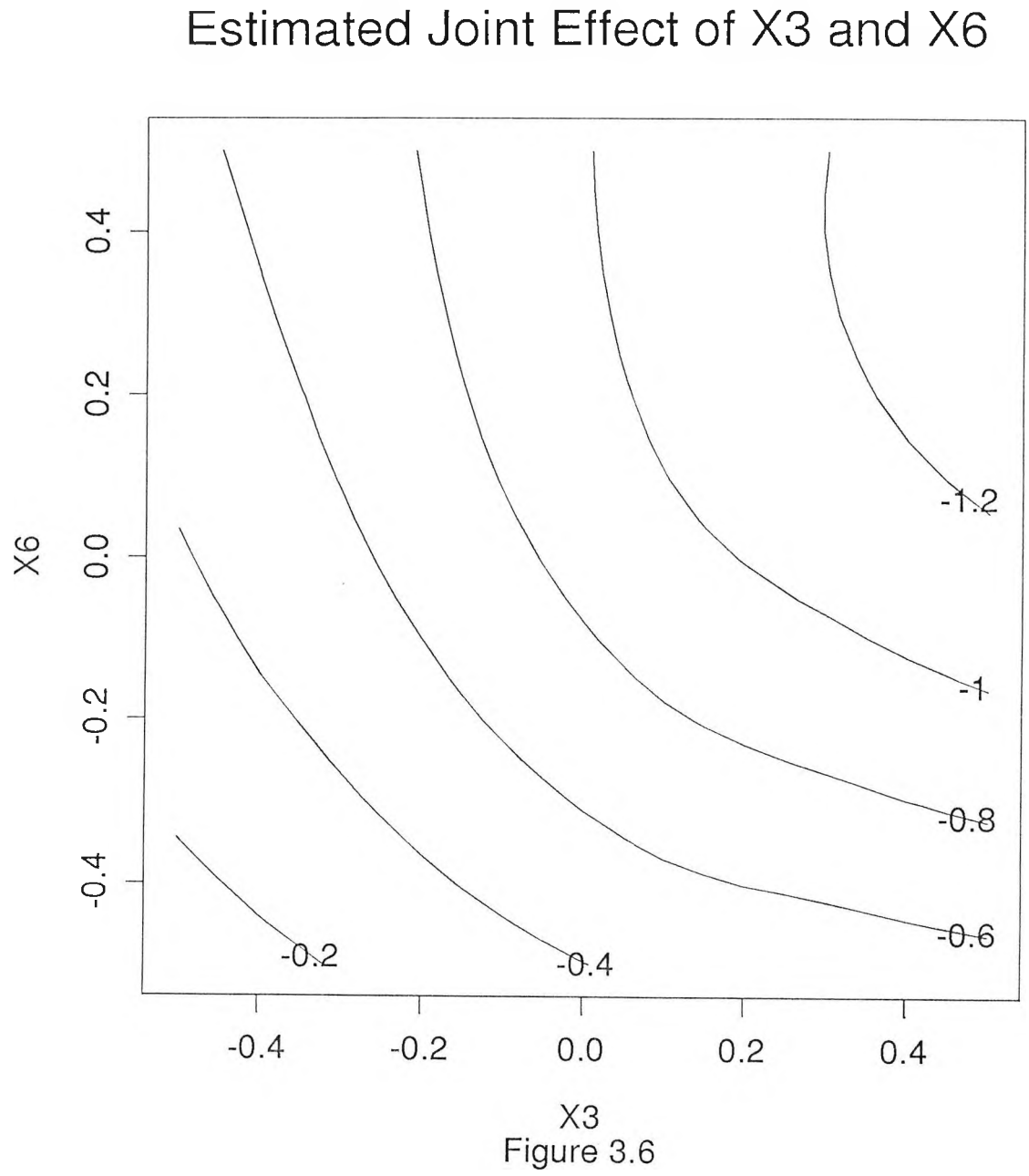
j	2	3	4	5	6
$\hat{\theta}_j$	0.035	0.058	0.15	0.065	0.97
$\hat{p}_j$	2.00	1.90	1.61	1.85	2.00

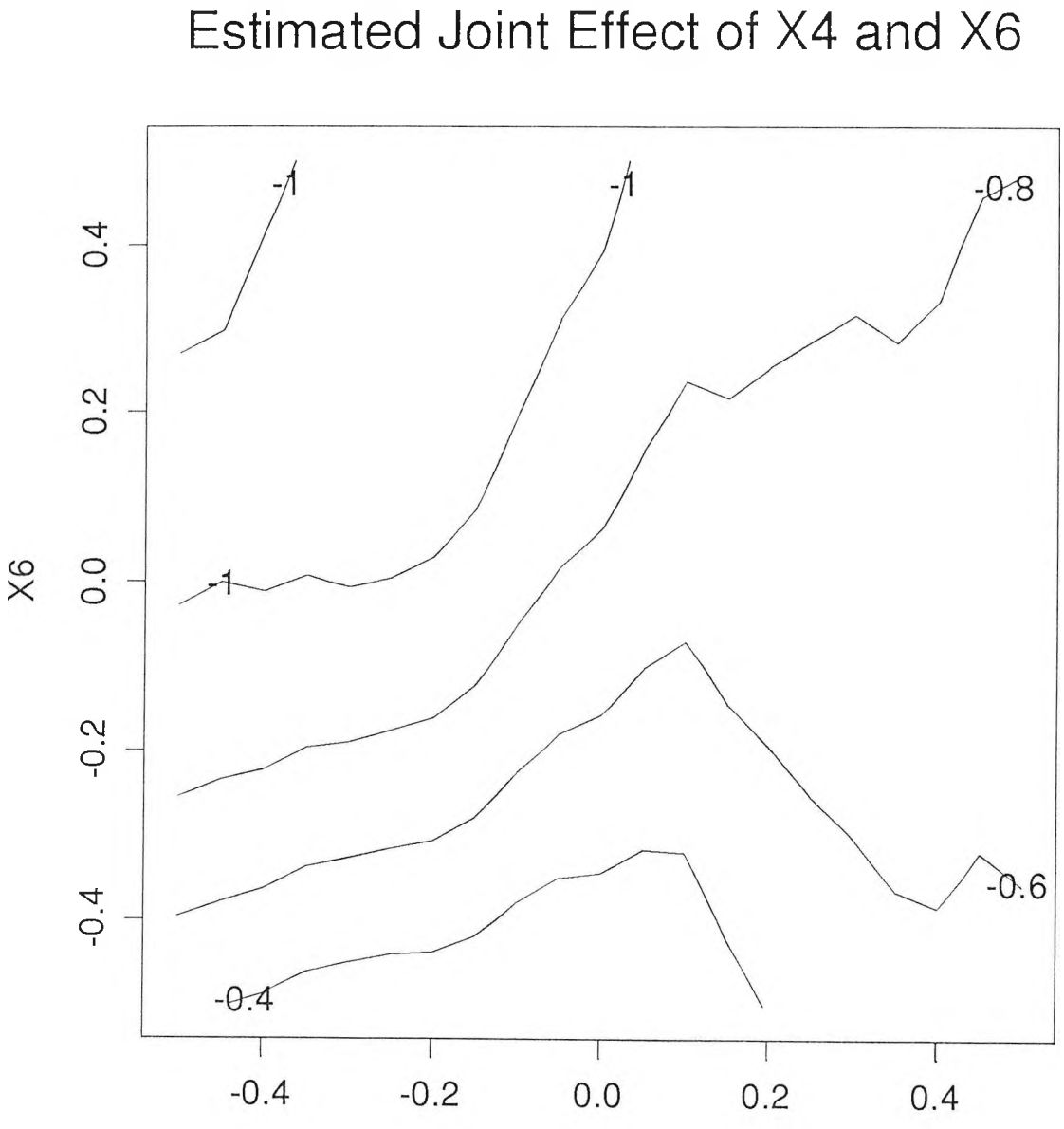
The estimated main effects are shown in Figure 3.4. Factors x_1 and $x_7, \ldots, x_{20}$, which share a common $\hat{\theta} = 0.00043$ and $\hat{p} = 1.96$, produce a blur of very small main effects in the figure.

Plots of the estimated interactions identify the interactions between x_3 and x_6 and between x_4 and x_6 as reasonably large relative to the main effects in Figure 3.4. The larger of these is the x_4—x_6 interaction, plotted in Figure 3.5, which was also found in the SWMW[1] analysis. As with standard factorial experiments, when two factors interact their joint effect on the response should be considered. Figures 3.6 and 3.7 show the estimated joint effect (overall mean plus main effects plus interaction effect) of x_3 and x_6 and of x_4 and x_6 on the clock skew. As skews close to zero are desirable, these joint effects call for small values of x_3 and x_6 with the value of x_4 less important. The main effects of factors x_2 and x_5 could also be used to bring the skew on target, and, in fact, there are many combinations of $x_2, \ldots, x_6$ that make the predicted skew close to zero.


Estimated Main Effects for Circuit Simulation
Estimated Main Effects
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
(X1,...,X20)=(1,...,9,A,...,K)
Figure 3.4


Estimated Interaction of X4 and X6
0.2
0.1
0
-0.1
0
0.1
0.1
0
-0.1
-0.2
X6
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
X4
Figure 3.5


Estimated Joint Effect of X3 and X6
X6
0.4
0.2
0.0
-0.2
-0.4
-1.2
-1
-0.8
-0.6
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
X3
Figure 3.6


Estimated Joint Effect of X4 and X6
-1
-1
-0.8
-1
-0.6
-0.4
X6
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
X4
Figure 3.7

The cross-validation ERMSE in (3.4.1) is 0.104, relative to a data range of -1.71 to -0.10. Therefore the predictor captures the major part of the variability in the function. At 100 new, random points the ERMSE is 0.102, so the cross validation is again a good indicator of prediction accuracy.

The cheap line search in Step 5′ of the algorithm in Section 3.3 is successful in identifying the important factors. Running the algorithm with the fuller search in Step 5 finds the same factors, but increases running time from about 5 minutes to over 1 hour on a Cray X-MP.

Fitting a first-order model in $x_1, \ldots, x_{20}$ to the skews, y, identifies $x_2, \ldots, x_6$ as having $|t|>2$. Using the ranks of the skews instead is less conclusive: $t=1.64$ for x_4, whereas $t=1.79$ for one of the unimportant factors. A second-order model fitted to $x_2, \ldots, x_5$, followed by backward elimination removing the term with the smallest $|t|$ value until $|t|>2$ for all terms, gives an ERMSE of 0.131 at the 100 random points, about 30% greater than that from our predictor.

Running the algorithm on data from a 30-run Latin hypercube successfully finds $x_2, \ldots, x_5$, so a smaller experiment would be adequate for screening in this case. The resulting predictor is, of course, not so accurate; it has an ERMSE of 0.167.

3.5 ONETIME Algorithm

The FORWARD algorithm has two drawbacks. The algorithm relies heavily on the assumption that a small number of variables effect the response. If this is not the case the computation of large optimization problems is still required. Also, it assumes the number of important factors is a low percentage of the total number of input factors. If the percent of significant factors is large, say more than 50% of the factors, FORWARD will screen out the unimportant factors and leave the important factors in C.

The new algorithm, ONETIME, is a numerical optimization algorithm rather than the model building algorithm of FORWARD. ONETIME reduces the calculation of the parameter estimates to a series of two dimensional optimizations over the pair (θ_i, p_i) in (θ_D, p_D). These small optimization problems are more likely to find global optima so a single starting point, the previous values for (θ_i, p_i), is sufficient. The steps in the algorithm are as follows:

Step 1.

Let $C_0 = D$ and $k = 0$ and compute $L_k^* = \max l(\theta_{C_0}, p_{C_0})$. Now let C be the empty set, so $C' = D$ and let $(\hat{\theta}_D, \hat{p}_D) = (\hat{\theta}_{C_0}, \hat{p}_{D_0})$.

Step 2.

For variable x_i, $i = 1, \ldots, d$ compute

$$L_i = \max_{(\theta_i, p_i)} l(\theta_i, p_i \mid \hat{\theta}_{D-\{i\}}, \hat{p}_{D-\{i\}}).$$

and let $(\hat{\theta}_i^*, \hat{p}_i^*)$ be the estimates that maximize L_i. Replace estimates for (θ_i, p_i) in $(\hat{\theta}_D, \hat{p}_D)$. with $(\hat{\theta}_i^*, \hat{p}_i^*)$ and let $L_{k+1}^* = L_d$.

Step 3.

Repeat Step 2 until $L_{k+1}^* - L_k^* < \varepsilon$.

This algorithm gives unconstrained estimates of $(\theta, \mathbf{p})$.

This algorithm is very similar to the FORWARD algorithm except that instead of computing the MLE over $(\beta, \sigma^2, \theta, \mathbf{p})$ we use the estimates of θ and $\mathbf{p}$ as a basis for improving the likelihood one variable at a time. The cost for Step 2 in this algorithm is only slightly larger than the cost of Step 5′ in FORWARD. The savings comes from never performing the costly step of estimating the MLE over all parameters through standard procedures. Of course this means that we can never be sure that the estimates we get are the maximum likelihood estimates of the parameters. The number of repetitions of Step 2 before convergence is dependent on the number of input variables. From the examples it appears that 15-20 iterations is adequate for most problems. Choosing the stopping rule is not straightforward; the use of the test statistic $2 \ln L$ is not advised since this is a numerical convergence algorithm. It appears that significant reduction in the likelihood value can still be made after $L_{k+1}^* - L_k^*$ approaches zero, so it is important not to stop too soon.

One can use these estimates for similar model building or screening practices as the FORWARD algorithm by testing the hypothesis $H_0 : \theta_i = 0$ for $i = 1, \ldots, d$. As in the FORWARD algorithm, $2 \ln L$ can be used as the test statistic under the assumption that it has a χ_2^2 distribution.

3.6 Comparison of FORWARD and ONETIME Algorithms

The FORWARD algorithm uses a fairly traditional approach to computing parameter estimates. The ONETIME algorithm is a numerical optimization

routine and why it should find the global optimum is an open question. Two possible explanations have to do with unimodality of the likelihood function and the starting optimization of the algorithm. If the likelihood function is unimodal then the algorithm should work quite well. It is unlikely that the likelihood function will be unimodal under all circumstances, but there is some (conflicting) evidence[14 15 16] about the prevalence of unimodality, none of which pertains to the covariance structure used here. Starting with a common (θ,**p**) and computing the MLE may also help to explain the success of the ONETIME algorithm. The MLE, ($\hat{\theta}$,$\hat{\mathbf{p}}$), behaves somewhat like a weighted average of the "important" factors (θ large) and the "unimportant" factors (θ small). The θ's for these two classes quickly diverge with the θ's for the important factors initially increasing and the θ's for the unimportant factors quickly being driven to near zero. Then it is just a matter of the important factors sorting themselves out to their respective values. From this, just the first few stages of the ONETIME algorithm could be used as a screening method and then a more traditional maximum likelihood approach could be used with the θ's for the unimportant variables set to zero.

Since the theoretical evidence available to show that the likelihood maximization of the ONETIME algorithm are MLE is extremely limited, empirical evidence must be used to show the efficiency of ONETIME. Any algorithm that uses likelihood calculations to obtain parameter estimates should accomplish three goals.

1) Find parameter estimates that have a likelihood value near the maximum likelihood when each x_i has its own unconstrained parameters (θ_i, p_i).

2) The predictive ability of these estimates should be nearly as good as the full MLE.

3) It must be cheaper than computing the actual MLE.

The algorithm is of no interest if it does not achieve the last two goals and it is difficult to determine its usefulness if the first goal is not accomplished.

Thirteen different response variables are used to compare the performance of the two algorithms. The 13 response variables span 4 different computer simulation models and 8 different experimental designs. All the experimental designs used are Latin hypercube samples.[11]

3.6.1 Description of Examples

Two of the simulation models are described thoroughly in Section 3.4. The results for the known function example in Section 3.4.1 are identified as TOY30, TOY40, TOY50 depending on the sample size and the results for the circuit example in Section 3.4.2 are labeled TAT30, and TAT50 respectively. The other 2 simulation models are also computer circuit simulation models. One is a VLSI Voltage-Shifter circuit and is referred to as the ATT example. The other is a proprietary circuit from INTEL.

In the ATT example we model 3 different responses: Voltage (VOLT), Gain (GAIN), and Bandwidth (BW). Each response is modeled using 14 input factors and a sample size of 62. This example is discussed in detail in Section 6.3.

For the INTEL example we ran two experiments using the same input factors. One experiment covered the full experimental space of the input variables and the second experiment looked at a subset of this space. For the experiment on the full space 3 response variables: TDH, VCTDL, and ICC, are used to compare the two MLE algorithms. The sample size for this experiment varied for the different responses since the responses at some input settings were not valid. The sample sizes were 63, 67 and 59 respectively. From the second experiment study two response variables, VSTDH and TDL, were used for comparision. The sample size for the responses in this experiment is 75 runs. The decision about which response variables to test for which experiment in the INTEL example was arbitrarily made, there were 8 response variables in the actual problem and it did not seem necessary to make comparisons for all 8 responses from all stages of the primary study. A more detailed description of this example is discussed in Section 6.4

3.6.2 Comparison Results

For each example the two algorithms, FORWARD and ONETIME, are compared on three points: computation time, 2 ln likelihood value, and predictive efficiency. We do not include a comparison with a true MLE calculation for two reasons. The first and simplest is that it would be prohibitively expensive, but this does not excuse the need to make the comparison. The second reason addresses that need. The FORWARD algorithm computes MLEs on a constrained set of parameters. As long as FORWARD correctly places the variables

in the sets C and C' and the factors in C are insignificant or have small effects then the MLE from the FORWARD algorithm should be nearly the same as those from an unconstrained optimization. In the two toy simulation models the important variables are known and FORWARD successfully captured all the important variables, so estimates should be essentially the same as if the parameters had been unconstrained. For the ATT and INTEL examples the important variables are not known, but the success in searching for optimal parameter values, which was the primary purpose for investigating these circuits, is supporting evidence for a good statistical model. It seems reasonable then, that if ONETIME compares favorably to FORWARD then it will compare favorably to the unconstrained MLE.

As mentioned in the introduction an algorithm needs to be able to accomplish three goals to be considered as a substitute for unconstrained maximum likelihood estimates. Two of these three criteria are easy to measure, the time it takes to reach a solution and the computed maximum likelihood value. The third criterion, predictive ability of the resulting model, is measured by one or two statistics. The two toy examples were inexpensive to run and the code was available for running further tests. For these examples Latin hypercube samples were generated with n=100 to calculate the RMSE of prediction. Random samples were not available for the two circuit simulator examples so for all four examples cross-validation RMSE of prediction was computed.

When optimizing the likelihood function one parameter at a time, as in ONETIME, parameter order may be important. To check the effect of parameter order the experimental design columns were randomly permuted to generate nine different data sets and parameter estimates are computed for each data set. The timing results for ONETIME in Table 3.3 is for one data set not the combined time for all nine permutations. All other results for ONETIME gives the best result of the nine data sets.

The timing results are in Table 3.3 and Table 3.4 gives the -2 ln likelihood values for the 13 data sets. The timing results in Table 3.3 clearly show that the ONETIME algorithm is much faster than FORWARD even if one considered the sum of all nine trials. Table 3.4 shows that the two algorithms give roughly the same likelihood value, except for TOY40 and VOLT where ONETIME did considerably better. The %OPT column of Table 3.4 gives the percentage of the 9 permutations that reach the best likelihood value. For 9 of the 13 responses the

results were stable over the permutation of design columns. Two responses with a low %OPT, TOY40 and TOY30, had considerable variation in the computed maximum likelihood values for the different permuations which is reflected in the high Coefficient of Variation, which is shown in the CV column of Table 3.4. For CUBIC TOY40, the variability is due to a major improvement in the maximum likelihood value for one of the permuted data sets. The improvement is more modest for VOLT and is more likely due to the factors in C not being homogeneous. The results show that parameter order will occasionally affect the likelihood computation, but using 4 or 5 permuted designs should capture the best result. This still leads to a considerable savings in time. The results also show that the variability in likelihood solutions between permuted data sets may be used to distinguish between stable and unstable optimizations.

The prediction and cross validation results are in Table 3.5. From the results of the two toy examples we can see that the ONETIME algorithm does only slightly worse predicting new input values than the FORWARD algorithm. One can also see that for the circuit simulation models that the two algorithms give similar cross-validation results. In all examples, except the cubic toy problem for $n=30$ and $n=40$, both algorithms give a prediction error $\leq 10\%$ of the response range. These results combined with the likelihood results are strong evidence that the ONETIME and FORWARD algorithms compute parameter estimates effectively for the examples given.

Both algorithms use -2 ln L as a test statistic, but for somewhat different purposes. FORWARD uses -2 ln L to determine which variable to add next to the model, the equivalent of an F-test in regression. The ONETIME algorithm uses the test statistic more like a multiple comparison t-test in regression. For both of these tests, assuming a χ^2_2 distribution, the critical value is 6.00. See Tables 3.6-3.8 for listing of $\Delta(-2\ln L)$ for the ONETIME algorithm. The equivalent results for the FORWARD algorithm give the same list with only a small number of exceptions, besides the failure of the FORWARD algorithm to identify important factors for TOY30 and TOY40. When the "significant" factors from the test statistic are compared with the size of their main effects plots there is strong agreement between the two measures of factor influence. The comparison does show that factors with test statistic values between 6.0 and 20.0 tend to be insignificant factors or have small linear effects. Multiple testing has some influence on overestimation of significant factors. For example,

$\chi^2_{.996} = 11.3$ would be the Bonferroni style critical value for 15 multiple comparisons and $\alpha = .95$. This brings the critical value much more into line with what the main effects show to be important factors.

3.7 Conclusion

The curse of dimensionality apparently requires a rapid increase in the number of observations as the dimension of the input grows. Yet, with factor sparsity, the problem is not nearly so bad, provided the few important factors can be identified. In this situation, the methods used here can find the important variables and can detect curvature and interactions, without explicitly modeling such effects. The simplicity of modeling the effects when using these methods carries over to the issue of experimental design as well. Since the methods do not explicitly model interactions, etc., the reasons for using orthogonal designs are diminished and with it the difficulties in choosing alias structures for experimental designs for unknown models. Fitting a useful predictor of the response is often possible without collecting further data. Cross validation seems to provide a good indication of accuracy.

Various methods based on fitting first-order models failed to find the important effects in the first example. This does not rule out the possibility that a determined application of residual analysis, etc., would improve these regression models to the point of being useful. On the other hand, searching for interactions would, at best, be fairly tedious with a large number of input factors. Moreover, the methods discussed here are data adaptive and can find interactions and other complexities in a fairly automatic way.

Computing time can be substantial for the FORWARD algorithm, but is reasonable for the ONETIME algorithm (30-45 minutes for problems given here on Sun SPARC2). Often, computer codes are themselves computationally expensive, so expensive data justify a careful analysis. Even if data are cheap to generate, it may be difficult to solve a high-dimensional problem simply by increasing the amount of data; a more intricate analysis may be necessary anyway.

EXAMPLE	FORWARD	ONETIME
CUBIC TOY50	283.6	16.9
CUBIC TOY40	181.7	9.8
CUBIC TOY30	108.5	6.1
TAT TOY50	373.9	13.9
TAT TOY30	70.5	5.6
ATT VOLT	561.4	22.1
ATT GAIN	504.8	20.7
ATT BW	505.1	13.8
INTEL TDH	349.6	15.7
INTEL VCTDL	279.4	16.4
INTEL ICC	201.9	11.6
INTEL VSTDH	350.2	16.9
INTEL TDL	604.3	24.6

Table 3.3 Computation Time (CPU seconds on Cray XMP)

EXAMPLE	FORWARD	ONETIME	%OPT	WORST	ICVI
CUBIC TOY50	-42.3	-42.9	100	-42.9	0.00
CUBIC TOY40	10.9	-11.8	11	19.2	1.06
CUBIC TOY30	7.6	7.6	33	29.4	0.54
TAT TOY50	-205.7	-209.6	78	-195.3	0.02
TAT TOY30	-115.5	-116.8	55	-114.9	0.01
ATT VOLT	-166.2	-179.6	100	-179.6	0.01
ATT GAIN	-61.9	-61.9	89	-54.0	0.04
ATT BW	-236.9	-237.2	33	-210.9	0.04
INTEL TDH	140.1	139.9	100	139.9	0.01
INTEL VCTDL	-247.7	-248.9	100	-248.9	0.00
INTEL ICC	342.8	342.6	100	342.6	0.00
INTEL VSTDH	-438.3	-440.5	100	-440.5	0.00
INTEL TDL	-344.7	-347.4	78	-335.4	0.01

Table 3.4 Maximum Likelihood Results (-2*ln Likelihood)

EXAMPLE	RANGE of Y	FORWARD	ONETIME
CUBIC TOY50	10.4	0.20 (0.20)	0.26 (0.20)
CUBIC TOY40	7.8	0.89 (1.6)	0.27 (0.31)
CUBIC TOY30	12.5	0.87 (1.10)	0.89 (0.81)
TAT TOY50	1.62	0.09 (0.10)	0.13 (0.14)
TAT TOY30	1.52	0.06 (0.16)	0.09 (0.15)
ATT VOLT	3.85	0.124	0.102
ATT GAIN	5.32	0.40	0.41
ATT BW	2.14	0.24	0.27
INTEL TDH	27.38	2.41	2.33
INTEL VCTDL	1.06	0.11	0.12
INTEL ICC	133.9	11.6	11.9
INTEL VSTDH	0.343	0.041	0.041
INTEL TDL	5.62	0.047	0.048

Table 3.5 Cross-Validation (Prediction) RMSE

INPUT	TDH	VCTDL	ICC	VSTDH	TDL
X_1	0.0	5.0	45.6	108.9	117.2
X_2	86.1	39.0	0.7	0.5	17.3
X_3	12.2	48.7	55.1	20.9	478.4
X_4	23.1	7.6	18.1	61.9	99.8
X_5	26.1	69.4	28.3	7.5	15.0
X_6	94.8	9.3	18.5	0.0	0.0
X_7	5.2	4.2	0.0	4.8	73.7
X_8	0.0	5.2	14.2	101.7	340.5
X_9	36.7	8.1	6.9	9.6	0.0
X_{10}	0.0	40.1	0.7	5.9	2.3
X_{11}	64.2	0.0	21.2	0.0	474.2

Table 3.6 Change in -2 ln L for Intel Example $\chi^2_{.95,2} = 6.0$

INPUT	TOY30	TOY40	TOY50	TAT30	TAT50
X_1	17.1	54.9	57.9	-0.5	1.4
X_2	0.0	0.0	0.5	-0.5	1.8
X_3	0.0	0.0	1.0	-0.5	2.3
X_4	34.6	67.2	139.5	33.3	84.8
X_5	9.2	35.5	63.5	62.8	142.0
X_6	0.0	0.0	0.0	-0.5	1.4
X_7	0.0	0.0	0.0	0.7	10.9
X_8	0.0	0.0	0.4	8.7	1.4
X_9	0.0	0.0	0.0	-0.5	7.5
X_{10}	2.5	0.0	0.0	-0.5	1.5
X_{11}	0.0	0.0	0.0	-0.5	4.6
X_{12}	81.6	114.9	196.1	59.2	84.4
X_{13}	0.0	0.0	0.0	-0.5	1.4
X_{14}	0.0	0.0	0.1	0.4	1.4
X_{15}	0.0	0.0	0.0	-0.5	1.4
X_{16}	0.0	0.0	0.0	-0.5	1.4
X_{17}	0.0	0.0	0.0	15.4	4.4
X_{18}	0.0	0.0	0.0	1.8	16.0
X_{19}	34.5	80.5	191.6	52.1	125.2
X_{20}	27.4	64.7	138.5	59.2	112.0

Table 3.7 Change in -2 ln L for Cubic Toy and Tat Toy Examples $\chi^2_{.95,2} = 6.0$

INPUT	VOLT	GAIN	BW
X_1	140.6	105.2	0.0
X_2	3.6	0.7	191.2
X_3	-0.4	37.6	25.9
X_4	43.9	9.2	61.0
X_5	20.9	0.0	71.6
X_6	2.6	0.0	0.0
X_7	106.7	0.0	97.2
X_8	237.2	122.2	4.2
X_9	237.9	84.3	32.5
X_{10}	86.6	0.0	0.0
X_{11}	90.1	2.1	0.0
X_{12}	-0.4	6.4	0.0
X_{13}	31.3	0.0	0.0
X_{14}	-0.4	15.9	0.0

Table 3.8 Change in -2 ln L for ATT Example $\chi^2_{.95,2} = 6.0$

3.8 References

1. Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., (1989) "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409-435.
2. Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D., (1992) "Screening, Predicting, and Computer Experiments," *Technometrics*, 34, 15-25.
3. Zhigljavsky, A. A., (1991) *Theory of Global Random Search*, Kluwer Academic Publishers.
4. Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., (1986) *Numerical Recipes*, Cambridge University Press, Cambridge.
5. Nelder, J. A. and Mead, R., (1965) "A Simplex Method for Function Minimization," *Computer Journal*, 7, 308-313.
6. Marshall, R. J. and Mardia, K. V., (1985) "Minimum Norm Quadratic Estimation of Components of Spatial Covariance," *Mathematical Geology*, 17, 517-525.
7. Kitanidis, P. K., (1985) "Minimum-Variance Unbiased Quadratic Estimation of Covariance of Regionalized Variables," *Mathematical Geology*, 17, 195-208.
8. Zimmerman, D. L., (1989) "Computationally Exploitable Structure of Covariance Matrices and Generalized Covariance Matrices in Spatial Models," *J. Statist. Comput. Simul.*, 32, 1-15.
9. Dietrich, C. R. and Osborne, M. R., (1991) "Estimation of Covariance Parameters in Kriging via Restricted Maximum Likelihood," *Mathematical Geology*, 23, 119-135.
10. Vecchia, A. V., (1988) "Estimation and Model Identification for Continuous Spatial Process," *Journal of the Royal Statistical Society - Series B*, 50, 297-312.
11. McKay, M. D., Conover, W. J., and Beckman, R. J., (1979) "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239-245.
12. Iman, R. L. and Conover, W. J., (1980) "Small Sample Sensitivity Analysis Techniques for Computer Models, With Application to Risk Assessment (with discussion)," *Communications in Statistics - Theory and*

Methods, A9, 1749-1874.

13. Plackett, R. L. and Burman, J. P., (1946) ''The Design of Optimum Multifactorial Experiments,'' *Biometrika*, 33, 305-325.
14. Mardia, K. V. and Watkins, A. J., (1989) ''On Multimodality of the Likelihood in the Spatial Linear Model,'' *Biometrika*, 76, 289-295.
15. Dietrich, C. R., (1991) ''Modality of the Restricted Maximum Likelihood for Spatial Gaussian Random Fields,'' *Biometrika*, 78, 833-840.
16. Warnes, J. J. and Ripley, B. D., (1987) ''Problems with Likelihood Estimation of Covariance Functions of Spatial Gaussian Processes,'' *Biometrika*, 74, 640-642.

- Chapter 4 -

Latin Hypercube Sampling

4.1 Introduction

Most experimental designs have been developed for physical experiments, but there are several important differences between physical and computer experiments that make these experimental design less than ideal. Computer experiments tend to have large number of input variables with large regions of interest and nonlinear response variables. These two traits create problems in choosing a strategy for experimental design. Large numbers of input variables make spreading points throughout the input space a difficult task without also making the sample sizes too large. If the function is nonlinear, then it is necessary to take more values across the range of the inputs so the function can be mapped accurately.

The first part of this section is devoted to an overview of some standard experimental designs and their use with computer experiments. The second part gives an introduction to Latin hypercube sampling (LHS). In section 4.2 it is shown that LHS asymptotically fills the entire experimental space. Section 4.3 gives a brief introduction to discrepency functions and describes how the variability of random designs, such as LHS, can be used as a measure or discrepency. This section also compares results for variance and other discrepency functions for LHS and simple random sampling (SRS).

4.1.1 Experimental Design and Computer Experiments

Most research in experimental design has been based on the assumption that the response can be effectively modeled by a low order polynomial and that the models are estimated using least squares. Steinberg and Hunter[1] give the most recent overview of experimental design. Concentrating for the moment just on the properties of experimental designs there are several difficulties in using standard experimental design methods that have been developed for physical experiments.

Factorial designs, and more generally, orthogonal arrays[2 3] are the classic experimental designs for low order polynomials but do not adequately address

the problems of modeling nonlinear functions and large numbers of input factors. For factorial designs, sample size increases rapidly as the number of input factors increases. The design could be highly fractionated, but this leads to estimation and confounding problems when using linear models. The number of levels at which each variable is tested also needs to remain small to keep the sample size small. This can make estimation of nonlinear functions difficult. Finding fractional designs is difficult for larger problems and must either be looked up in published tables, which only addresses a finite number of possibilities, or computer algorithms could be used to determine the alias structure.[4 5] These algorithms do not necessarily find the best designs and can be computationally expensive.

Many orthogonal arrays have been developed that are smaller than regular fractional factorials. A notable example is Plackett-Burman type designs,[6] which are designed for estimating main effects. Wang and Wu[7 8] is some of the more recent work to make orthogonal arrays flexible in the selection of factor levels and keeping the number of runs small. These orthogonal arrays tend to be difficult to compute and still have the same problem with the tradeoff between sample size and number of factor levels. Results are scattered throughout the literature so determining what types of designs are available and best for any given situation is not a simple task.

Orthogonality is a useful property for experimental designs, but it is not an essential one, especially if the purpose of the experiment is to develop a prediction model and not estimate parameters. For computer experimentation orthogonal arrays have a major drawback. One of the properties of an orthogonal array is that if the design is projected onto a smaller space, as is the case if input variables are nonsignificant, the corresponding design is still orthogonal, but with more replicates at each design point. For computer experiments this replication is a waste of sampling points since there is no measurement error.

For other classes of experimental design, such as α-optimality and maximin, the choice of sample size is more flexible than for orthogonal arrays. However, these designs have difficulties of their own. Optimal designs usually require that model parameters be known to develop the experimental design. Of course, it is usually just these parameters that the investigator is trying to estimate. These designs also tend to be very expensive to compute when there is a moderate number of input factors.

Initially, experimental designs for computer experiments were developed for numerical integration problems.[9] [10] These attempts did not have much success beyond d=1 and the numerical analysis literature [11] also have had difficulties devising strategies for d>1. Monte Carlo simulation studies, which typically used simple random sampling, quickly became the sampling method of choice for computer experiments because they are quick and easy to implement for high dimension problems and many of the initial studies in computer experiments investigated the distribution of the response given "random" inputs. However, SRS can be improved upon as a design strategy.

4.1.2 Latin Hypercube Sampling - A Review

Latin hypercube sampling, introduced by McKay, Conover and Beckman (MC&B),[12] is an extension of lattice sampling described by Patterson.[13] Initially, LHS was used in sensitivity analysis as an improvement to SRS for estimating cumulative distribution functions.[14] [15] The length of the citation list for MC&B indicates that LHS is in common use today.

Constructing a Latin hypercube sample is a straightforward process. Assume the experimental space has been scaled to $[0,1]^d$, where d is the number of input variables. Let $\mathbf{z} = [0,1, \ldots, n-1]$, where n is the number of runs in the experimental plan. Then

$$s_{.j} = \frac{\pi_j(\mathbf{z})+1/2}{n}, j = 1, \ldots, d$$

is the j^{th} column of the experimental design S, where $\pi_1, \ldots, \pi_d$ are independent uniform random permutations of $\mathbf{z}$. This algorithm places the design points in the center of the randomly selected sections of the grid. MC&B suggest randomizing the location within these selected locations, i.e. let $s^*_{ij} = s_{ij} + \gamma_{ij}$, where γ_{ij} is a uniform random variable on $[-1/2n, 1/2n]$. This helps to simplify some theoretical calculations, but does not improve the structure of the design. This becomes apparent when n is a moderate size, causing the range of γ_{ij} to be trivially small. For the designs in this thesis the design values will not be randomized.

Several variations on MC&B initial LHS algorithm have been developed. Iman and Conover [16] discuss methods for inducing correlations between the input variables. Handcock[17] develops an algorithm called cascading Latin hypercube sampling. This type of Latin hypercube sampling generates clusters of

points at several levels, the number of levels chosen by the designer. At each level a standard LHS is carried out to locate the clusters, then at the lowest level one further LHS is created to locate the design points for the cascading Latin hypercube sampling. This design was developed to improve estimates of scale and smoothing parameters in the Matérn class of covariance functions.

MC&B show that for estimators of the form

$$T(Y_1, \ldots, Y_n) = \frac{1}{n} \sum_{i=1}^{n} g(Y_i),$$

where $Y_i = h(\mathbf{x}_i)$ and $h(\mathbf{x}_i)$ is a square integrable function, the variance is smaller using LHS than it is using random sampling or stratified sampling if certain conditions of monotonicity are met. Stein[18] and Owen[19 20 21] expand on the work by MC&B on variance estimation and asymptotic behavior of the estimates. Stein extends the work on comparing variances by showing that asymptotically $Cov(h(X_1), h(X_2)) \leq 0$ so that $Var(h(\mathbf{X}_{LHS})) \leq Var(h(\mathbf{X}_{SRS}))$ without the monotonicity restrictions given by MC&B. Stein also shows that the more additive the function $Y = h(\mathbf{x})$, the smaller the variance is when using LHS. Stein derives a central limit theorem for $E(h(X))$ and outlines another algorithm as an alternative to the one given in Iman and Conover.[16] Owen extends Stein's work on the central limit theorem and shows that the variance for integrals of $h(\mathbf{X})$ is less using LHS than SRS.

None of these papers address the physical, or space filling properties, of LHS. Two concepts can be used to investigate how well a design fills up the input space.

1. How completely does the design cover the input space?
2. How uniformly are the design points spread throughout the input space?

The answer to the first question tries to get a measure on how well the code has been exercised, i.e. how well the full range of all input variables is covered. The latter question is addressed by using discrepency functions to compare designs to a uniform distribution. For example, a 2^k full factorial design does not cover the input space very completely only having observations in the corners while SRS covers the full input space more completely. Conversely, the 2^k design spreads points more uniformly through the input space than SRS.

4.2 Asymptotic Space Filling Property

The purpose of this section is to show that for any neighborhood around an arbitrarily selected point **x**, the probability of at least one design point being in the neighborhood tends to one as $n \to \infty$. This statement implies that for some n there will be no point in the input space that will not be within some prescribed distance of a design point, with probability 1. After the proof, an application of this property is given.

Proof:

Let a neighborhood of $\mathbf{x} = (x_1, \ldots, x_d)$ be defined as $N_\delta(\mathbf{x}) = [\mathbf{x} \pm \delta]$. Let $\mathbf{s}_i^* = (s_{i1}^*, \ldots, s_{id}^*) = \mathbf{s}_i$, if $s_{i1} \in N_\delta(x_1)$. If $n > 1/\delta$, then there exists a design point $\mathbf{s}_i^*$ in S. Since $n > 1/\delta$ and the values of the design points are equally spaced at intervals of $1/n$, $P(s_{ik}^* \in N_\delta(x_k)) = p_\delta$, where $p_\delta = \max_{j=1,\ldots,n} \{\frac{j}{n} : \frac{j}{n} < \delta\}$. The design points are randomized independently over the input variables so these probabilities are independent for $k = 2, \ldots, d$. Then

$$P(s_{ik} \notin N_\delta(\mathbf{x}), k=2,\ldots,d) = 1 - \prod_{k=2}^{d} P[s_{ik}^* \in N_\delta(x_k)] = 1 - p_\delta^{d-1}.$$

It is sufficient that there is at most one point in $N_\delta(\mathbf{x})$ and the probability of this is

$$1 - \prod_{s_{i1} \in N_\delta(x_1)} (1 - p_\delta^{d-1}) = 1 - (1 - p_\delta^{d-1})^{np_\delta} \to 1, \; n \to \infty \tag{4.2.1}$$

since p_δ is a constant.

For SRS the first dimension has the same probability distribution as any other dimension so

$$P(\mathbf{s} \in N_\delta(\mathbf{x})) = 1 - (1 - \delta^d)^n,$$

so SRS has this property as well.

4.2.1 Application

Computer simulation models typically do not have any measurement error, which implies that theoretically the $MSE(Y(\mathbf{x}))$ can be zero. This will happen if $h(\mathbf{x})$ is a simple linear model and regression is used to estimate the model. However, if the model is incorrect then bias will prevent $MSE(Y(\mathbf{x})) \to 0$ for regression no matter how large the sample size.

Assume that $y(\mathbf{x})$ is a realization of a stochastic process which has the form

$$Y(\mathbf{x}) = \beta + Z(\mathbf{x}).$$

Where $Z(\mathbf{x})$ is a stochastic process with mean zero and correlation

$$Corr(Z(\mathbf{x}), Z(\mathbf{w})) = R(\mathbf{x}, \mathbf{w})$$

between the responses at two inputs $\mathbf{x}$ and $\mathbf{w}$ and the variance

$$Var(Z(\mathbf{x})) = \sigma^2.$$

Let R_S be the correlation matrix for the experimental plan, S and $r(\mathbf{x}) = R(\mathbf{x}, \mathbf{s}_i)$ be the vector of correlations between $\mathbf{x}$ and each point in the experimental plan. If the best linear unbiased predictor (2.2.3) is used and it is assumed β is known, then $MSE(Y(\mathbf{x})) = \sigma^2(1 - r'(\mathbf{x})R_S^{-1}r(\mathbf{x}))$. Details of this model can be found in Chapter 2. The result given in this section can be used to show that $MSE(Y(\mathbf{x}))\rightarrow 0$ for any $\mathbf{x}$ as the design points get close to $\mathbf{x}$ ($n\rightarrow\infty$) for an interpolating predictor.

Proof:

Let $\mathbf{x} = \mathbf{s}_i^* + \delta$, where

$$\mathbf{s}_i^* = \min_{i=1,\ldots,n} \|\mathbf{x} - \mathbf{s}_i\|, \text{ then } \mathbf{s}_i^* \in N_\delta(\mathbf{x}).$$

Then $MSE(Y(\mathbf{x}))$ can be rewritten as

$$\sigma^2(1 - r'(\mathbf{s}_i^* + \delta)R_S^{-1}r(\mathbf{s}_i^* + \delta)).$$

Let $r_\delta(\mathbf{s}_i) = r(\mathbf{s}_i + \delta)$ then by the Cauchy-Schwarz theorem:

$$|(r'_\delta(\mathbf{s}_i^*)R_S^{-1}r_\delta(\mathbf{s}_i^*))(r'(\mathbf{s}_i^*)R_S^{-1}r(\mathbf{s}_i^*))|^{\frac{1}{2}} \geq |r'_\delta(\mathbf{s}_i^*)R_S^{-1}r(\mathbf{s}_i^*)|.$$

Since $MSE(Y(\mathbf{x}))\geq 0$ and $MSE(Y(\mathbf{s}_i^*))=0$, this implies that $r'(\mathbf{s}_i^*)R_S^{-1}r(\mathbf{s}_i^*) = 1$ and

$$1 \geq |r'_\delta(\mathbf{s}_i^*)R_S^{-1}r_\delta(\mathbf{s}_i^*)|^{\frac{1}{2}} \geq |r'_\delta(\mathbf{s}_i^*)R_S^{-1}r(\mathbf{s}_i^*)|.$$

Now because $r(\mathbf{x})$ is a continuous function, as $\delta\rightarrow 0$

$$r'_\delta(\mathbf{s}_i^*)R_S^{-1}r(\mathbf{s}_i^*)\rightarrow r'(\mathbf{s}_i^*)R_S^{-1}r(\mathbf{s}_i^*) = 1$$

and $MSE(Y(\mathbf{x}))\rightarrow 0$. Combined with the argument in Section 4.2 it is clear that $MSE(Y(\mathbf{x}))$ can be made arbitrarily small (in probability) by taking n sufficiently large: i.e. under LHS there exists n_0 such that for $n > n_0$ and η,ε

arbitrarily small, $P(MSE(Y(\mathbf{x})<\eta)>1-\varepsilon$. This is true even if $R(\mathbf{x},\mathbf{w})$ is misspecified since the predictor is an interpolator and $r(\mathbf{s})=0$ even if $R(\mathbf{x},\mathbf{w})$ is incorrect.

4.3 Discrepency Functions

Discrepency functions are used to measure how well points are uniformly spread throughout the design space. Let $\mu(A)$ be the volume of the region A. Then $\mathbf{x}=\{x_1,x_2,\ldots,x_N\}$ is a *uniform sequence* if for any A

$$\frac{\#\{x_i\in A\}}{N}\rightarrow\mu(A),\ N\rightarrow\infty.$$

Discrepency functions measure the deviation of finite sequences from $\mu(A)$. Two common discrepency functions are ordinary discrepency, or Kolmogorov deviation, and L_2 discrepency. Ordinary discrepency is defined as

$$D_N=\sup_A\left|\frac{\#\{x_i\in A\}}{N}-\mu(A)\right|$$

for $A=\{[0,b_1]\times\cdots\times[0,b_d]\}$. L_2 discrepency is defined as

$$D_N^{L_2}=\int_0^y\left[\frac{\#\{x_i\in A\}}{N}-\mu(A)\right]^2dy,$$

where $A=\{[0,y]\text{x}\cdots\text{x}[0,y]\}$. For random designs, such as SRS and LHS, L_2 discrepency can be written as

$$D_N^{L_2}=\int_0^y Var(X)dy,$$

where X is a random variable with a binomial distribution.

4.3.1 Variance as Discrepency Function

Most experimental plans, such as orthogonal arrays, D-optimal and Maxi-min designs can be considered deterministic, i.e. the design points are determined numerically and there is no random component involved. When a LHS or a SRS is generated the resulting experimental plan is a random event or random design. Deterministic designs typically are difficult to derive for large numbers of input variables or may be inflexible in the number of experimental runs in the design, such as for orthogonal arrays. Random designs are more readily computed than deterministic designs since they rely on the process of randomization to spread out the design points while deterministic designs rely on various, often complex,

mathematical structures to derive the design. This is one of the reasons why SRS has been so popular in large experimental settings.

An experimental plan has good space filling properties if the experimental region is covered fully and uniformly. The results in Section 4.2 show that the experimental region is completely covered asymptotically for LHS. One way to determine how uniformly a random design fills the experimental region is to measure the variability in the number of design points that fall in an arbitrary region of the experimental region. If the variability is small then the number of points in a randomly placed region will be close to the mean for that sized region. Since the region is randomly placed the mean number of design points in different regions must be approximately the same. This implies that the design points are dispersed over the experimental region in a more uniform manner than a experimental plan with higher variance.

Stein[18] has shown this to be true asymptotically by his proof that the variance of any square integrable function from a Latin hypercube sample is less than from a simple random sample. Let $h(\mathbf{x})$ be an indicator function where

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is in } V^d \\ 0 & \text{otherwise} \end{cases}, \tag{4.3.1}$$

where V^d is a region in $[0,1]^d$. The following result shows that this can be extended to all n for the function $h(\mathbf{x})$ given in (4.3.1).

One of benefits of using variance as a measure of discrepency for random designs is that no simulation studies are necessary for comparing different random designs and average squared discrepency, which for random designs is equivalent to $\int Var_p(X)dp$ can be used for comparison to deterministic designs. The mean and variance are now computed for SRS and LHS.

In general the problem can be set out as follows. Let the full experimental region be scaled to be on the unit cube $[0,1]^d$. A rectangular subregion, $V^d = V_1 \times, \ldots, \times V_d$, of the experimental space can be described as follows. Select $0 \le p_i \le 1$ and $0 \le v_i \le 1-p_i$ for $i = 1,...,d$. Then $[v_1, v_1+p_1] \times, \ldots, \times [v_d, v_d+p_d] = V^d$ is a subregion of the full experimental space, where p_i is the length of the i^{th} side. Let $p_1 = \cdots = p_d = p$. Let S be a design for d input variables with n runs and s_{ij} be the i^{th} observation for the j^{th} input variable.

Let $X = |V^d \cap S|$, be the number of design points in V^d. Now let

$$Q_{ij} = \begin{cases} 1 & \text{if } s_{ij} \text{ is in } V_j \\ 0 & \text{otherwise} \end{cases},$$

then

$$X = \sum_{i=1}^{n} \prod_{j=1}^{d} Q_{ij}.$$

Let X_{LHS} and X_{SRS} be the number of points in V^d for LHS and SRS respectively.

Note that for fixed i, Q_{ij} is independent of Q_{ik} for all j and k for both LHS and SRS since randomization on each input variable is taken separately in both sampling methods. However, Q_{ik} is not independent of Q_{jk}, $i \neq j$, for LHS but is for SRS. Since the probability of a point from a SRS being in V^d is p^d, the relative volume of V^d to $[0,1]^d$, $X_{SRS} \sim Bin(n, p^d)$ and $E(X_{SRS}) = np^d$ and $Var(X_{SRS}) = np^d(1-p^d)$.

Let $m = [np] + 1$, where $[np]$ is the integer part of np. Fix j and let $Q_j = \sum_{i=1}^{n} Q_{ij}$. Then Q_j is the number of design points for the j^{th} variable that are in V_j and $Q_j = m$ or $Q_j = m-1$ depending on the position of V^d in $[0,1]^d$. Then

$$P(Q_{ij} = 1 \mid Q_j = m) = m/n$$

and

$$P(Q_{ij} = 1 \mid Q_j = m-1) = (m-1)/n.$$

Define $P(Q_j = m) = p^*$ and assume that $p^* = np - (m-1)$. Then

$$E(X_{LHS}) = \sum_{i=1}^{n} \prod_{j=1}^{d} E_{Q_j}(E(Q_{ij} \mid Q_j)).$$

Since the columns of S and the marginal distribution of the rows of S are identically distributed

$$E(X_{LHS}) = \sum_{i=1}^{n} \prod_{j=1}^{d} (p^* \frac{m}{n} + (1-p^*) \frac{m-1}{n}) = n (\frac{p^*}{n} + \frac{m-1}{n})^d = np^d.$$

Assume Q_j is binomially distributed with probability $p^* = np - (m-1)$, then $E(X_{LHS}) = E(X_{SRS})$

To compute $Var(X_{LHS})$ it remains to compute $E(X_{LHS}^2)$.

$$E(X_{LHS}^2) = \sum_{i=1}^{n}\prod_{j=1}^{d} E_{Q_j} E(Q_{ij}^2 | Q_j) + 2\sum_{i<k}\prod_{j=1}^{d} E_{Q_j} E(Q_{ij}Q_{kj} | Q_j)$$

$$= E(X) + n(n-1)(p^* \frac{m(m-1)}{n(n-1)} + (1-p^*)\frac{(m-1)(m-2)}{n(n-1)})^d .$$

Then

$$Var(X_{LHS}) = np^d + n(n-1)(\frac{(m-1)}{n(n-1)})^d (2np - m)^d - (np^d)^2.$$

To prove that $Var(X_{LHS})$ is uniformly less than $Var(X_{SRS})$ it needs to be shown that $Var(X_{SRS}) \geq Var(X_{LHS})$ for all p which follows if

$$np^d(1-p^d) \geq np^d + n(n-1)(\frac{(m-1)}{n(n-1)})^d (2np - m)^d - (np^d)^2.$$

This can be reduced to

$$p^{2d} \geq (\frac{m-1}{n})^d (\frac{2np - m}{n-1})^d \tag{4.3.2}$$

and can be rewritten as

$$f(p) = n(n-1)p^2 - 2np(m-1) + m^2 - m \geq 0.$$

The first and second derivatives of $f(p)$ show that the extremum is a minimum and is found at $p = (m-1)/(n-1)$; the value of $f(p)$ at the minimum is $n-m$ which is greater than zero and the inequality is shown to be correct.

This result shows that $Var(X_{LHS}) \leq Var(X_{SRS})$ over all randomly placed cubic regions. The result is readily extended to any rectangular region by replacing p by p_j, p^* by p_j^* and m by m_j in the equations above. The equations remain fundamentally the same and equation (4.3.2) can be rewritten as

$$\prod_{j=1}^{d} p_j^2 \geq \prod_{j=1}^{d} (\frac{m_j - 1}{n}) \prod_{j=1}^{d} (\frac{2np_j - m_j}{n-1}).$$

Since the result holds for any p such that $(m-1)/n \leq p \leq m/n$ when $d = 1$, then it also holds for the product when the p_j's are different.

The result also holds for any given (nonrandomly placed) rectangular region by using the design S^t where $s_{ij}^t = s_{ij} + \gamma_j$ and γ_j is a random variable with a uniform distribution on $[-1/2n, 1/2n]$.

4.3.2 Behavior of Latin Hypercube Sampling

For the results in this section assume that V^d is cubic with sides of length p. Variance is a function of n, p, and d and p^d is the percent volume of $[0,1]^d$ which the randomly placed cubic region covers. For SRS, variance is only a function of n and $v = p^d$. This is not the case for LHS, but from a few empirical studies and examination of (4.1.1) it is apparent that changing d has only a minimal effect on the results if n and v are fixed. Figures 4.1 and 4.2 show the typical shape for the standard deviations of X for LHS and SRS in relation to n and v respectively. Figure 4.2 shows that $Var(X_{LHS})$ is nearly the same as $Var(X_{SRS})$ for small regions V^d where the chances of any points falling in V^d are small; then well before $v = 0.5$ the two variances diverge only to converge again at $v = 1.0$.

The relative efficiency shown in Figures 4.3 and 4.4 is the ratio of standard deviation of X_{SRS} to X_{LHS}. Figure 4.3 shows that as $n \to \infty$ the relative efficiency has a maximum limit which it approaches fairly quickly. This behavior is true for all values of v. Figure 4.4 is a plot of this limit of relative efficiency versus v. The figure shows that the amount of clustering of design points in LHS becomes relatively less than that for SRS as the scale of observation increases. The relative efficiency quickly drops to one for $0.98 < v \leq 1.0$.

The scatter of points in Figure 4.3 is due to oscillatory behavior of $Var(X_{LHS})$ as a function of n. The reason for this can be seen from the expansion

$$Var(X_{LHS}) = E_U \; Var(X_{LHS} \mid U) + Var_U(E(X_{LHS} \mid U)).$$

Since U is binomial with probability p^*, $Var_U(E(X_{LHS} \mid U))$ is a quadratic function for $(m-1)/n \leq p \leq m/n$ and

$$Var(X_{LHS}) = E_U \; Var(X_{LHS} \mid U) \text{ for } p^* = 0 \text{ or } 1.$$

The oscillations dampen to the limit as $n \to \infty$ since $np \approx [np]$ for all n.

The mean and variance of X_{SRS} is the same whether X_{SRS} is a combination of two samples of size n_1 and n_2 or one sample of size $n_1 + n_2$. This can readily be seen from the equations for the mean and variance. This is true for the mean of X_{LHS} but is not the case for the variance. However, Table 4.1 shows that the variance for two combined samples is only slightly larger than a single sample. The efficiency will vary somewhat given different n and v

Std. Dev. vs. Sample Size

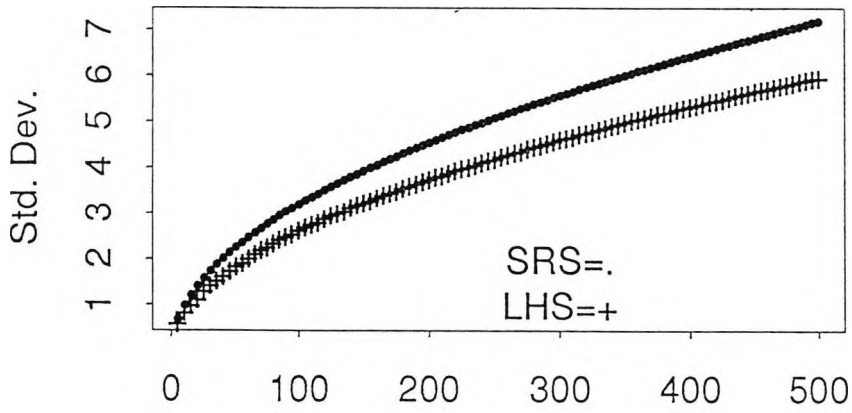


Fig. 4.1

Std. Dev. vs. % Volume

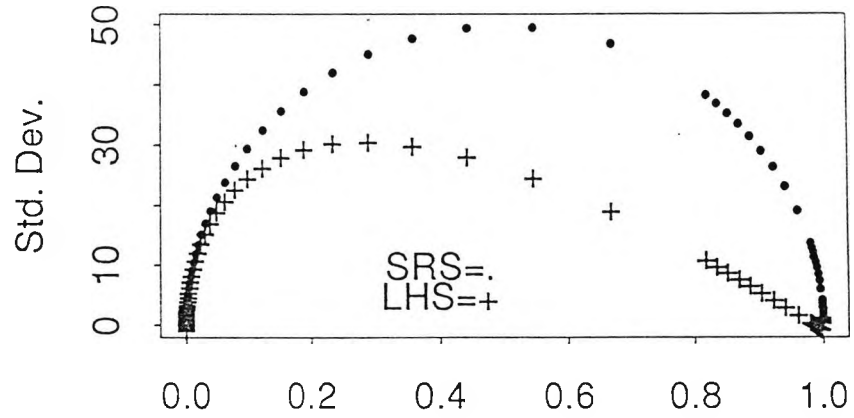


Fig. 4.2

Efficiency vs. Sample Size

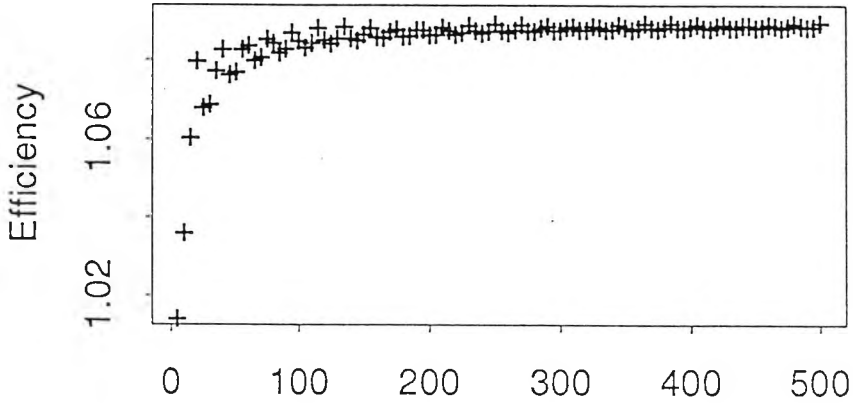


Fig. 4.3

Efficiency vs. % Volume

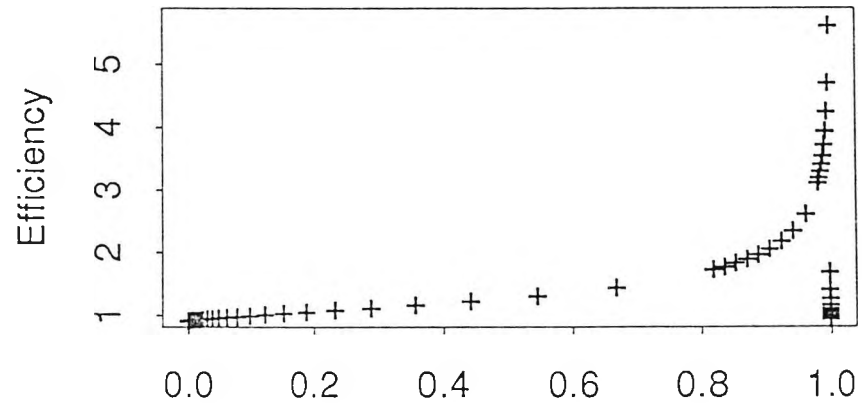


Fig. 4.4

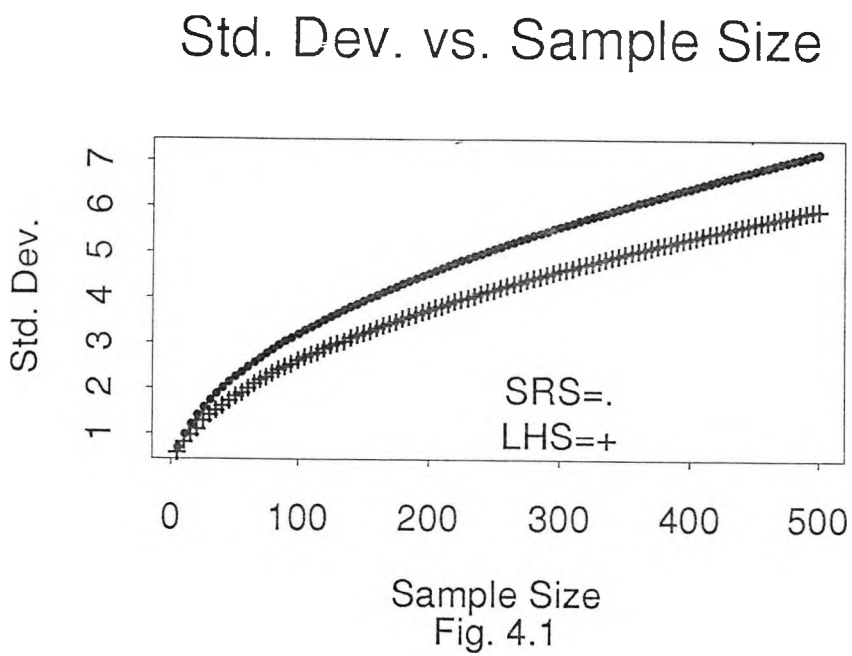


Fig. 4.1

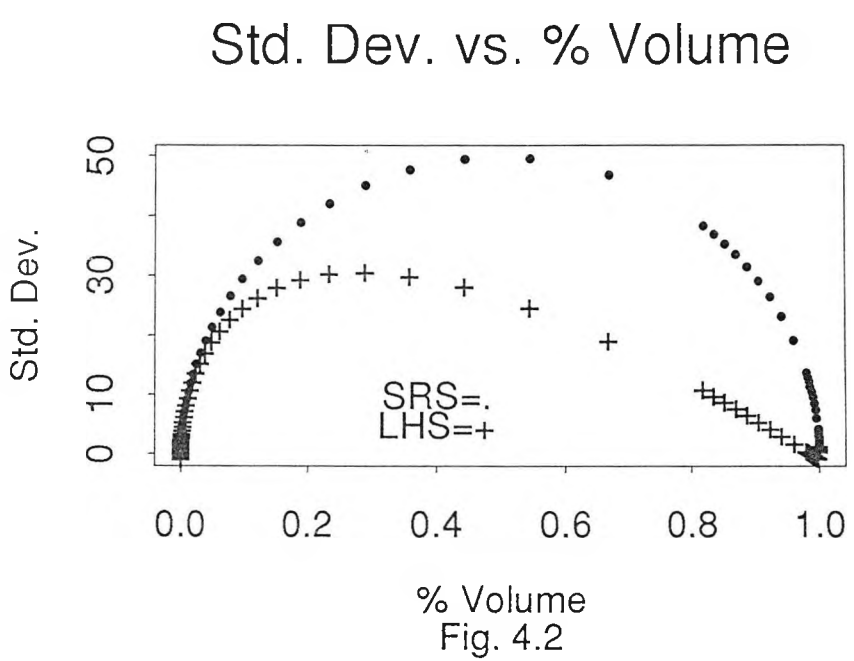


Fig. 4.2

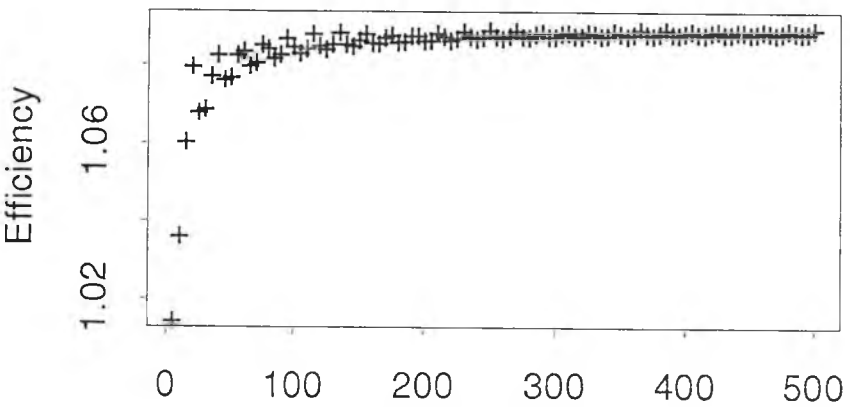


Fig. 4.3

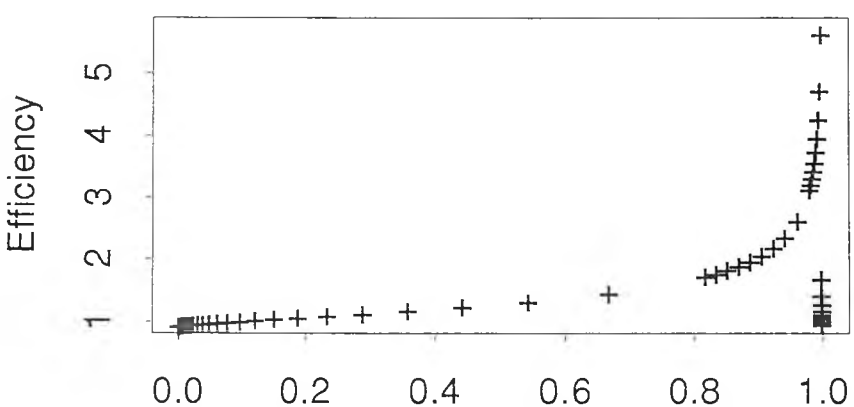


Fig. 4.4

because of the oscillatory behavior of $Var(X_{LHS})$.

	% Volume with d=10		
n	0.1	0.4	0.7
29	1.89	3.04	1.03
58	3.75	5.27	2.04
116	7.46	10.44	4.04
232	14.89	20.64	8.01

Table 4.1: Variance of Combined LHS

4.4 Conclusion

Latin hypercube sampling was developed to improve the variance of simulation study estimates. LHS has been shown to have better variances for these estimates than simple random sampling. LHS has also been used as the experimental design for model estimation for linear models and Gaussian stochastic processes. The benefits of using LHS for model estimation is that it tests each variable at many different input values. The variance of these model estimates have also been shown to be better than for SRS, especially if the models are nearly linear.

The work in this chapter uses the variance of the number of design points in an arbitrary region to measure the uniformity of a design and shows that LHS cover the experimental space more completely and more uniformly than SRS. It also shows that several small LHS, with a total number of n runs, have nearly the same space filling capacity as one large LHS with n runs.

4.5 References

1. Steinberg, D. M. and Hunter, W. G., (1984) "Experimental Design: Review and Comment (with discussion)," *Technometrics*, 26, 71-130.
2. Box, G. E. P. and Draper, N. R., (1987) *Empirical Model Building and Response Surfaces* , John Wiley & Sons , New York, N.Y. .
3. Box, G. E. P., Hunter, W. G., and Hunter, J. S., (1978) *Statistics for Experimenters* , John Wiley , New York .

4. Franklin, M. F., (1985) "Selecting Defining Contrasts and Confounded Effects in p^{n-m} Factorial Experiments," *Technometrics*, 27, 165-172.

5. Wynn, H. P., Sivaloganathan, S., Buck, R. J., and Lewis, S. M., (1992) "Generate: An Algorithm for the Computer Generation of Orthogonal Arrays with Specific Alias Structure," Engineering Design and Quality Centre Technical Report #30.

6. Plackett, R. L. and Burman, J. P., (1946) "The Design of Optimum Multifactorial Experiments," *Biometrika*, 33, 305-325.

7. Wang, J. C. and Wu, C. F. J., (1991) "An Approach to the Construction of Asymmetrical Orthogonal Arrays," *Journal of the American Statistical Association*, 86, 450-456.

8. Wang, J. C. and Wu, C. F. J., (1992) "Nearly Orthogonal Arrays with Mixed Levels and Small Runs," *Technometrics*, 34, 409-422.

9. Sacks, J. and Ylvisaker, D., (1970) "Statistical Designs and Integral Approximation," in *Proc. 12th Bien. Sem. Canad. Math. Congress*, ed. R. Pyke, pp. 115-136, Canadian Mathematical Congress, Montreal.

10. Ylvisaker, D., (1975) "Designs on Random Fields," in *A Survey of Statistical Design and Linear Models*, ed. J. N. Srivastava, pp. 593-607, North-Holland, Amsterdam.

11. Davis, P. J. and Rabinowitz, P., (1984) *Methods of Numerical Integration,* 2nd ed., Academic, Orlando, Fla..

12. McKay, M. D., Conover, W. J., and Beckman, R. J., (1979) "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239-245.

13. Patterson, H. D., (1954) "The Errors of Lattice Sampling," *Journal of the Royal Statistical Society - Series B*, 16, 140-149.

14. Iman, R. L. and Conover, W. J., (1980) "Small Sample Sensitivity Analysis Techniques for Computer Models, With Application to Risk Assessment (with discussion)," *Communications in Statistics - Theory and Methods*, A9, 1749-1874.

15. Downing, D. J., Gardner, R. H., and Hoffman, F. O., (1985) "An Examination of Response Surface Methodologies for Uncertainty Analysis in Assessment Models," *Technometrics*, 27, 151-163.

16. Iman, R. L. and Conover, W. J., (1982) ''A Distribution-Free Approach to Inducing Rank Correlation Among Input Variables,'' *Communications in Statistics- Simulation and Computation*, 11, 311-334.

17. Handcock, M. S., (1991) ''On Cascading Latin Hypercube Designs and Additive Models for Experiments,'' *Communications in Statistics-Theory and Methods*, 20, 417-440.

18. Stein, M., (1987) ''Large Sample Properties of Simulations Using Latin Hypercube Sampling,'' *Technometrics*, 29, 143-151.

19. Owen, A. B., (1992) ''A Central Limit Theorem for Latin Hypercube Sampling,'' *Journal of the Royal Statistical Society - Series B*, 54, 541-552.

20. Owen, A. B., (1991) ''Orthogonal Arrays for Computer Experiments, Integration and Visualization,'' *Technical Report, University of Stanford.*

21. Owen, A. B., (1992) ''Lattice Sampling Revisited: Monte Carlo Variance of Means Over Randomized Orthogonal Arrays,'' Stanford University Technical Report.

- Chapter 5 -

Numerical Optimization Algorithms

5.1 Introduction

To avoid having the topic of optimization repeating itself at various points in other chapters a description and discussion of the numerical optimization algorithms used in this thesis has been left to this chapter. The robust engineering design method described in Chapter 3 uses numerical optimization algorithms at two separate times: for maximization of the likelihood equation and when searching for a robust engineering design with the statistical model as a cheap emulator of the simulation model. The MLE optimization problem is a univariate optimization problem but has a large number of inputs. The search for optimal engineering designs in the described examples were multivariate optimization problems with a moderate to large number of inputs.

Although these optimization problems play an integral part in developing the predictor and finding optimal engineering designs, we did not have the inclination or expertise to develop new optimization algorithms. Our goal was to find an optimization algorithm that was already available and achieved good results. During the investigation of the examples in Chapter 6, several different optimization algorithms were used. A brief outline and discussion of the algorithms are given in Section 5.2-5.4. The rest of this section gives a brief overview of loss functions, optimization algorithms and their relationship.

Numerical optimization algorithms search for the minimum (maximum) of the objective function given possible constraints on (functions of) the inputs. The objective function for model parameter estimation is the likelihood function as stated in (3.1.1). This objective function has a univariate response. The objective function for optimization of the engineering design is the loss function. As mentioned in Section 1.3, the loss function may be a multivariate response function.

Two ways to handle problems with a multivariate response objective function are: to create a univariate response from the multiple responses, or to treat one of the responses, or a function of several responses, as the target to be minimized and the rest held within stated boundaries or constraints. Frequently,

the organization of the response vector into targets and constraints follows readily from the specification description. Several possibilities for the loss function were considered from these two approaches, especially for the voltage-shifter circuit example in Chapter 6. However, a thorough study for the best loss function was not carried out because it was beyond the scope of the main focus of our research.

The optimization algorithm that can be used for finding the optimal solution depends on the objective function. For example, the Nelder-Mead simplex method does not allow constraints on the inputs, while the adaptive random search (ARS) algorithm and the non-linear programming method, NPSOL, do allow constraints. The Nelder-Mead simplex algorithm has been used for computing maximum likelihood estimates throughout the research. For multivariate objective functions a univariate response needs to be created from the multivariate response if the Nelder-Mead simplex is to be used. If constraints are added then ARS or NPSOL can be used.

There is some common notation for all algorithms. Let $\mathbf{x}$ be the d-dimensional vector of inputs over which the function $y(\mathbf{x})$ is to be optimized, where $y(\mathbf{x})$ may be a multivariate response.

5.2 Nelder-Mead Simplex

The Nelder-Mead simplex is an optimization algorithm which requires a univariate response objective function and no constraints on the inputs. This optimization algorithm is used to compute the maximum likelihood estimates for the model described in Chapter 2. Since the correlation parameters are constrained, $\theta \geq 0$ and $1 \leq p \leq 2$, and the Nelder-Mead simplex algorithm requires no constraints, the correlation parameters are translated to $\theta' = \ln \theta$ and $p' = \sin^{-1}(2*(p-1.5))$ during the search part of the optimization.

The basic idea of the Nelder-Mead simplex algorithm is to take $d+1$ points, $\mathbf{x}_1, \ldots, \mathbf{x}_{d+1}$, in the input space and replace the highest point, $\mathbf{x}_H$ with $\mathbf{x}_H = \alpha \mathbf{x}_H$, where $\alpha = -1, 0 < \alpha < 1$, or $\alpha > 1$ or a suitable combination of the three. The different α's are referred to as a reflection, contraction or expansion. A stepwise outline of the algorithm as loosely described in Numerical Recipes is:

1. Starting point $\mathbf{x}_0$ and

$$\mathbf{x}_i = \mathbf{x}_0 + \lambda \mathbf{e}_i, i = 1, \ldots, d.$$

2. Compute reflection. If there is an improvement then try expansion, if not try contraction.

3. If no movement in Step 2. try multiple contraction around the value of **x** which gives a minimum.

4. Repeat Steps 2 and 3 until stopping rule.

Two stopping rules: either the change in **x** from one step to the next is less than *tol* or the value of $y(\mathbf{x})$ from one step to the next is less than *ftol*, where *tol* and *ftol* are user defined tolerances.

This algorithm will always find a local minimum, but does not necessarily find the global minimum, a common problem with optimization algorithms. To improve the chances that the minimum found is a global minimum it is suggested that the algorithm be run several times, each time from a different starting point. From personal experience starting from 3 to 5 different points is usually sufficient.

Two different versions of this algorithm were used at various times for computing maximum likelihood estimates, the subroutine E04CCF from the NAG library and AMOEBA from Numerical Recipes.[1] From informal comparisons of the two algorithms we decided that the AMOEBA version of the algorithm was more efficient and regularly found better solutions than the NAG version.

5.3 Adaptive Random Search

The first algorithm used for the engineering design optimization problem with constraints was adaptive random search[2 3 4] as described by Pronzato, *et al.*[5] As the name implies this is a global random optimization algorithm. Most random search algorithms are an iteration of two steps: generate new observations from some distribution and select the best setting from the new observations according to a given rule and repeat with new observations.

The adaptive random search algorithm chooses its new observations from distributions with different variances to create ever decreasing search areas. The best of the search areas is then investigated more closely before the procedure repeats itself. Let $\mathbf{z} = \mathbf{x} + \mathbf{r}$ where $\mathbf{r}$ is a random vector, normally distributed with mean zero and variance

$$\Sigma(\sigma) = \text{diag}\,[\sigma_1, \sigma_2, \ldots, \sigma_d].$$

Let $\sigma^{(1)} = \mathbf{x}_{max} - \mathbf{x}_{min}$, where $\mathbf{x}_{max}$ and $\mathbf{x}_{min}$ are the upper and lower bounds of

the input space and $\sigma^{(i)}=0.1^{i-1}\sigma^{(1)}$, $i=2,\ldots,f_1$. An outline for the algorithm is:

1. Select $\mathbf{x}_{max}$ and $\mathbf{x}_{min}$ and set the starting point $\mathbf{x}_0=0.0$. Let $k=0$.
2. For $j=1,\ldots,f_1$:

   Generate $f_2(j)$ observations of $\mathbf{y}(\mathbf{x}_k+\mathbf{r})$ where $\mathbf{r}$ is normally distributed with variance $\Sigma(\sigma^{(j)})$. If $\mathbf{z}$ is outside the input space then $\mathbf{x}$ is ignored. Let $\mathbf{x}_k$ equal the best set of inputs, $\mathbf{z}$, from the new observations. Let $k=k+1$.
3. For the most successful $\sigma^{(j)}$, generate f_4 new observations of $\mathbf{y}(\mathbf{x}_k+\mathbf{r})$ and let $\mathbf{x}_k$ equal the best set of inputs, $\mathbf{z}$, from the new observations. Let $k=k+1$.
4. Repeat Steps 2 and 3 until a stopping rule has been reached. There is a series of three stopping rules, the order of implementation is: Stop if
   a. a maximum number of iterations is reached.
   b. $\sigma^{(f_1)}$ has been selected f_5 times.
   c. $\mathbf{y}(\mathbf{x})$ has reached some predetermined criterion.

The last rule typically is not implemented because of the difficulty in establishing criteria approximating the optimal solution.

There are a number of parameters in the algorithm that need to be assigned. We used the values suggested in Pronzato, *et al.*: $f_1=f_5=5$, $f_2(i)=f_3/i$, and $f_3=f_4=100$. Since the input variables are scaled to [−0.5,0.5] for our prediction models, $f_1=4$ was also used because $f_1=5$ did not generate a lot of movement relative to the required accuracy in $\mathbf{x}$.

The adaptive random search algorithm can handle any type of objective function and constraint problem, as long as appropriate decision making rules are coded, because the new observations are selected at random. It is only a matter of testing whether the new point fulfills the criterion better than previously selected points. If the new point is better then the previous point it is kept for reference and additional points are tested to try to improve on it.

5.4 NPSOL

A method for handling optimization problems with a smooth nonlinear objective function and both linear and nonlinear constraints is a nonlinear programming algorithm which has several acronyms, but will be referred to here as NPSOL. The implementation of the algorithm used is the NAG version of

NPSOL, subroutine E04UCF. The description that follows is an overview of the NAG manual description. See also Gill, *et al.*[6]

A formal construction of the problems which NPSOL is designed to solve can be stated as

$$\text{(5.4.1)} \qquad \text{Minimize } y(\mathbf{x}) \text{ subject to: } l \le \begin{Bmatrix} \mathbf{x} \\ A\mathbf{x} \\ c(\mathbf{x}) \end{Bmatrix} \le u,$$

where $y(\mathbf{x})$ is the objective function and $\mathbf{x}$, A, and $c(\mathbf{x})$ are the bound, linear and nonlinear constraint functions respectively.

An initial estimate of the solution is given and the routine first finds a solution that satisfies the bound and linear constraints. Once this occurs a series of major and minor iterations are carried out. The major iterations are used to find the optimal solutions, while the minor iterations are used to find solutions to quadratic programming subproblems needed for the major iteration. Derivatives for $y(\mathbf{x})$ and $c(\mathbf{x})$ are requested for the algorithm, but finite differences are used to compute estimates for those derivatives that are not furnished.

The major iterations create a sequence $\{\mathbf{x}_k\}$ that converge to $\mathbf{x}^*$, a first order Kuhn-Tucker point of (5.4.1). The sequence is of the form $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}$, where x_k is the current value, $\alpha > 0$ is the step length and $\mathbf{p}$ is the search direction. Given the search direction the major iteration calculates a steplength α that produces a sufficient decrease in an augmented Lagrangian merit function.

The search direction is determined from the solution of the quadratic programming subproblem and is the minor iteration of NPSOL. The quadratic programming subproblem can be specified as

$$\text{Minimize } g'\mathbf{p} + \frac{1}{2}\mathbf{p}'H\mathbf{p} \quad \text{subject to} \quad l^* \le \begin{Bmatrix} \mathbf{p} \\ A\mathbf{p} \\ A_N\mathbf{p} \end{Bmatrix} \le u^*,$$

where g is the gradient of $y(\mathbf{x})$ at $\mathbf{x}$, H is an approximation to the Hessian of the Lagrangian function, A_N is the Jacobian matrix of $c(\mathbf{x})$ and l^* and u^* are new bounds, which are a function of l and u and the constraints in (5.4.1).

5.5 Conclusion

Most deterministic algorithms only guarantee a local optimum and several starting points are used to help ensure a global optimum. This is not necessary for the ARS algorithm because the algorithm moves around the input space in a random manner. Eliminating multiple starting points reduces the total number of observations that may be used. From casual observations we felt that ARS did not consistently give as good results as AMOEBA or NPSOL.

Numerical optimization methods have not received much attention in the discussion about robust engineering design methods, but there are many benefits to their use. The use of existing optimization algorithms can lead to better solutions more quickly than the sequential improvement philosophy of Taguchi. The search for the best optimization algorithm was not exhaustive by any means and an improvement in the optimization algorithm would immediately improve the efficiency of the search for maximum likelihood estimates and optimal engineering designs.

5.6 References

1. Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., (1986) *Numerical Recipes,* Cambridge University Press, Cambridge.

2. Andronikou, A. M., Bekey, G. A., and Masri, S. F., (1982) "Identification of Nonlinear Hysteretic Systems using Random Search," *Proceedings 6th IFAC Symposium on Identification and System Parameter Estimation*, 1, 263-268.

3. Bekey, G. A. and Masri, S. F., (1983) "Random Search Techniques for Optimization of Nonlinear Systems with Many Parameters," *Math. Comput. Simulation*, 25, 210-213.

4. Masri, S. F., Bekey, G. A., and Safford, F. B., (1976) "An Adaptive Random Search Method for Identification of Large-Scale Nonlinear Systems," *Proceedings 4th IFAC Symposium on Identification and System Parameter Estimation*, 246-255.

5. Pronzato, L., Walter, E., Venot, A., and Lebruchec, J., (1984) "A General-Purpose Global Optimizer: Implementation and Applications," *Mathematics and Computers in Simulation*, 26, 412-422.

6. Gill, P. E., Hammarling, S. J., Murray, W., Saunders, M. A., and Wright, M. H., (1986) "User's Guide to LSSOL (Version 1.0)," Dept. of Operations Research, Stanford University (Report SOL 86-1), Stanford University.

- Chapter 6 -

Case Studies in Robust Engineering Design

6.1 Introduction

A major problem for integrated circuit designers is how to design circuits so that performances, as predicted by a circuit simulator, are insensitive to uncontrollable variations in the manufacturing process and in the operating conditions. Statistical circuit design methods [1] attempt to optimize the performances of a circuit design in the presence of these uncontrollable variations.

Statistical modeling of circuit and process simulators to achieve consistently good performance has become increasingly prominent.[2 3 4] In some examples,[3 4] circuit performances are modeled as quadratic functions of all the input factors of interest: the designable factors, the uncontrollable statistical variations, and the operating conditions. The statistical model is fitted from relatively few runs of the simulator. A key characteristic of these approaches is that the statistical model can be used as a computationally cheap surrogate for maximizing yield, minimizing variations of performances around targets, or optimizing other measures of quality.

These and other existing tools for statistical circuit design work well when the number of circuit factors is small, no more than about ten, and the space of factor values is sufficiently restricted to admit simple modeling. With more factors, the number of circuit simulations required to fit quadratic models can exceed practical limits. Also, if the factors are allowed wide ranges the simple quadratic models may not be effective for approximating the performance functions.[3 4] The methods of Taguchi[5] for finding robust designs likewise appear inadequate for treating many nonlinear situations.[3] In these circumstances optimization is a formidable task. All standard routines have difficulties that inhibit or prohibit their use. Non-differentiability of performance functions stops some. Failure to converge and getting trapped at local optima are common occurrences. Many have trouble incorporating constraints on the performance functions. Most serious is that all require large numbers of function evaluations.

Unless function evaluations are cheap, direct application of an optimization routine to the simulated performances is not feasible. Optimizing via an inexpensive approximation to the simulator is more feasible, but the approximation itself must be accurate for reliable optimization. Unfortunately, these two requirements---few simulator runs and an accurate approximation---conflict with each other.

These difficulties are overcome by multi-stage experimentation and, during each stage, by building a statistical model (predictor) of the simulator. The advantages of sequential experimentation for optimization are well known, e.g. Box and Draper,[6] in which the term "evolutionary operation" is used. Such methods have typically relied on the adequacy of simple models over small factor ranges to indicate a local direction of improvement. Instead, a predictor is built on the region of interest and obtain sub-ranges of the factors where the optimum is predicted to lie. On the smaller region we build a more accurate model and continue the search. To cope with many factors, their large ranges, and, hence, complex performances, a class of approximating functions (predictors) that is highly data-adaptive is used. These predictors often have much less error than polynomial models.[7] [8] This blend of sequential experimentation and modeling allows the optimum to be found in a few stages (usually 2 or 3) and with comparatively few simulation runs.

In outline the approach is:

Step 1. Postulate a statistical model for each performance.

Step 2. Plan an experiment and run the simulator to collect the data.

Step 3. Use the data to fit the models.

Step 4. Check the accuracy of prediction and plot the factor effects.

Step 5. If the models are insufficiently accurate, choose a subregion for the next experiment and return to Step 1.

Step 6. When the models are sufficiently accurate, optimize the objective (loss, yield, etc.,) using the fitted model in place of the performance functions.

Section 6.2 formulates the problem and elaborates on these steps. In Section 6.3 and 6.4 these techniques are applied to multiple performances and criteria, and incorporate manufacturing variations. The example used in Section 6.3 is that of a GaAs voltage shifter circuit. In Section 6.4 the methods are applied

to a digital logic circuit with 20 performances of interest, but with no manufacturing variations. In Section 6.5 the methods and the results of their applications are discussed.

6.2 Modeling and Optimization

Circuit performances generally depend on design factors, operating conditions (environmental noise factors), and on statistical variations of device factors (uncontrollable manufacturing noise factors). Some designable factors may have uncontrollable variations superimposed. We focus initially on a single performance, denoted by y. Let $\mathbf{X}=(X_1,...,X_d)$ denote the d-dimensional vector of varying input factors to the circuit simulator, all the other inputs remaining fixed. We typically normalize each X_i to lie in [-0.5, 0.5]. We write $X_i=c_i+U_i$ to differentiate between the controllable and uncontrollable components of X_i. If an input factor has no designable adjustment, then c_i has a fixed value (make it 0) and is ignored. Similarly, if there is no uncontrollable variation, then $U_i=0$. The performance y is, therefore, a function of $\mathbf{X}=\mathbf{c}+\mathbf{U}$, where $\mathbf{c}=(c_1,...,c_d)$ and $\mathbf{U}=(U_1,...,U_d)$. Finally, let $\mathbf{x}$ be a realization of $\mathbf{X}$.

We adopt the Taguchi[5] objective of minimizing a "loss", for example a measure of variability around a target performance. For recent accounts of Taguchi methods see Dehnad[9] , Phadke[10] or Chapters 1.2-1.4, 7. The particular loss used invariably depends on the problem. For the example of Section 6.3, the target value of the output voltage is 5 V and it is the fluctuation around 5 V due to $\mathbf{U}$ that we want to minimize. This suggests the loss structure

$$L_{MAX}(\mathbf{c}) = \int |y(\mathbf{c}+\mathbf{U})-5|^2 \, d\Gamma(\mathbf{U}), \tag{6.2.1}$$

where $\Gamma(\mathbf{U})$ is the noise factor distribution and the ultimate objective to minimize this loss by choice of $\mathbf{c}$. A more complicated loss involving many more performance functions and criteria is exemplified in Section 6.4.

We now detail the six step scheme outlined in Section 1.

Step 1. Postulate a tentative approximating model for the performance.

Low order polynomial models are well known, but, unless the performance function is simple, which is likely to occur only when the input ranges are small, these models can be misleading. For this and other reasons [7] [8] we have adopted a different class of models. A brief overview is given here, but see Chapter 2 for a thorough discussion. If $\mathbf{x}$ is the vector of input factors, let

(6.2.2) $$y(\mathbf{x}) = \beta + Z(\mathbf{x}).$$

Here, $Z(\mathbf{x})$ is a stochastic process having a correlation structure

$$Corr\ (Z(\mathbf{x}),Z(\mathbf{w})) = R(\mathbf{x},\mathbf{w})$$

between the responses at two vectors of inputs $\mathbf{x}$ and $\mathbf{w}$ and having the constant variance

$$Var\ Z(\mathbf{x}) = \sigma^2.$$

We can also include, for example, linear terms:

(6.2.3) $$y(\mathbf{x}) = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + Z(\mathbf{x}).$$

The correlation structure of $Z(\mathbf{x})$ captures the systematic departures from a simple model, such as the constant in (6.2.2). Typically, $Z(\mathbf{x})$ and $Z(\mathbf{w})$ are highly correlated for $\mathbf{x}$ near $\mathbf{w}$. As y is deterministic and is often smooth, Z (through R) should have corresponding properties. The specific R we use is of the form

(6.2.4) $$R(\mathbf{x},\mathbf{w}) = \prod_i \exp(-\theta_i |x_i - w_i|^{p_i}).$$

The correlation constants θ and $\mathbf{p}$ are unknown, as are σ^2 and β or $\beta_1, \ldots, \beta_d$. The θ's are non-negative and the p's are between 1 and 2. These constants are estimated in Step 3.

Whether to include the linear terms in model (6.2.3) can be dealt with in each individual problem. In most cases we have found little advantage to using (6.2.3) so we typically start with (6.2.2) to allow more data to be used for estimating the correlation constants of R (higher order models such as (6.2.3) "use up" degrees of freedom).

Step 2. Plan an experiment and run the simulator to collect the data.

We use a Latin hypercube experimental plan to select the inputs. These plans have some attractive properties for computer experiments. They are simple to generate and cover the experimental region fairly uniformly. See Chapter 4 for a further discussion of Latin hypercube sampling.

The question of sample size is a difficult one. Earlier empirical evidence indicated that the estimation procedure used in Step 3 needs about 3 observations per constant. Since we may want to discard outlying data, some extra

observations should be added at initial stages of experimentation. Discarding data is suspect if the goal is to model performances over the entire region. Our purpose, however, is optimization, so deleting outlying data can be helpful by allowing more accurate fitting on more relevant parts of the region. Criteria for selecting sample sizes are the subject of ongoing research.

Step 3. Use the data to fit the model.

We estimate the correlation constants in R via maximum likelihood and obtain $\hat{y}$. The mathematical details and computational methods are described in Chapters 2 and 3.

Step 4. Check the accuracy of prediction and plot the factor effects.

To measure the accuracy of the predictor $\hat{y}$ from the fitted model, we compute a root mean squared error of prediction using (2.2.4) at a number of randomly selected points in the region. In the example of Section 6.3 we choose 20 points. The range of these errors is a good indicator of the accuracy of the predictor. In the example of Section 6.4 extra runs were not available, so cross-validation estimates of error (3.4.1) were used to estimate the accuracy of the predictor.

To visualize the fitted models, we decompose $\hat{y}$ into a mean value, main effects due to individual factors, and second-order interactions between them.[8] This is described in Section 2.4 as well. The estimated main effect due to x_i is the average of $\hat{y}$ over all factors except x_i minus the mean value. These effects are then plotted. The main effect can provide an excellent, yet simple, indication of how a factor affects the performance. Contour plots of the estimated interaction effects are useful for indicating the joint influence of pairs of factors.

At this point we reach a fork in the procedure. If prediction is sufficiently accurate for the particular problem go to Step 6; otherwise proceed to Step 5.

Step 5. Choose a subregion for the next experiment.

An optimization routine (see Step 6 below) can be used to find the center of the new subregion, while the plots in Step 4 are useful in choosing new limits for the factors. The new region has to be selected to take into account the fact that uncontrollable variations cannot be restricted. Then repeat Steps 1 to 4, with data drawn from the new subregion.

Step 6. Optimize the loss function.

The loss depends on the performance function. We replace y by $\hat{y}$ and seek to optimize the resulting predicted loss. For example, in Section 6.3 we minimize over $\mathbf{c}$, $\hat{L}_{MAX}(\mathbf{c}) = \int |\hat{y}(\mathbf{c}+\mathbf{U})-5|^2 \, d\Gamma(\mathbf{U})$ as an estimate of (6.2.1). Because we commonly meet non-differentiable functions subject to complicated constraints various numerical optimization algorithms have been tried. See Chapter 5 for more details. Inspection of the main effects plots can help choose a starting point. This algorithm is also used at Step 5. After finding an estimate of the optimum we do a confirmatory run. If the confirmatory run is unsatisfactory, we take steps to improve the models; see Section 6.3 for an example. A new stage with further data might be necessary if we can not improve the fit of the models.

When there are multiple performances we model the performances individually following Steps 1, 3 and 4. Only one experimental plan is carried out. Optimization is then performed on a single loss function which combines the multiple criteria.

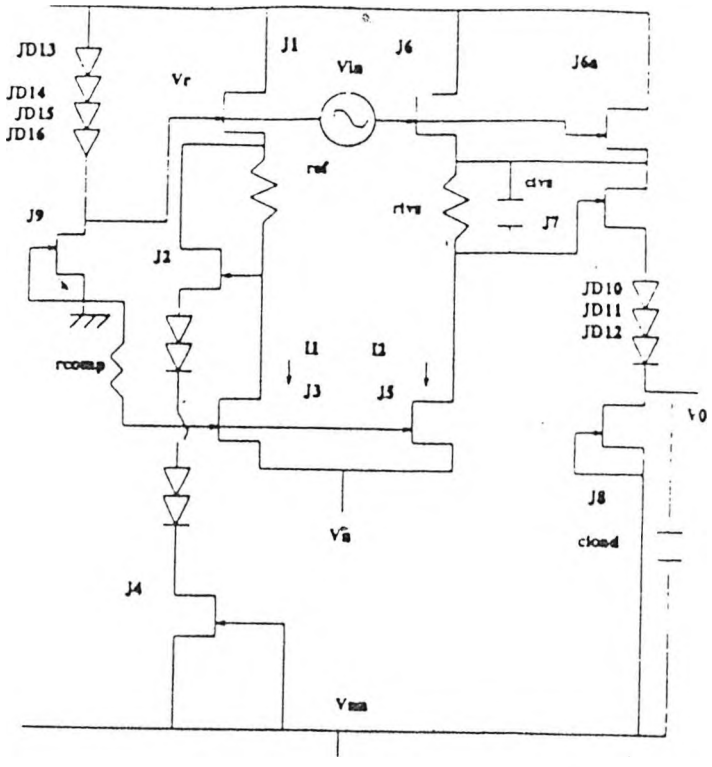
The six steps just described clearly can accommodate other classes of models in Step 1 and other optimizing algorithms in Step 6. We have found that our particular choices make the sequential process efficient.

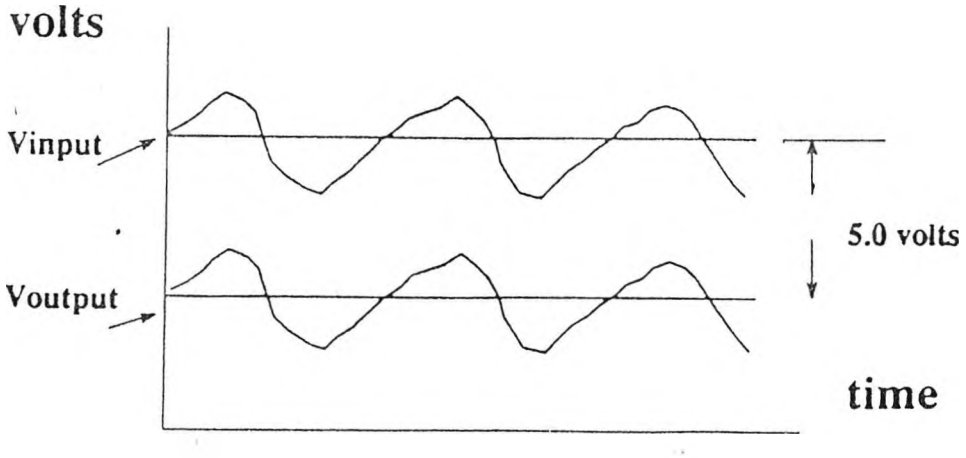
6.3 Voltage-Shifter Circuit Example

Figure 6.1(a) shows a GaAs voltage-shifter circuit. It ideally shifts the circuit input signal by 5 V as in Figure 6.1(b). Such a circuit can be applied to amplifiers or other larger circuits requiring a level-shifting function; in our case it is used for high-frequency transmission. This means that as well as achieving an accurate level shift of 5 V, the circuit should provide a large AC gain and a broad AC frequency bandwidth so that high-frequency systems can operate properly.

As the bandwidth becomes broader, the waveform of the gain may rise before falling. For stability of the circuit, this ripple effect should be minimized. The objective here is to maximize gain and bandwidth while keeping the voltage shift close to the target of 5 V and minimizing ripple.

Table 6.1 gives the ranges of the 14 varying inputs to the circuit simulator. We use logarithmic scales for some factors. Nine of the factors are designable; of these, log(ref), log(cload), log(clvs), and log(j1) ($x_1, \ldots, x_4$ respectively)


JD13
JD14
JD15
JD16
J1
J6
J6a
Vr
Vin
ref
J9
J2
J7
JD10
JD11
JD12
J3
J5
J8
closed
J4


volts
Vinput
Voutput
5.0 volts
time

have significant additive uncontrollable variations. For example, log(ref) has a total range of [log(0.9 KΩ), log(6.4 KΩ)], but a designable range of [log(1.2 KΩ), log(4.8 KΩ)]. The other designable factors, j7, j9, log(rcomp), log(rlvs/ref) and j6/j1 ($x_5, \ldots, x_9$ respectively), have negligible noise components which were set equal to zero. The noise factors, $x_{10}, \ldots, x_{14}$ are noises in the power supplies (vpp, vn, and vnn), the threshold voltage (jmodvto), and the ohmic resistance (jmodrs). The ranges of all variations correspond to $\pm 3\sigma$. All of the total ranges are later normalized to [-0.5, +0.5].

i	Factor	Range of c_i	Range of u_i
1	log(ref in K Ω)	[log(1.2), log(4.8)]	± log(1.33)
2	log(cload in pF)	[log(0.05), log(0.2)]	± log(2)
3	log(clvs in pF)	[log(0.01), log(1.0)]	± log(2)
4	log(j1 in mm)	[log(0.02), log(0.045)]	± log(1.1)
5	j7 in mm	[0.015, 0.045]	---
6	j9 in mm	[0.0075, 0.0225]	---
7	log(rcomp in K Ω)	[log(1.35), log(5.4)]	---
8	log(rlvs/ref)	[log(0.2475), log(1.0)]	---
9	j6/j1	[0.5, 2.0]	---
10	vpp in V	---	[4.5, 5.5]
11	vn in V	---	[-3.3, -2.7]
12	vnn in V	---	[-5.72, -4.68]
13	jmodvto in V	---	[-0.9375, -0.5625]
14	jmodrs in Ω	---	[1.05, 2.45]

Table 6.1 Input factors and their ranges: Voltage-shifter circuit

The performances we model are

y_1 = log(3dB bandwidth), bandwidth measured in GHz,

y_2 = voltage shift (V),

y_3 = gain in dB, the frequency response at 0.1 GHz.

To monitor ripple we also model $y_4, \ldots, y_9$, the gains at frequencies of 0.191, 0.363, 0.692, 1.318, 2.512, and 4.786 GHz. We use as a numerical measure of the ripple:

(6.3.1) $$RIP = \max_{j=3,\ldots,9} (y_j - y_3).$$

Thus RIP measures the size of the upward fluctuation of the frequency response curve.

The most desirable circuit has maximum bandwidth and gain, a voltage shift of 5 V, and zero ripple. These goals cannot be met because of the uncontrollable variations, and because some of these criteria may conflict. We therefore need a means to combine these performances in order to measure the quality of a particular design.

The route we follow is to form a loss statistic to balance the need for good nominal performance and for low variability over the uncontrollable variations. With the notation of Section 6.2, each y_j is a function of $\mathbf{x} = \mathbf{c} + \mathbf{U}$, where $\mathbf{x} = (x_1, \ldots, x_{14})$, and similarly for $\mathbf{c}$ and $\mathbf{U}$. So by good nominal ($\mathbf{U} = 0$) performance, we mean $y_1(\mathbf{c})$ and $y_3(\mathbf{c})$ large, $|y_2(\mathbf{c}) - 5|$ small, and RIP($\mathbf{c}$) small.

To measure the variabilities of bandwidth and gain around the nominal we use

$$Var(j,\mathbf{c}) = \int (y_j(\mathbf{c}+\mathbf{U}) - y_j(\mathbf{c}))^2 \, d\Gamma(\mathbf{U})$$

for j = 1 and 3. Here Γ is the distribution of $\mathbf{U}$, which we take to be independent normals with standard deviations given by 1/6 of the ranges of the u_i's. We also let

$$Var(2,\mathbf{c}) = \int |y_2(\mathbf{c}+\mathbf{U}) - 5|^2 \, d\Gamma(\mathbf{U})$$

measure the variability of the voltage shift around the target. We do not use a corresponding variability measure for ripple; the first stage experiment suggests it is sufficient to only look at nominal ripple.

Formally, the problem we pose is

(6.3.2) $$\max_{\mathbf{c}} \left[y_1(\mathbf{c}) + y_3(\mathbf{c}) - \sqrt{Var(1,\mathbf{c})} - \sqrt{Var(3,\mathbf{c})} \right]$$

subject to

$$\sqrt{Var(2,\mathbf{c})} \leq 0.1V$$

and

$$RIP(\mathbf{c}) \leq 0.01dB.$$

We now apply the six-step modeling and optimization strategy described in Section 6.2 to the loss function and constraints given above. Each performance y_j is modeled as

(6.3.3) $$y_j(\mathbf{x}) = \beta_j + Z_j(\mathbf{x}),$$

but all the p_i's in (6.2.4) are taken to be equal. Since there are nine responses there are nine sets of correlation constants to be estimated.

At the first stage we select a Latin hypercube experimental design of 75 runs for the 14 input factors. Table 6.2 lists the values of y_1, y_2, y_3, and RIP for the first five runs. There are some very badly behaved circuits. For example, runs 2 and 3 have very low gains. Thirteen points in all have gain $< -7\ dB$, and these outlying data are deleted for the statistical modeling (as noted in Step 2).

Run	Volt	Gain	log(BW)	Ripple
1	5.31	-5.58	9.5992	1.09
2	6.53	-15.51	9.9065	0.00
3	6.60	-19.95	9.4026	0.00
4	4.48	-2.65	9.9210	0.00
5	5.44	-2.37	9.7036	0.54

Table 6.2 Performances for the first five experimental-design points: Voltage-shifter circuit.

Model (6.3.3) is fit separately to each of y_1 ,..., y_9 using the remaining 62 runs (Step 3). Typical root mean squared errors associated with the predictors $\hat{y}_1$, $\hat{y}_2$, and $\hat{y}_3$ are 0.05 log GHz, 0.14 V, and 0.30 dB (Step 4). They are sizeable, suggesting the need to reduce the experimental region to improve accuracy.

To guide the choice of the second-stage experimental region (Step 5) we optimize (6.3.2) with respect to the design factors $\mathbf{c}$ and with y_j replaced by $\hat{y}_j$. The integrals in (6.3.2) for $Var(j,\mathbf{c})$ are estimated from 100 Monte Carlo samples of $\mathbf{U}$ from $\Gamma(\mathbf{U})$. Around this tentative optimum we choose a sub-region for $c_1,...,c_9$ using main-effect plots and interaction plots.

Figures 6.2, 6.3, and 6.4 show the main effects of the 14 inputs on bandwidth, voltage, and gain. As the interaction plots contain no exploitable information, it is sufficient to consider these figures only. There are several notable features:

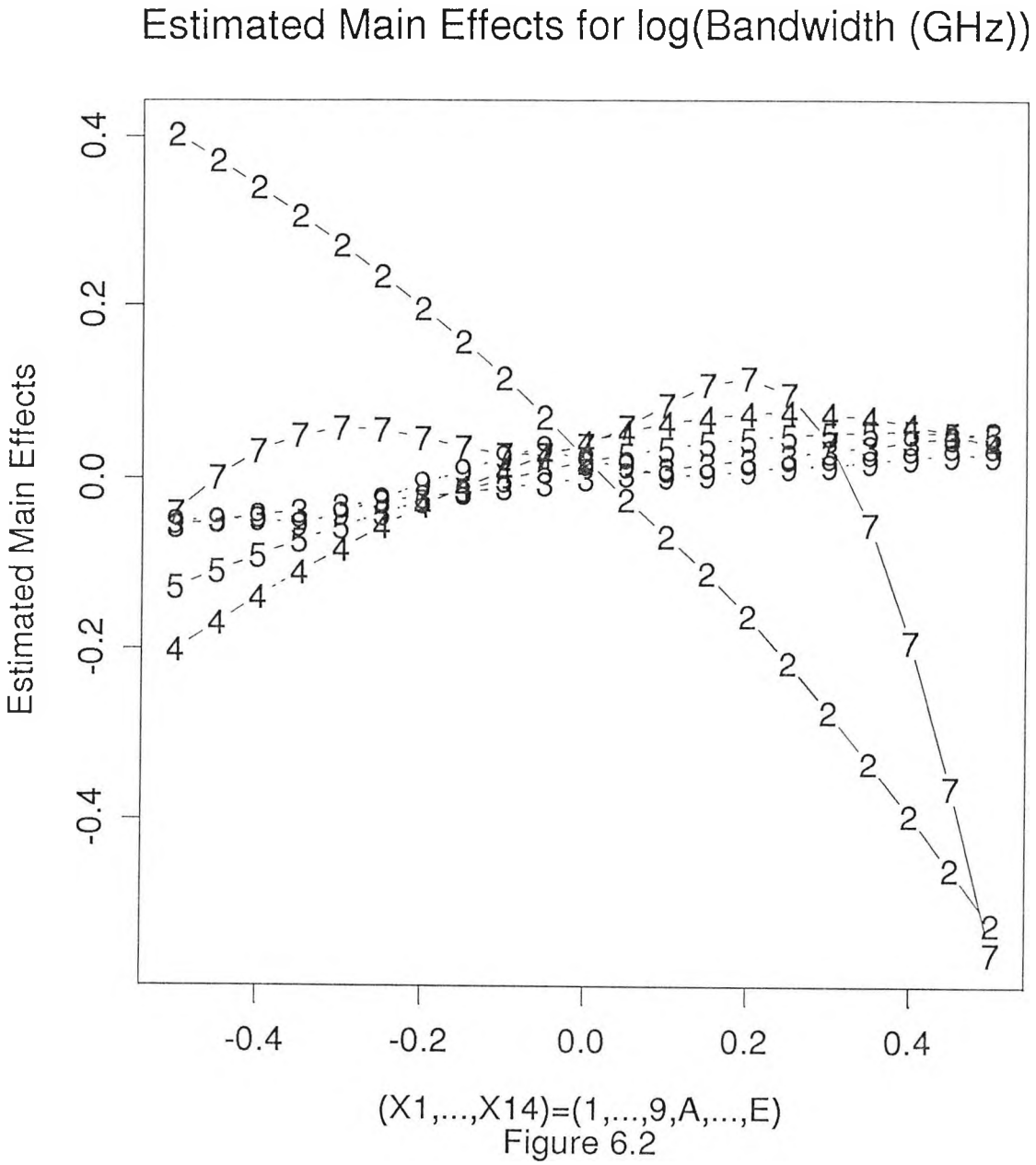
(a) The estimated bandwidth, $\hat{y}_1$, is strongly dependent on cload (x_2), with lower values giving higher values of bandwidth. There is some dependence on j1 (x_4), j7 (x_5), rcomp (x_7), and little effect from the uncontrollable factors $x_{10}, \ldots, x_{14}$.

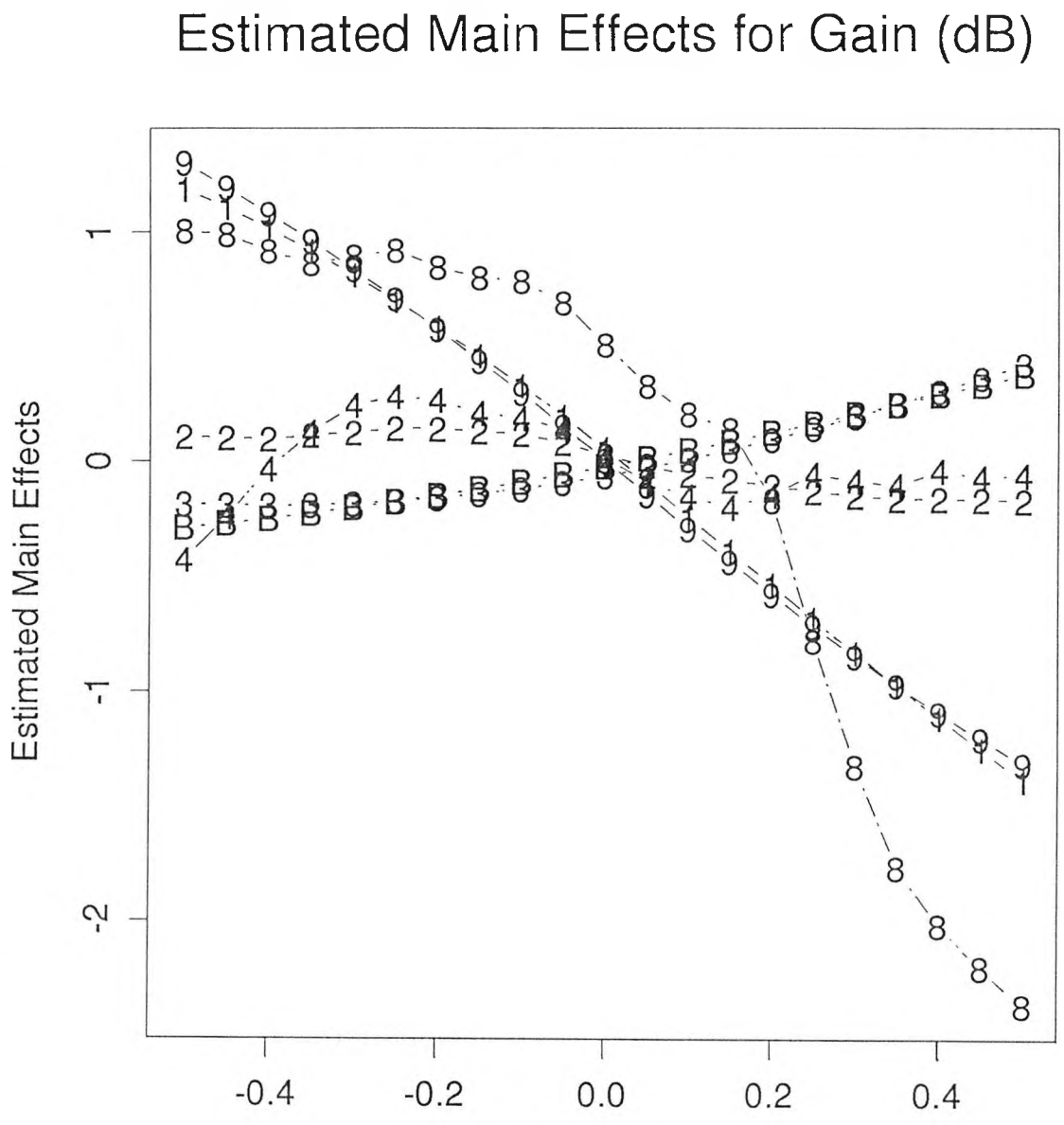
(b) The estimated voltage, $\hat{y}_2$, depends strongly on rlvs/ref (x_8) and j6/j1 (x_9). Also, ref (x_1), j1 (x_4) and rcomp (x_7) have effects of a practical magnitude. This means that there are many tradeoffs between c_8, c_9, etc., that can keep voltage on target. Of considerable importance is the fact that the uncontrollable factors vpp (x_{10}), vn (x_{11}), and jmodvto (x_{13}) have a significant effect on $\hat{y}_2$, indicating potential difficulty in controlling variability of the voltage shift.

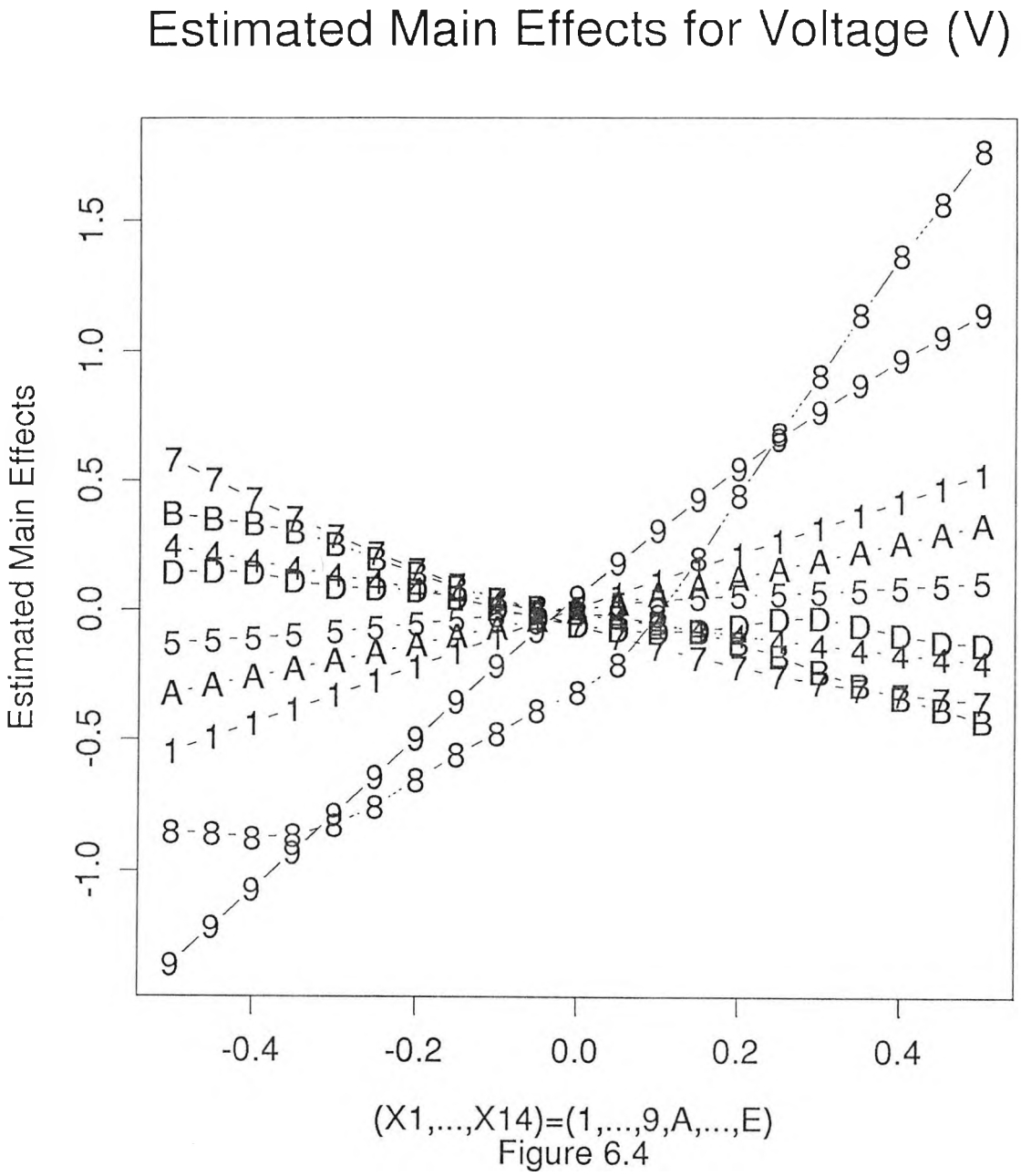
(c) The estimated gain, $\hat{y}_3$, has large effects from ref (x_1), rlvs/ref (x_8), and j6/j1 (x_9), with smaller but practical effects from clvs (x_3), j1 (x_4) and from the uncontrollable factor jmodrs (x_{14}).

Based on the tentative optimization and these plots, we reduce the ranges for the controllable factors. For example, on the normalized, logarithmic scale, the "optimal" value of (c_1) is roughly at its lower bound of -0.35. Figures 6.3 and 6.4 show that the only way to get larger gain values while maintaining voltage shift near 5 is to take small values of x_1. Figure 6.2 shows that x_1 is unimportant for bandwidth. Thus, we choose a fairly tight sub-region for c_1: the interval [-0.353, -0.3]. After adding in ± 0.147, the $\pm 3\sigma$ range for u_1, which cannot be reduced, the total range for x_1 becomes [-0.50,-0.15] after some rounding. Similarly, promising sub-ranges are identified for $c_2, \ldots, c_9$.

We now repeat steps 2 - 4 on the new region. Fifty runs from a Latin hypercube design produce no poor circuits, a reflection of the move to an appropriate part of the space. The root mean squared error of prediction drops to about 0.004 log GHz, 0.035 V, and 0.01 dB for $\hat{y}_1$, $\hat{y}_2$, and $\hat{y}_3$, suggesting that we now have predictors reliable enough for optimization. However, when the "optimal" $\mathbf{c}$ is tested by a confirmatory run for the nominal circuit ($\mathbf{U} = 0$), the nominal voltage is 4.88 V rather than the predicted 4.95 V. Rather than taking more data we add first-order regression terms to the model for voltage, as in


Estimated Main Effects for log(Bandwidth (GHz))
Estimated Main Effects
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
(X1,...,X14)=(1,...,9,A,...,E)
Figure 6.2


Estimated Main Effects for Gain (dB)
Estimated Main Effects
1
0
-1
-2
-0.4
-0.2
0.0
0.2
0.4
(X1,...,X14)=(1,...,9,A,...,E)
Figure 6.3


Estimated Main Effects for Voltage (V)
Estimated Main Effects
-1.0
-0.5
0.0
0.5
1.0
1.5
-0.4
-0.2
0.0
0.2
0.4
(X1,...,X14)=(1,...,9,A,...,E)
Figure 6.4

(6.2.3). This improves the accuracy of the voltage predictor enough so that the estimated optimum meets the required voltage constraints. The new "optimal" c_i's lead to accurate predictions at the nominal ($\mathbf{U} = 0$) and produce low variabilities as shown in Table 6.3. The improvement over the initial design is indicated in Table 6.3. The confirmations for $Var(j,\mathbf{c})$ required a Monte Carlo sample of 100 circuit simulation runs---we do this just for demonstration purposes. The Monte Carlo sample is obtained by picking 100 random vectors **U** according to their distribution.

	Initial	Nominal		$\sqrt{Var(j,\mathbf{c})}$	
		Predicted	True	Predicted	True
log(bandwidth (GHz))	log(3.33)	log(6.62)	log(6.41)	.044	.045
voltage (V)	4.99	4.975	4.929	.091	.111
gain (dB)	-2.786	-1.913	-1.928	.037	.038
ripple (dB)	0.000	0.000	0.000	---	---

Table 6.3 Nominal performances and variabilities at the second-stage optimal **c**: Voltage-shifter circuit.

6.4 Output Buffer Example

The example is an output buffer. Output buffers translate logic signals between integrated circuits and external connections. Of particular interest to the designer when considering components are:

1. The time between state transitions.
2. The control of voltage spikes due to changing currents.

These two performance criteria are in direct conflict, i.e. faster switching means more noise and so a trade-off is necessary.

The example is a proprietary circuit from INTEL so certain information has been left out or altered to mask the true results. The input ranges for the experimental plan are scaled to $[-0.5, 0.5]$ for analysis and this is the range in which the results will be reported. Also, the values used for targets in the constraints have been replaced with constants, K_ν, where ν is the variable associated with the constant.

There are 11 input variables used in experimentation, ten device sizes (T_i, P_j, N_j $i=1,2$ and $j=1,...,4$) and C_{load}. All the input variables were assumed for this exercise to not have uncontrollable variation, i.e. $\mathbf{U}=0$. The ranges for these input variables covered a large area to look for as many solutions as possible. No nominal settings were given as in the example in Section 6.3.

There are 16 response variables used to define the specifications for the circuit. There are two types of specifications, constraints and targets. The response variables are:

1. The delays in nanoseconds, T_{DH} and T_{DL}.
2. The four power supply noises in volts, V_{CTDH}, V_{CTDL}, V_{STDH}, and V_{STDL}.
3. The two drive strengths in volts, V_{UP} and V_{DN}.
4. The two DC Current drives in milliamps, I_{OL} and I_{OH}.
5. The two loaded output transition times in nanoseconds, T_R and T_F.
6. The four peak currents in milliamps, I_{STDL}, I_{STDH}, I_{CTDL}, and I_{CTDH}.

The goal is to find a circuit that minimizes the four power supply noises given a value of the input variable C_{load} and a set of constraints on the remaining 12 response variables. The three values of C_{load} that were investigated are -0.25, 0.0, and 0.25. The general requirements for the optimal circuit design are the following.

Given the following constraints:

1. $T_{DH} < K_{TDH}$ and $T_{DL} < K_{TDL}$.
2. $I_S = K_{IS} I_{STDL} - I_{STDH} > 0$ and $I_C = K_{IC} \mid I_{CTDH} \mid - \mid I_{CTDL} \mid > 0$.
3. $T_R < K_{TR}$ and $T_F < K_{TF}$.
4. $I_{OL} > K_{IOL}$ and $I_{OH} > K_{IOH}$.
5. $V_{UP} > K_{VUP}$ and $V_{DN} < K_{VDN}$.

Find the device sizes that minimize $V_{\max} = \max(V_{STDL}, V_{STDH}, V_{CTDH}, V_{CTDL})$.

6.4.1 Results

For this example it took two stages, each with an experimental plan of 75 runs, to produce statistical models accurate enough to locate an accurate estimate of the input factor values for the optimal circuit. We found that for $C_{load}=0.25$ there were no viable solutions. For $C_{load}=0.0$ we were able to find factor

values so that $V_{max}=0.875V$. For $C_{load}=-0.25$ we found factor values so that $V_{max}=0.595V$.

A question posed at the conclusion of the second stage led to a third stage experiment. The goal for this stage was to find the minimum V_{max} at $C_{load}=-0.25$ and $C_{load}=0.0$ with no constraints except for the delay constraints. For $C_{load}=0.0$ we were able to find factor values so that $V_{max}=0.765V$. For $C_{load}=-0.25$ we found factor values so that $V_{max}=0.500V$. The rest of this section gives a detailed account of how these solutions were located.

Stage 1

The primary goal of this stage was to reduce the size of the problem. This was accomplished by reducing the region of the input space where the search for the optimal factor values was conducted and by trying to reduce the number of response variables that needed tending. Since we were looking for results at three separate values of C_{load} there are two possible strategies for reducing the region, separate regions for each value of C_{load} or one region which is large enough to contain the solutions for all C_{load} values of interest. After analyzing the data from the first stage a single subregion was used to search for solutions for all values of C_{load}.

There are three ways in which the number of response variables can be reduced:

1. Several response variables can be combined into one function.
2. They are superfluous for the circuit specifications given.
3. They can be made superfluous by restricting the search area to a region where the response always meets the relevant constraint.

From viewing the scatter plots of input vs. response it was apparent that there were many observations, particularly on the edges of the input space, that had response values far from the performance specifications. This led to the use of three "sub" designs. The subdesign for T_R, T_F, T_{DL}, and T_{DH} contained 63 points. The subdesign for the response variables V_{STDL}, V_{STDH}, V_{CTDL}, and V_{CTDH} contained 67 points. The third subdesign used 59 points and was used to model the two constraints I_C and I_S.

It was also apparent that four of the response variables: V_{UP}, V_{DN}, I_{OL}, and I_{OH} are dependent on a single input variable, either T_1 or T_2, so there was no

need to model these responses. See Figure 6.5a-d. From the graphs of the experimental data it is clear that to meet the constraints on V_{UP}, V_{DN}, I_{OL}, and I_{OH} that T_1 and T_2 have to be greater than -0.25. Because of the clear relationship between these responses and the input factors the number of response variables which need to be considered can be reduced to twelve.

There were two opportunities to combine response variables into a single function. We started by modelling I_S and I_C and their corresponding constraints. By generating predictors for I_S and I_C as well as for the individual responses we found that the cross-validation ERMSE (3.4.1) for I_S and I_C were only slightly larger than the cross-validation ERMSE for I_{STDL} and I_{CTDH} respectively, while the CV RMSE for I_{STDH} and I_{CTDL} were an order of magnitude larger. From these results a decision was made to use I_S and I_C rather than the individual responses. We also considered a similar approach for $V_{\max}$; this is a much more complex function and did not give as accurate a predictor as $max(\hat{V}_{STDL}, \hat{V}_{STDH}, \hat{V}_{CTDL}, \hat{V}_{CTDH})$ so was abandoned. By considering these options we were able to reduce the number of "response variables" from 12 to 10.

The remaining 10 responses are modeled as (6.3.7); CV RMSE estimates and main effects plots were produced as well. The CV RMSE estimates showed that the statistical models do not predict the response variables accurately enough for making precise statements about an optimal solution. The main effects plots showed that most variables depend on C_{load} and that response variables could be divided into roughly two groups, those that depend on T_1, P_1 and N_3 and those that depend on T_2, N_2 and P_4. See Table 6.4 for a list of significant input variables for all responses and the CV ERMSE for the respective response variables. Some of the more important interactions are :

1. T_1, P_3 for I_S.
2. T_2, N_2 and T_2, C_{load} for T_{DH}.
3. T_2, N_3 for T_{DL}.
4. P_1, P_2 for V_{CTDL}.
5. N_2, C_{load} for V_{STDH}.

Since the predictors were not sufficiently accurate, the goal at this time is not to find a precise value of an optimal solution, but to narrow our search to a

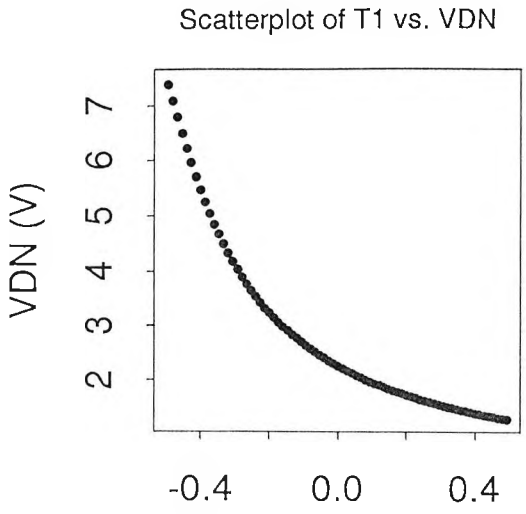


Figure 6.5(a)

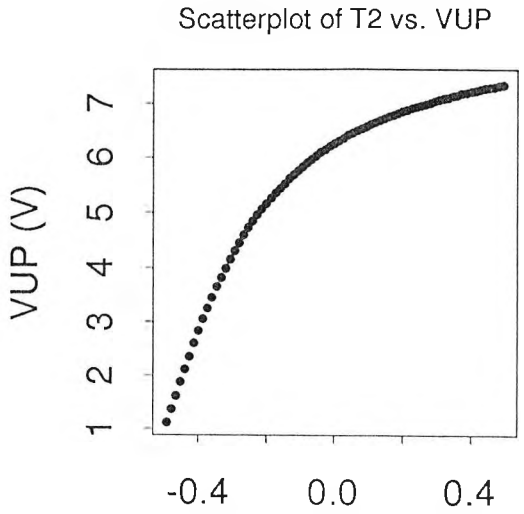


Figure 6.5(b)

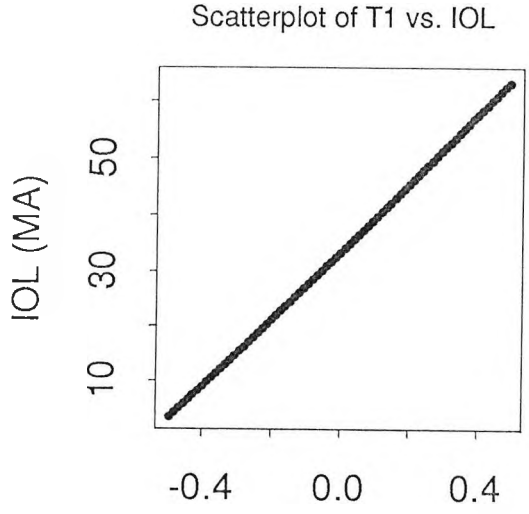


Figure 6.5(c)

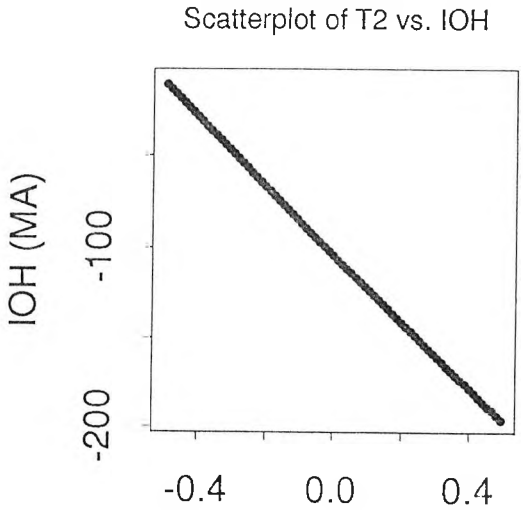


Figure 6.5(d)

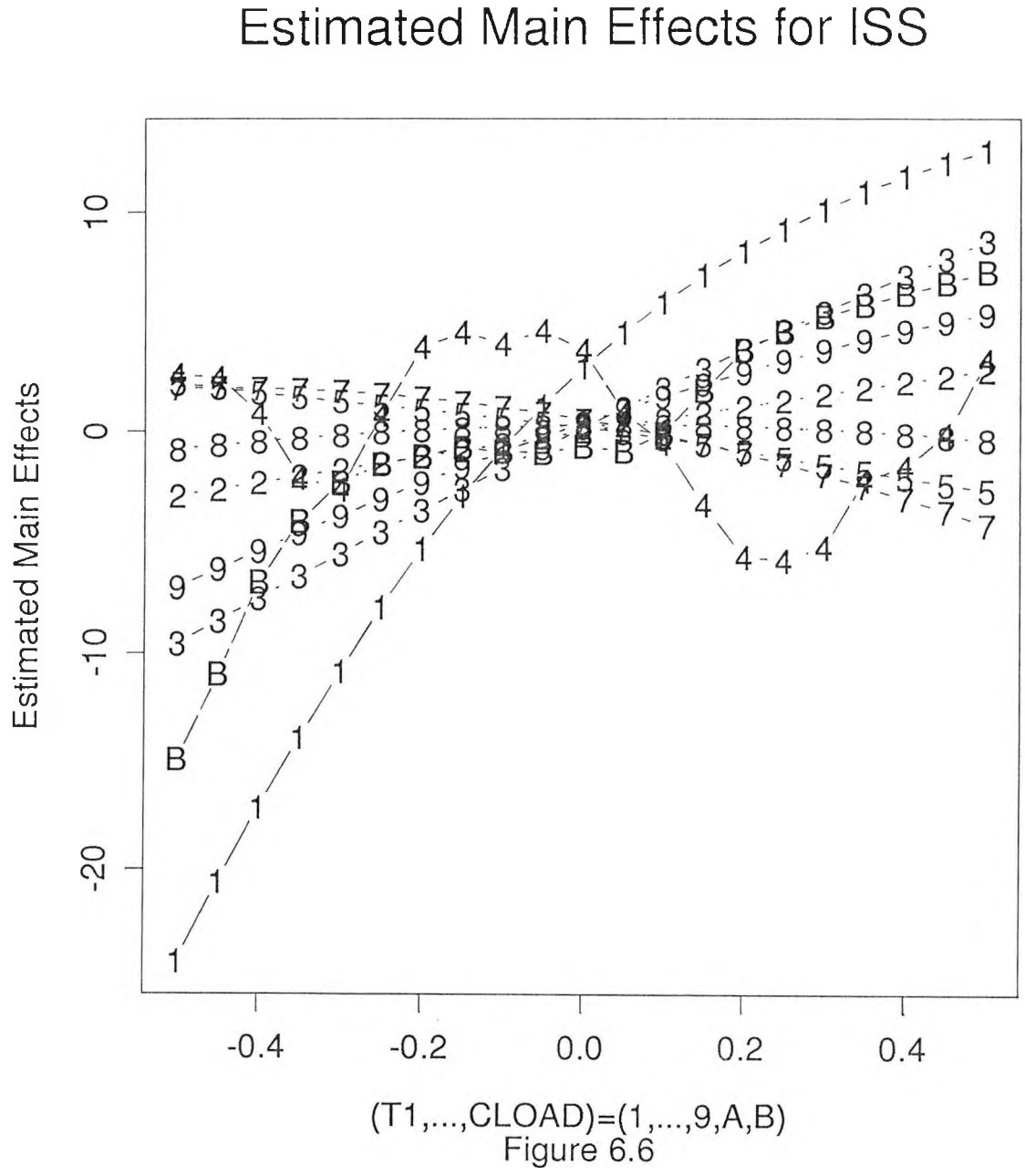
smaller region of the original input space. The next step (Step 5) was to use the statistical models of the response variables to determine the subregion of the input space where the optimal configuration of factor values was located.

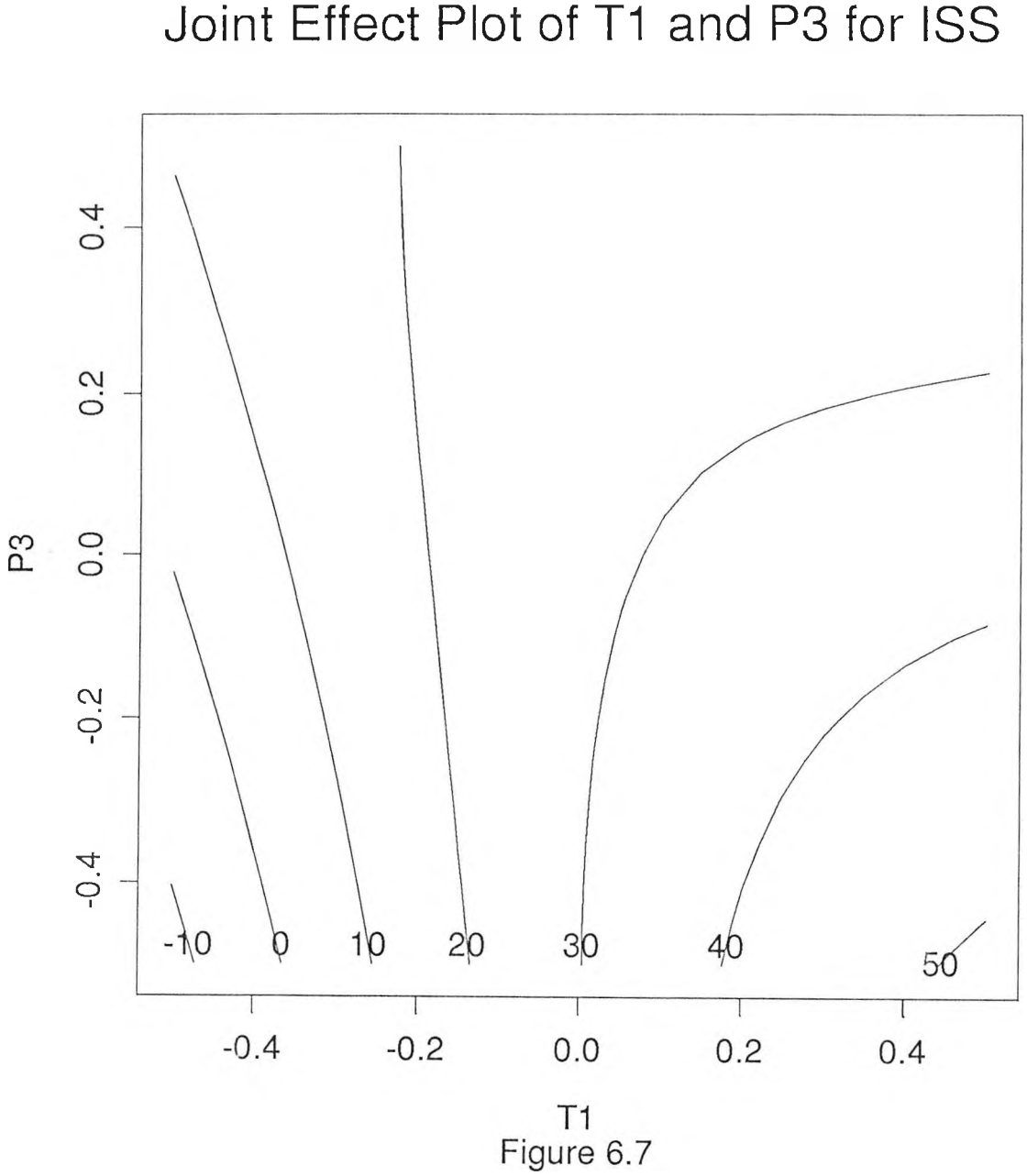
Response Variable	Influential Input Variables	CV ERMSE	Data Range
T_F	T_1, C_{load}	0.82	20.1
T_R	T_2, C_{load}	0.80	22.3
I_S	$T_1, C_{load}, P_1, P_4, N_1, P_3$	4.86	82.75
I_C	$N_3, P_1, N_2, T_1, C_{load}, P_4, N_3, P_2$	13.8	133.75
T_{DH}	$T_2, C_{load}, P_4, N_2, P_2, P_3, P_1$	2.14	27.4
T_{DL}	$C_{load}, T_2, T_1, P_1, N_3, N_1$	1.27	13.9
V_{CTDH}	$N_2, T_2, P_4, C_{load}, N_4, P_2, T_1$	0.08	1.07
V_{CTDL}	$P_2, P_1, N_4, T_2, N_1, N_2, P_4$	0.10	1.07
V_{STDH}	$T_1, N_1, P_3, C_{load}, N_3, N_2, N_4, P_2$	0.10	0.82
V_{STDL}	$P_1, T_1, N_4, T_2, N_1, C_{load}, N_3$	0.13	1.48

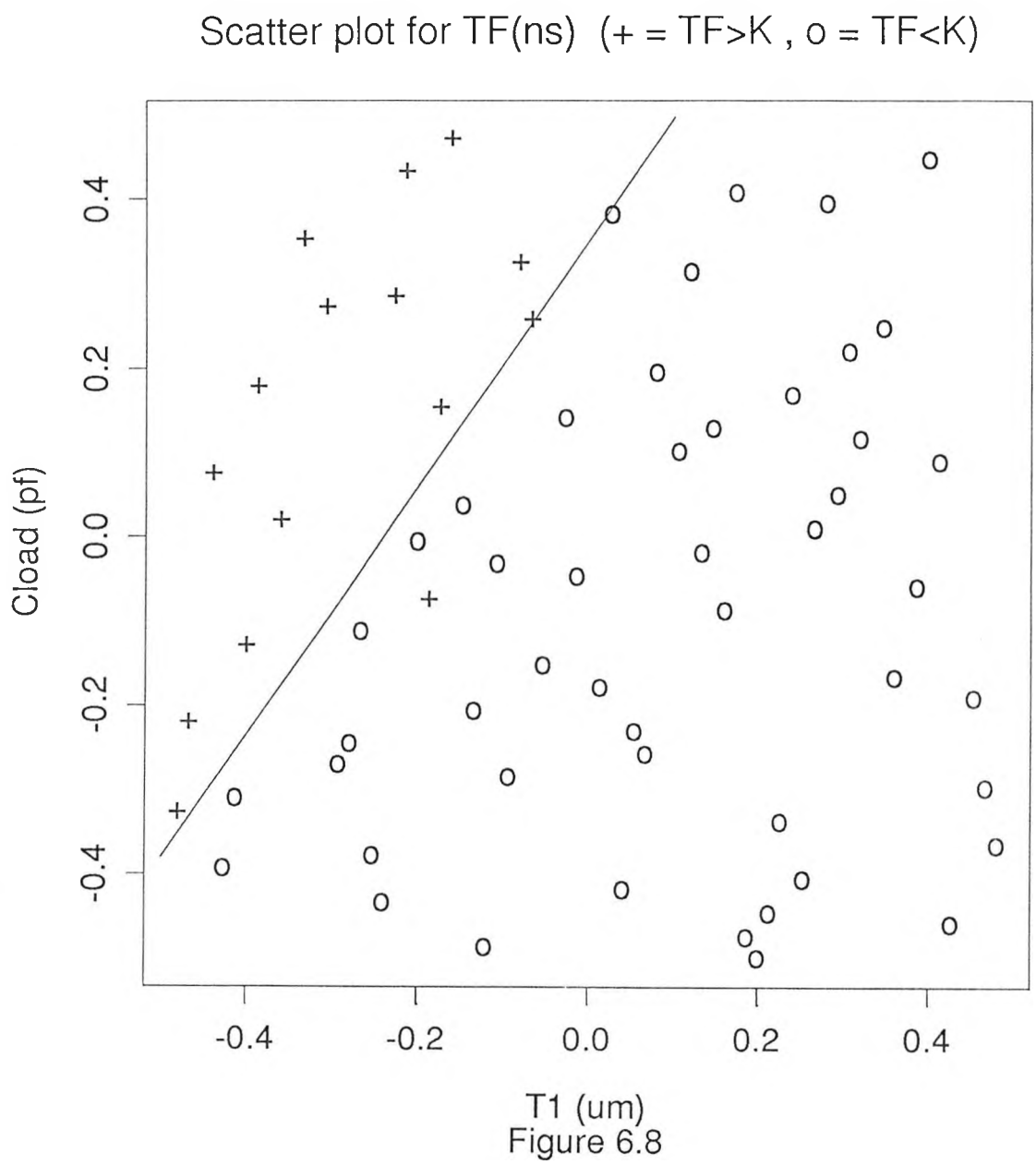
Table 6.4 Influential Variables in Stage 1 in order of importance.

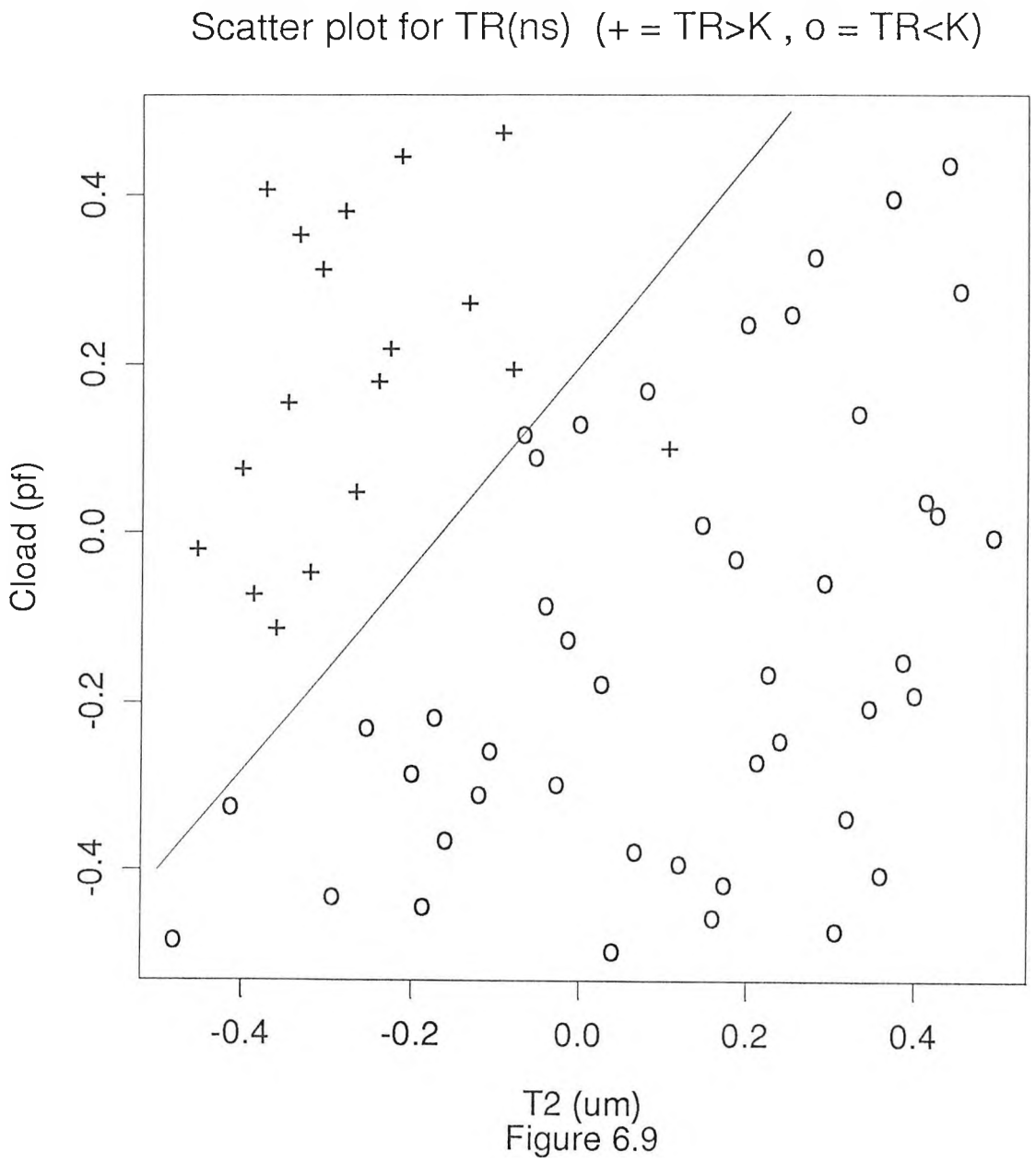
First, response variables that are only affected by two inputs were studied. The main effects plots, see Figs. 6.6 and 6.7, show that the I_S constraint is met when T_1 and $T_2 > -0.25$, so can be dropped along with V_{UP}, V_{DN}, I_{OL}, and I_{OH}. The models for T_R and T_F show that T_1 or T_2 and C_{load} are the only influential variables. Figure 6.8 shows that the equation $aT_1 + b*C_{load} < K_{TF}$, gives an accurate picture of the relationship between T_1 and T_F. The variables T_2 and T_R have a similar relationship as is shown in Figure 6.9. These functional estimates show that when $C_{load} < 0.0$ the constraints for T_R and T_F are met if T_1 and T_2 are > -0.25 and when $C_{load} > 0.0$ the constraints are met if T_1 and $T_2 \geq 0.0$. By dealing with just T_1 and T_2 we have reduced the number of response variables to seven.

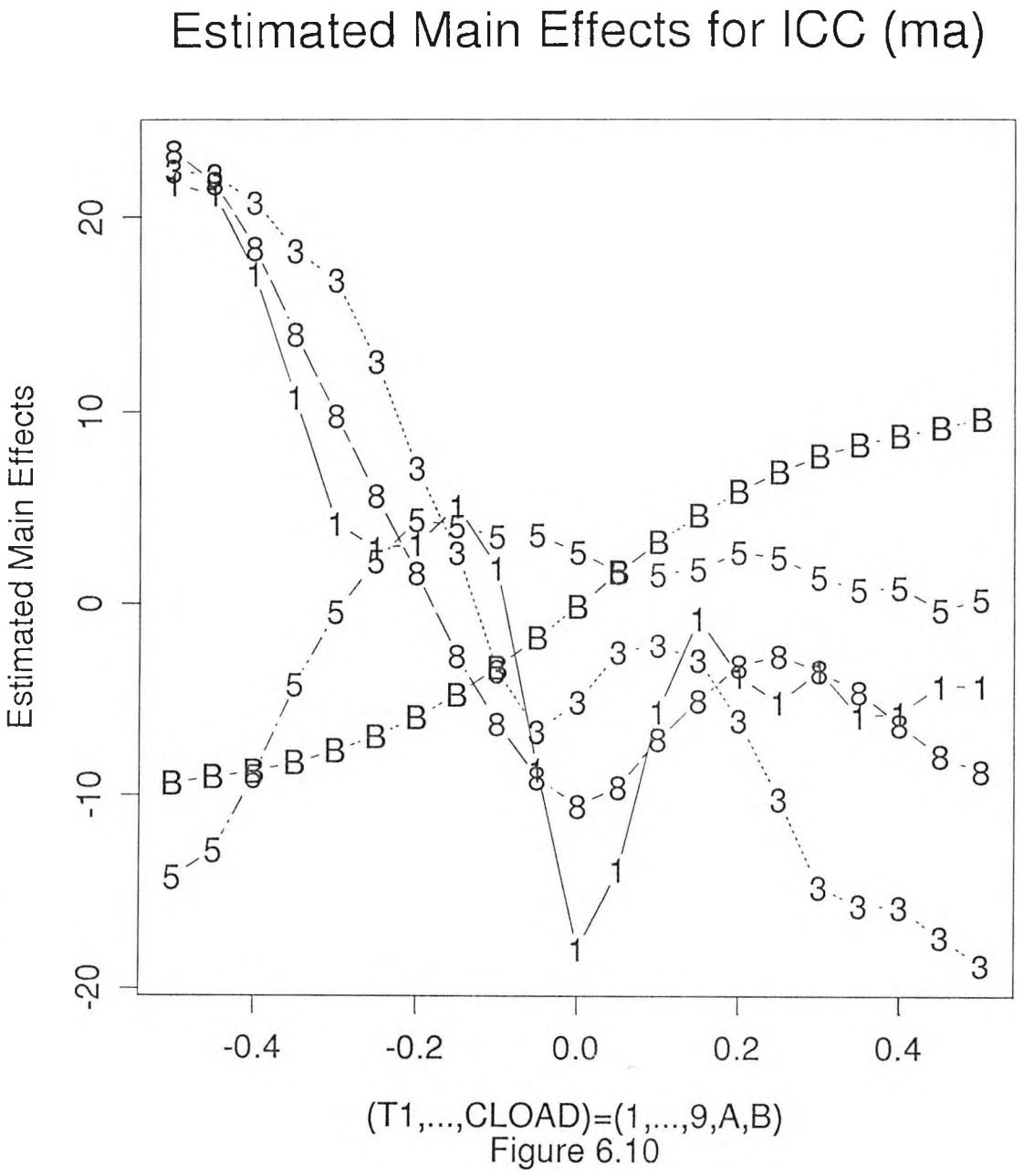
The problem has now been reduced to finding the region(s) in the factor space that meet the constraints on T_{DH}, T_{DL}, and I_C and minimizes $V_{\max}$. The main effects plots, Figs. 6.10-6.16, indicate that the inputs and responses could


Estimated Main Effects for ISS
Estimated Main Effects
10
0
-10
-20
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)
Figure 6.6


Joint Effect Plot of T1 and P3 for ISS
P3
0.4
0.2
0.0
-0.2
-0.4
-10
0
10
20
30
40
50
-0.4
-0.2
0.0
0.2
0.4
T1
Figure 6.7


Scatter plot for TF(ns) (+ = TF>K , o = TF<K)
Cload (pf)
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
T1 (um)
Figure 6.8


Scatter plot for TR(ns) (+ = TR>K , o = TR<K)
Cload (pf)
0.4
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
T2 (um)
Figure 6.9


Estimated Main Effects for ICC (ma)
Estimated Main Effects
20
10
0
-10
-20
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)
Figure 6.10

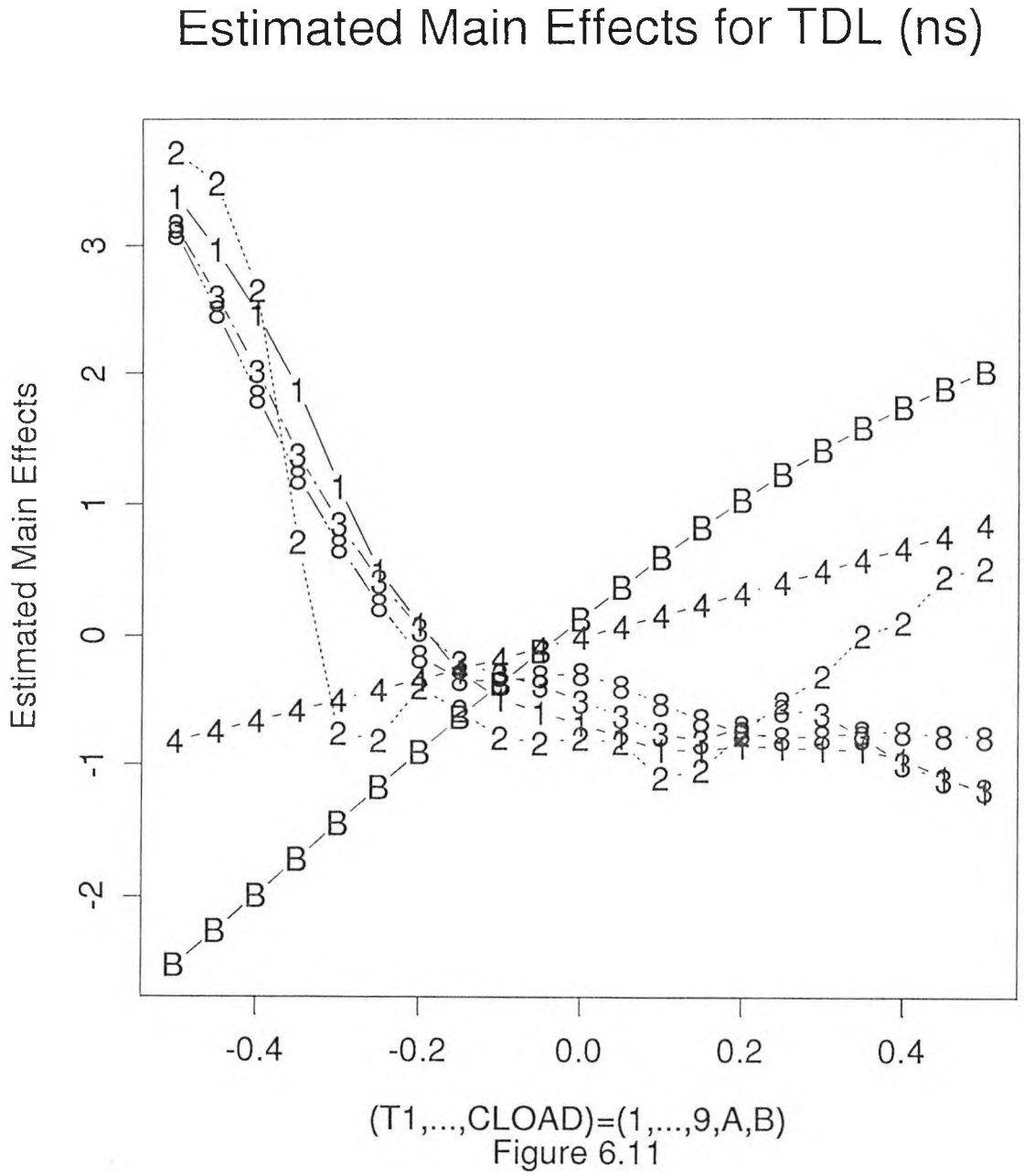
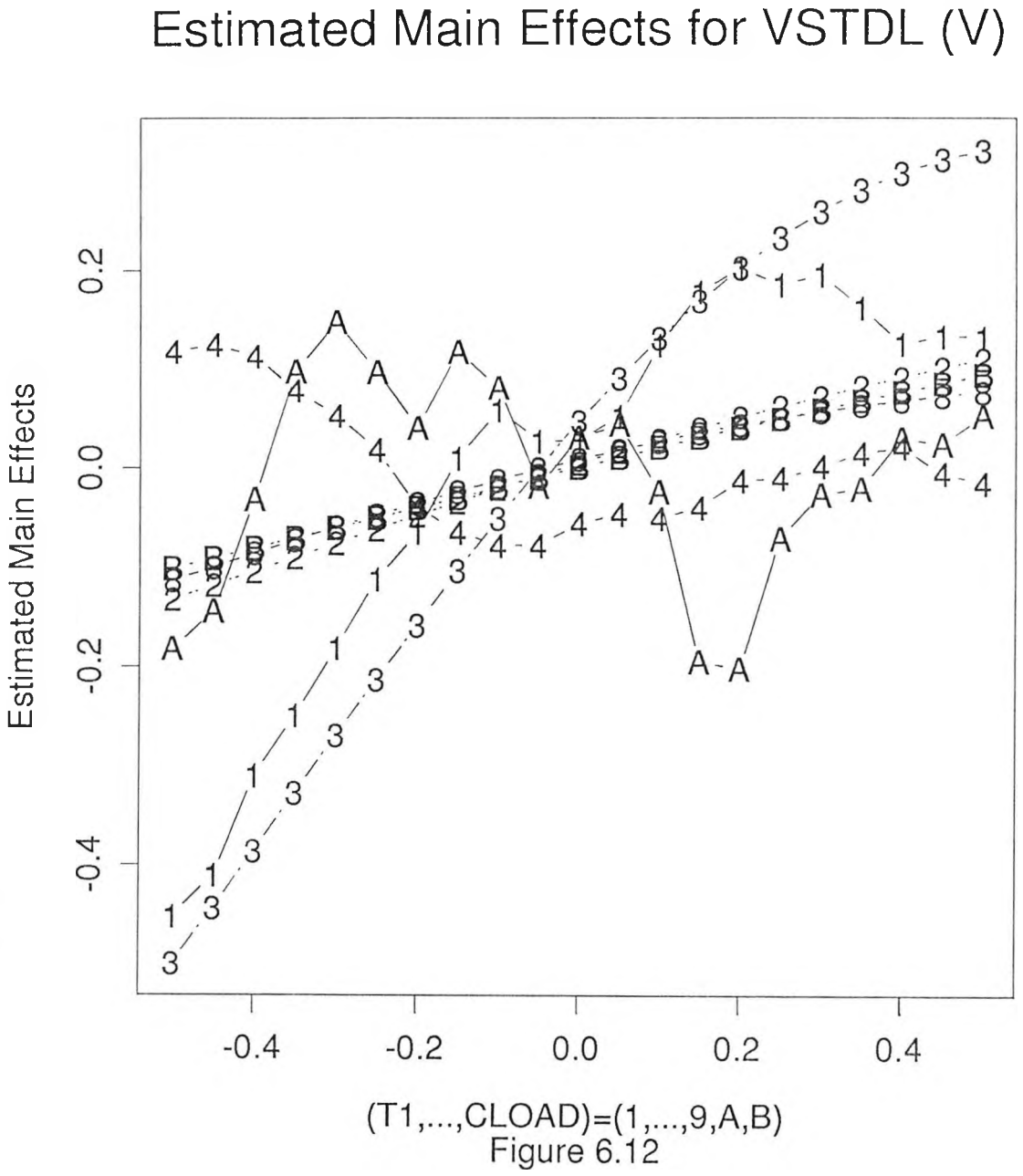
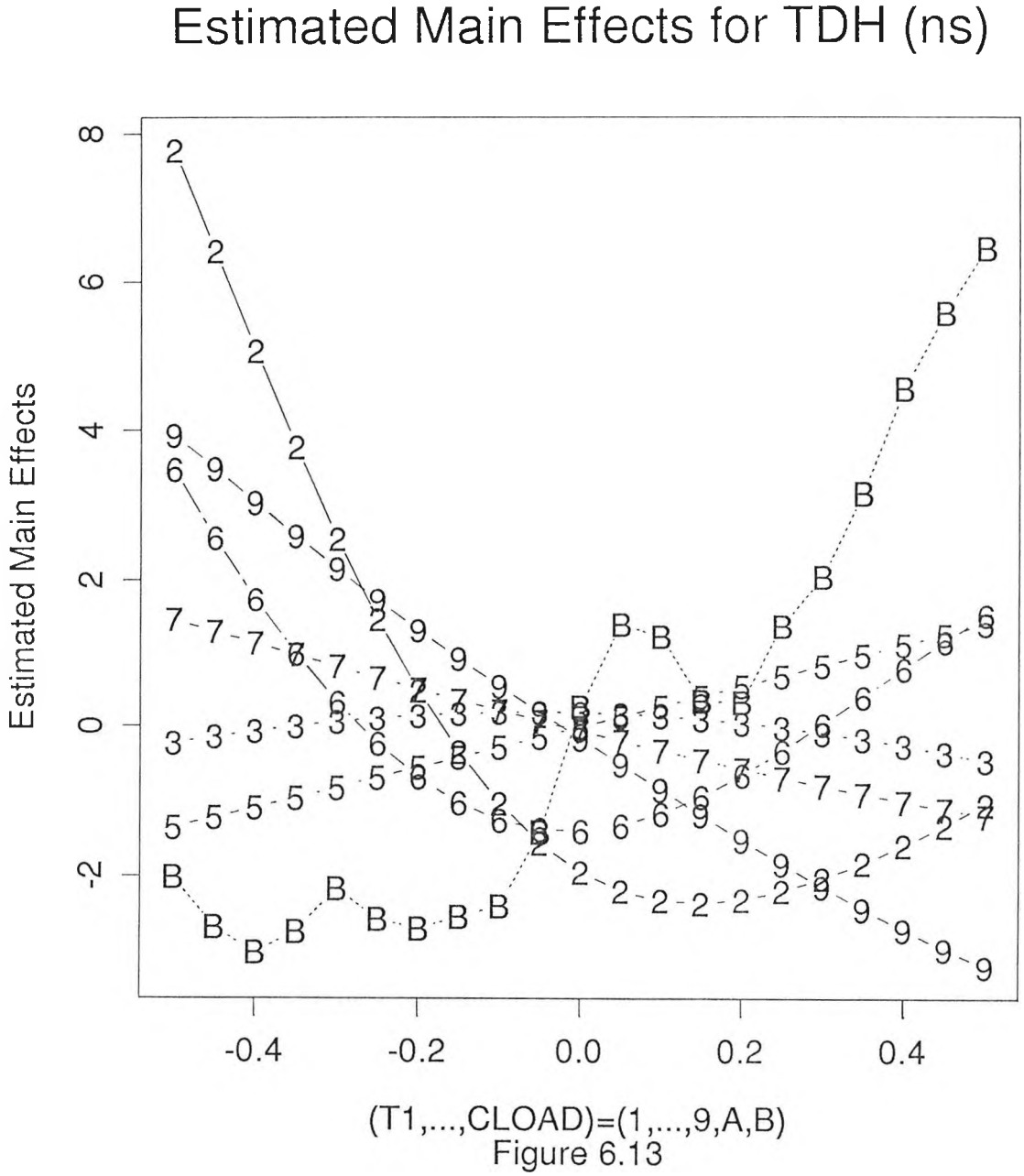
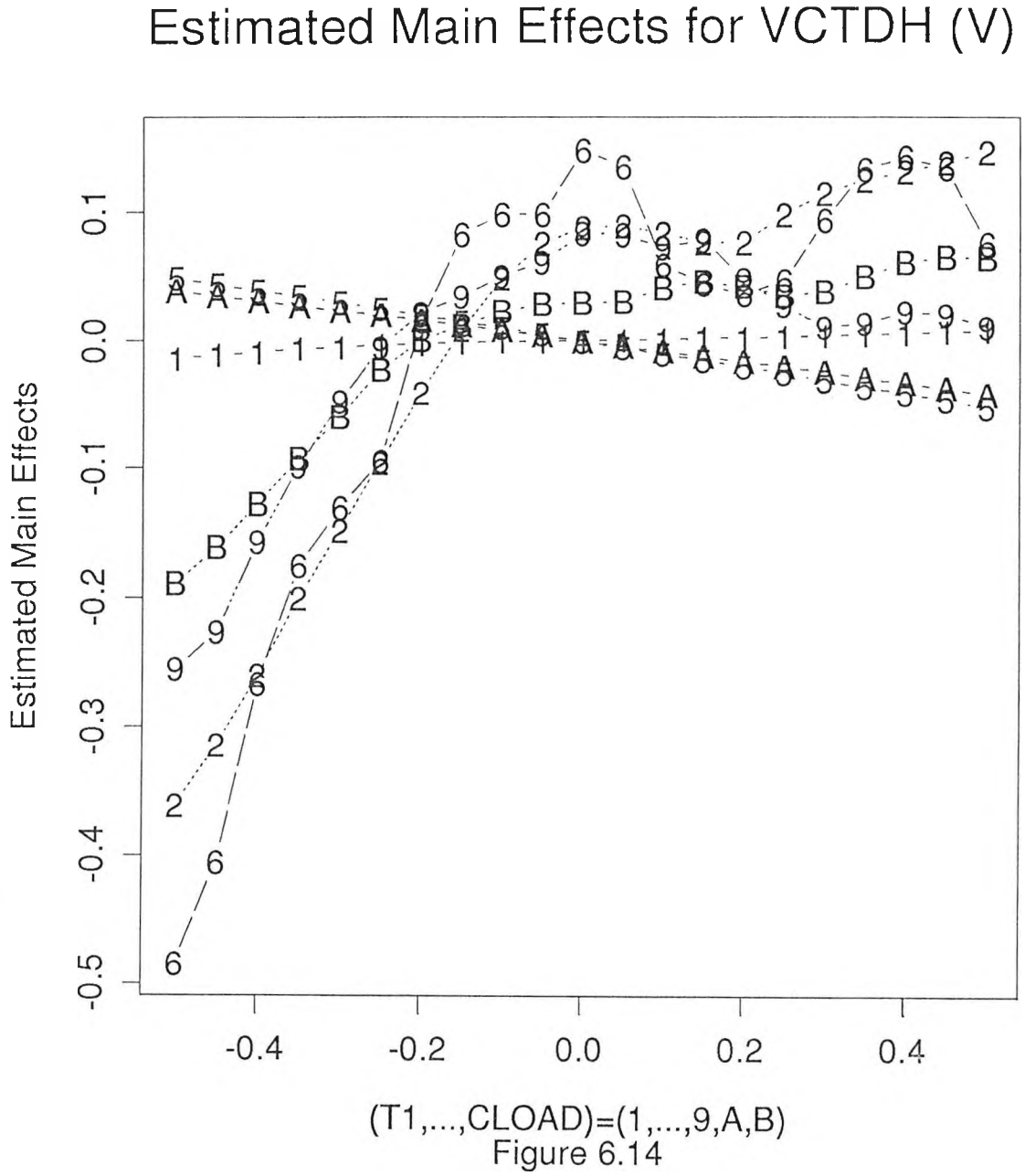

Estimated Main Effects for TDL (ns)
Estimated Main Effects
3
2
1
0
-1
-2
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)

Figure 6.11


Estimated Main Effects for VSTDL (V)
Estimated Main Effects
0.2
0.0
-0.2
-0.4
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)
Figure 6.12


Estimated Main Effects for TDH (ns)
Estimated Main Effects
8
6
4
2
0
-2
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)
Figure 6.13


Estimated Main Effects for VCTDH (V)
Estimated Main Effects
0.1
0.0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)
Figure 6.14

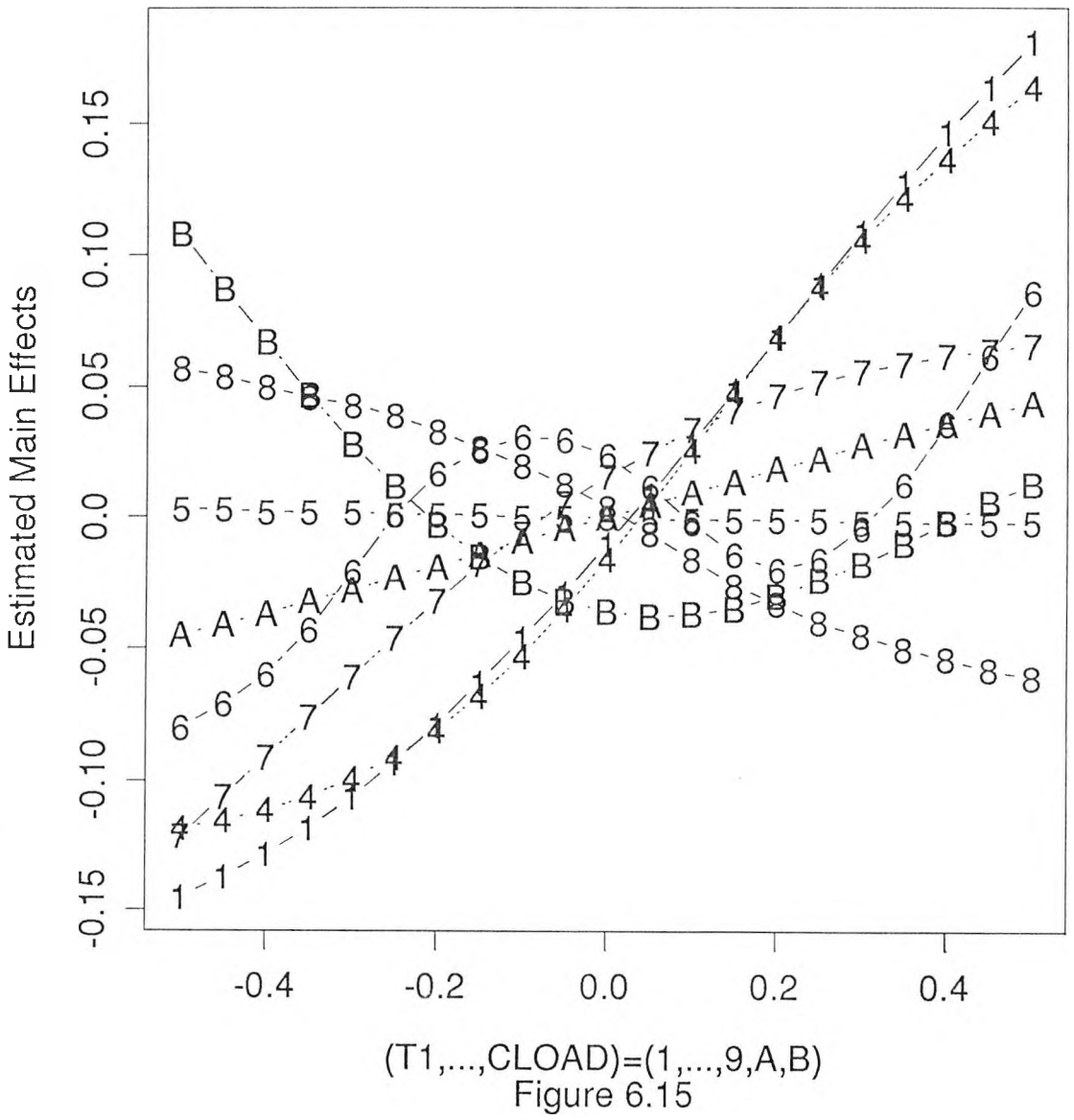
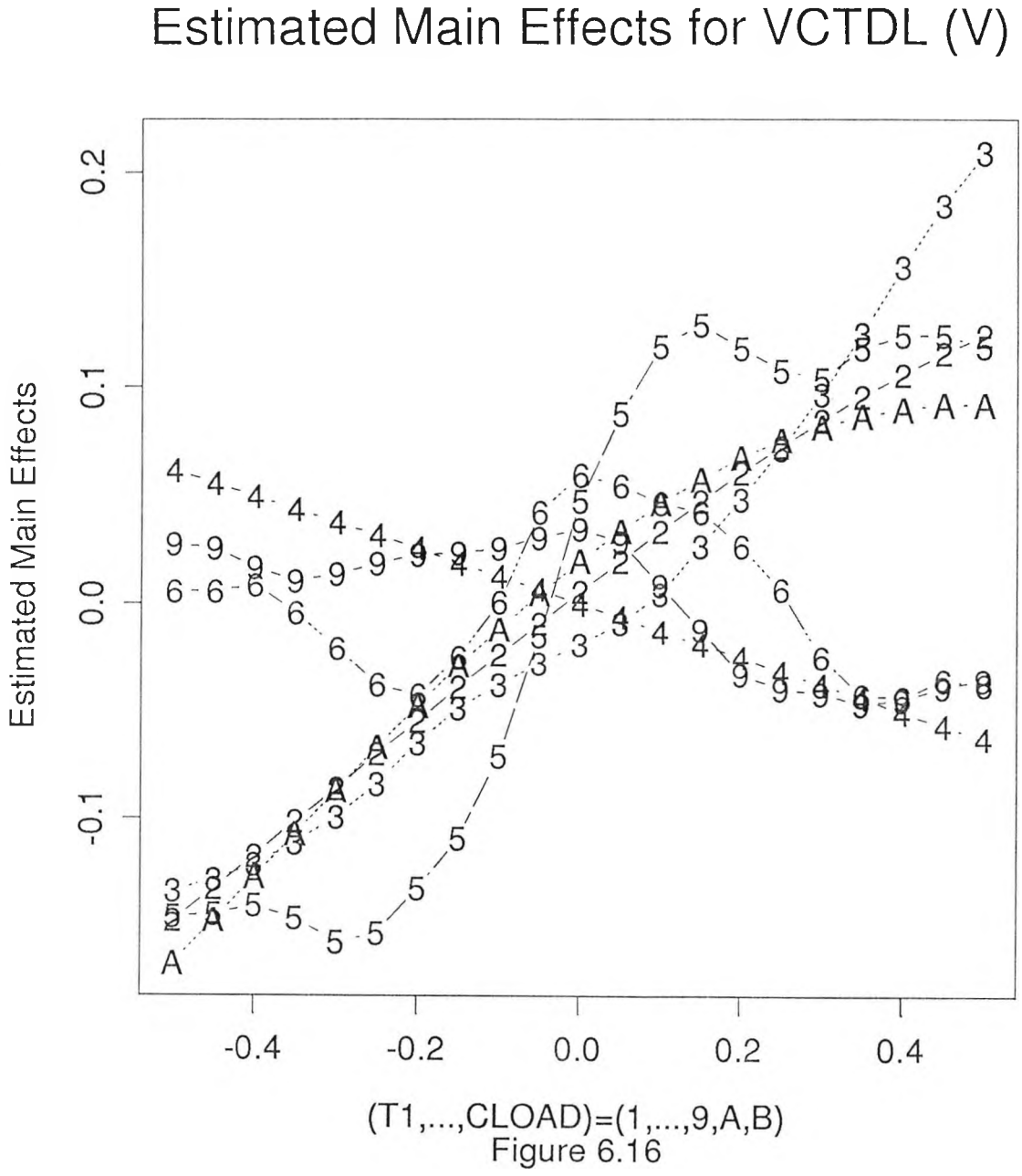

Estimated Main Effects for VSTDH (V)
Estimated Main Effects
0.15
0.10
0.05
0.0
-0.05
-0.10
-0.15
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)

Figure 6.15


Estimated Main Effects for VCTDL (V)
Estimated Main Effects
0.2
0.1
0.0
-0.1
-0.4
-0.2
0.0
0.2
0.4
(T1,...,CLOAD)=(1,...,9,A,B)
Figure 6.16

be roughly divided into two groups. T_1 and P_1 influence T_{DL}, I_C, and V_{STDL}, while T_{DL} and I_C are also strongly influenced by N_3. The response variables T_{DH} and V_{CTDH} are affected by T_2, N_2, and P_4. V_{STDH} and V_{CTDL} appear to be affected by many variables making them easier to adjust and are typically the smallest of the four noises over the region, allowing us to focus our attention elsewhere initially.

The question now is, how can the noise variables be minimized and still meet the delay and I_C constraints? First, the main effects plots show the region of T_2 that does the most to reduce V_{CTDH} is out of bounds due to other constraints. This leads us to consider keeping T_2 to the right and N_2 and P_4 as far left as the T_{DH} constraint allows. This also works well for the T_R constraint, because we now do not need to worry about $C_{load} > 0.0$. The main effects plots show that P_2 only influences I_C and V_{CTDL}. If we let P_2 be slightly negative we maximize the positive effect on I_C while reducing V_{CTDL}.

Input Variable	Stage1 Range	Stage2 Range	Stage3 Range
T_1	[-0.5,0.5]	[0.0,0.3]	[-0.4,0.1]
T_2	[-0.5,0.5]	[-0.1,0.3]	[-0.1,0.3]
P_1	[-0.5,0.5]	[-0.4,-0.1]	[-0.4,0.0]
N_1	[-0.5,0.5]	[0.0,0.25]	[-0.3,0.1]
P_2	[-0.5,0.5]	[-0.2,-0.1]	[-0.4,-0.1]
N_2	[-0.5,0.5]	[-0.5,0.0]	[-0.5,0.0]
P_3	[-0.5,0.5]	[0.0,0.2]	[0.0,0.2]
N_3	[-0.5,0.5]	[-0.4,0.0]	[0.0,0.4]
P_4	[-0.5,0.5]	[-0.4,0.25]	[-0.4,0.25]
N_4	[-0.5,0.5]	[0.25,0.5]	[0.25,0.5]
C_{load}	[-0.5,0.5]	[-0.4,0.3]	[-0.4,0.3]

Table 6.5 Regions of Inputs Modeled at 4 Different Stages

We then move on to dealing with T_{DL}, V_{STDL}, and I_C. The main components of these three responses are T_1, P_1 and N_3. For all three responses the

affect of these three inputs are nearly identical. For example, decreasing T_1, P_1 or N_3 from .1 to -.2 increases the value for T_{DL} by the same amount. We are now in somewhat of a quandary, decreasing T_1, P_1 or N_3 increases I_C and decreases V_{STDL} which is the goal. However, decreasing these inputs increases T_{DL} which we do not want. The problem is to find a favorable tradeoff for these three variables. After selecting the ranges for these variables we choose the remaining input ranges to minimize V_{STDH} and V_{CTDL}. The subregion used in the next stage is in Table 6.5.

Stage 2

The data from the second stage experiment show that the region selected meets the constraints for T_R, T_F, and I_S as expected. We also find that the constraint for I_C is fulfilled over the full region and that V_{CTDL} is always less than one of the other noise variables. This reduces the number of response variables that need to be modelled to five. We compute the parameter estimates for our statistical models of the responses. After looking at the CV ERMSE (Table 6.6) we decide our models are accurate enough to make a serious attempt at finding an optimal circuit design.

Response Variable	Influential Input Variables	CV ERMSE
T_{DL}	C_{load}, P_1, T_1, P_3, P_2	0.05
T_{DH}	T_2, C_{load}, N_1, T_1, N_2	0.11
V_{CTDH}	N_2, T_2, C_{load}, P_3, N_1, N_4,N_3, T_1	0.05
V_{STDH}	N_1, N_2, C_{load}, P_2, N_4, T_1, T_2, P_3, P_1	0.09
V_{STDL}	P_1, N_1, P_2, C_{load}, T_2, P_4, N_4, N_3	0.07

Table 6.6 Influential Variables in Stage 2 in order of importance.

The number of constraint variables that still may be an issue has been reduced to the two delays. The objective function therefore is now:

1. Given: $T_{DL}<K_{TDL}$ and $T_{DH}<K_{TDH}$.
2. Find V_{max}.

Since V_{CTDL} is always less than the other peak noises we feel it can be eliminated from the computation of $\hat{V}_{\max}$, to generate some savings in the cost of running the optimizer. Since the values of the delays that are used in the optimization are only estimates of the true values it is prudent to run the optimization for constraints $K_{TDL} \pm \varepsilon$ and $K_{TDH} \pm \varepsilon$, where ε can be taken to be approximately the $RMSE(T_{DH})$ and $RMSE(T_{DL})$ respectively. This helps to ensure that the delay constraints are met when the true values are actually computed. Six inputs generated using the optimization routine and different constraint values, three for $C_{load} = -0.25$ and three for $C_{load} = 0.0$, were run through the simulation model for confirmation. The results from the confirmation points give circuits that meet all the constraints and $V_{\max} = 0.585$ for $C_{load} = -0.25$ and $V_{\max} = 0.875$ for $C_{load} = 0.0$. We also were able to determine that there was no circuit that met all the constraints for $C_{load} = 0.25$.

Stage 3

After Stage 2 a new question was posed, if all the constraints except the delays were eliminated how would the factor values for the optimal circuit from Stage 2 change? One of the reasons that the region for Stage 2 was chosen, given the similarity of the behavior of T_1, P_1, and N_3, was the constraints on the response variables forced us to stay away from large negative values of T_1. This led us to switch the range for T_1 and N_3 for the experimental region of this stage. Since T_1 could be set further to the left than the constraints previously allowed we could expect lower noise values at least for V_{STDL}. However T_2 increased T_{DH} much more than N_2 or P_4 so we could not try a similar swap with these variables. Also, from Stage 1 it appeared that when T_2 is very small it caused T_{DL} to increase as well. This additional source of increasing T_{DL} could not be countered effectively by other input variables. So only V_{STDL} is influenced by eliminating the constraints.

Since this region had also been under consideration for the search for an optimal circuit whether constraints were included or not, we ran the optimizer under both conditions. When searching for the optimal circuit with constraints the objective function included many of the constraints that had been eliminated by the choice of the region in Stage 2. The constraints and targets for this stage include: T_R, T_F, I_C, I_S as well as the delays and the three noises V_{STDL}, V_{STDH}, V_{CTDH}; again V_{CTDL} is not a factor in determining the maximum noise level.

Response Variable	Influential Input Variables	CV ERMSE
T_F	T_1, C_{load}	0.07
T_R	T_2, C_{load}, N_2	0.02
I_S	$T_1, P_1, C_{load}, N_1, P_2$	1.39
I_C	$T_1, P_2, T_2, P_1, C_{load}, N_2$	2.78
T_{DL}	T_1, C_{load}, P_1	0.09
T_{DH}	$P_4, C_{load}, N_2, T_2, P_2$	0.15
V_{CTDH}	$N_2, P_4, T_2, C_{load}, T_1$	0.02
V_{STDH}	$N_1, T_1, T_2, P_1, P_4, P_3, N_3, C_{load}$	0.05
V_{STDL}	$P_1, C_{load}, N_3, T_2, T_1$	0.06

Table 6.7 Influential Variables in Stage 3 in order of importance.

The results for this stage are summarized in Table 6.7. There is little difference in the estimates of the optimization with or without constraints. Upon confirmation we find that given $C_{load} = -0.25$ the circuit with constraints does slightly better, 0.5V, than does the circuit found in Stage 2. The circuit found for the optimization without constraints was not predicted well and did not give as good results as the circuit found with constraints. However, by looking at the main effects plots it is apparent that we could have expected only slight improvements since we were not able to influence the other noise variables by the change in the location of the search.

The circuit had previously been "optimized" using standard industry methods. Our solutions for $C_{load} = -0.25$ and $C_{load} = 0.0$ were as good or slightly better than those previously discovered. The strategy used by industry to determine their solutions required 3000 simulator runs compared to the 225 runs used by the sequential strategy in this example.

6.5 Discussion

6.5.1 Polynomial Models

As noted in Section 6.2, various methods of approximating the performance functions could be substituted in a modular way. To illustrate some of the disadvantages of using low order polynomials, a fairly standard choice for modeling,

we repeat parts of the voltage-shifter example in Section 6.3. At the first stage, a full quadratic model in 14 dimensions would require 120 points, considerably more than is necessary for our predictor. To avoid a comparison based on a different design and different data, a regression model is fit using the same data by selecting terms from a full quadratic model. For each performance function our selection process finds the same active variables as we found in Steps 3 and 4 and a quadratic model is fit to these variables.

The prediction errors are 1.5 to 3 times those from model (6.2.2). Moreover, the constrained optimization (6.4.2) leads to a different part of the space when using these less accurate quadratic regressions. A confirmation run at the regression optimum has large ripple and poor bandwidth-regression leads to a wrong part of the space.

At the second stage, regression models (fitted from the 50-point data set in the "correct" subregion) fare somewhat better. The prediction errors are now from 1.3 to 2 times those from model (6.2.2), and the optimization leads to essentially the same optimal point as model (6.2.2). This is not surprising. When factor ranges narrow low order polynomials become more competitive predictors.

Increasing the amount of data will not in itself guarantee substantially improved regression models. Reduction of systematic error from these models may also require including more model terms (e.g., third-order terms). As the models become larger, though, they will become computationally more demanding because of the need to do model selection.

6.5.2 Taguchi Approach

A traditional approach to resolving multiple performances is to combine them into a single objective function.[1] Yield is a common choice.[2 4] In contrast to yield, Taguchi[5] emphasizes reduction of variability in the performances of interest to minimize measures of economic loss. The resulting choice of designable factors is intended to be robust, that is, to make the circuit insensitive to manufacturing and other uncontrollable variabilities. Although yield is a convenient single criterion, one concern leading to Taguchi's view is the fact that the performance constraints defining yield are often arbitrary and create impractical differences between circuits that just pass and those that just fail.

In the voltage-shifter example we combined performances and criteria into a single objective and also followed Taguchi's route by incorporating reduction of variability as one of the criteria. However, by modeling performances, the user can substitute these models into tailor-made objective functions at the optimization stage. If desired, these models could also be used to predict, and hence optimize, yield.

Though our approach has the same overall objectives as the Taguchi approach, there are a number of differences between the strategy in Steps 1 to 6 of Section 6.2 and the *methods* advanced by Taguchi. Taguchi seeks to model criteria or losses. As noted,[3] this can not only lead to inefficient use of the data, but also to a poor engineering design. Furthermore, the Taguchi methods have difficulty reconciling multiple criteria in the optimization. An analysis of the voltage-shifter example based on Taguchi's signal-to-noise ratios, using an experimental plan of 729 simulator runs, had to be abandoned because of difficulty in reconciling the signal-to-noise ratios. For a more complete discussion of the different strategies for robust engineering design see Chapter 7.

6.5.3 Region Reduction

Region reduction can be approached in a sequential manner for the multiple criteria optimization problem. There are a few ad hoc rules to remember during region reduction.

1. Constraint response variables that are only influenced by one or two input variables should be handled first.
2. If $g_i(\mathbf{Y})$ is part of the objective function, $\hat{g}_i(\mathbf{Y})$ should be used if the prediction error for this model is not much worse than the error for $g_i(\hat{\mathbf{Y}})$.
3. The information gained from modeling the responses can be used to help cluster inputs and the responses they influence into fairly disjoint groups. Then region reduction is a series of "independent" region reduction problems rather than one large one. The variables that form these disjoint sets should be handled next.
4. Input variables that have a minimal effect on the target responses should be used to make sure the new region is as far inside the constraint region as possible.
5. The ranges for input variables that most influence target response variables should be adjusted last.

The decomposition of the optimization problem into clusters may also be useful to reduce the cost of numerical optimization. More work needs to be done to automate this procedure.

Another benefit of using statistical models with numerical optimization algorithms is brought out in the example. Often investigators will be interested in finding a solution for a fixed value of one or more of the input variables. They may also want to look at several different settings for these inputs. The use of statistical models gives more flexibility to the investigator in experimenting with the selection of input values.

This example showed several advantages in using a sequential strategy to find an optimal circuit over single stage methods. The sequential method allows the investigator to develop successively more accurate predictors using many fewer simulation runs than a single stage method would to get a predictor of equivalent accuracy. For example, if one reduces the range of each of 20 input variable by half, the area of the new range for the circuit described is 0.005% of the initial range. One would need to take a huge number of observations in a single stage method to produce a design with a similar density of simulation runs and hence similar accuracy of the predictor. Also, by reducing the experimental space there is a strong possibility of simplifying the performance function used during optimization. This was seen in our example by the elimination of various constraints from the performance function at the end of stage 1. In a one stage method this reduction would not be possible.

6.5.4 Latin Hypercube Sampling

We, and other teams, have found Latin hypercubes useful for simulation experiments. Instead of complete randomization of the Latin hypercube columns with respect to each other, we now use a method that controls the pairwise correlations between factors. For the examples of this article, near-orthogonality of the columns is probably desirable. Alternatively, if two uncontrollable variations are correlated, this could also be accommodated. For more details on the properties of Latin Hypercube Sampling see Chapter 4.

The use of LHS designs led to some unsuspected benefits in the digital logic circuit example. Because LHS designs use many levels for each input variable, drawing conclusions for simple response functions V_{DN}, V_{UP}, I_{OL}, and I_{OH} was straightforward. No modeling was required; plotting some simple graphs,

which should be plotted as a matter of course, gave enough information to come to the necessary decisions. Even the decisions for T_F and T_R could be reached by looking at some simple contour plots. This is because LHS when projected to lower dimensions is still a LHS. Even though the general relationships between inputs and the responses were known in advance and are relatively simple, without using LHS it would not have been possible to reach the appropriate conclusions as quickly, if at all. If the true relationships are not known even in these simple responses it could become difficult to model them with the two or three input levels typically used in most experimental plans.

6.5.5 Plots of Factor Effects

Our sequential strategy is assisted by graphical display of the effects of individual factors (Step 4). More automation in moving to the next subregion is desirable, and we are working on this problem, but engineers nonetheless often find these plots revealing. The information is easily displayed and little experience is required to interpret them.

6.5.6 Computation Time

When the stochastic-process model (6.2.2) is used, computing the maximum likelihood estimates of the correlation parameters is often the biggest computational expense. In the voltage-shifter example, the total time for Steps 3 and 4 (fit, check, and plot) for all nine performances is 6 hours on a SUN 3/80 at Stage 1. For comparison, generating the 75 simulator runs takes about 2 hours, and the optimization requires about 30 minutes.

In our experience these times are quite reasonable for moderately complicated problems like that of the voltage shifter, which was chosen because previous attempts at analysis had not been entirely successful. Fitting polynomials, usually a cheaper alternative, requires more data to achieve commensurate accuracy. More data obviously increases simulation time and, if used to fit larger polynomial models, model fitting and selection may become burdensome.

The model fitting strategies in Steps 1 and 3 have been used by other workers (A. Owen, J. Koehler, and S. Sharifazadeh at Stanford) for modeling device simulators (e.g., PISCES, SUPREM). Often, generating data is much more expensive for such simulators than for circuit simulation. With expensive data, the greater efficiency of the stochastic-process model (6.2.2) becomes even more advantageous.

6.6 Conclusion

We have given a strategy for finding optimum circuit designs with comparatively few circuit simulator runs. This strategy uses sequential experimentation and statistical modeling of performance functions to accurately approximate the simulator over successively smaller sub-regions on which to search for the optimum. We use models that treat a performance function as the realization of a stochastic process. These models are more data adaptive and flexible than polynomials, hence better accuracy typically follows. Many factors, performance functions and criteria can be treated using this strategy. The features of this strategy and its advantages are exemplified in two instances.

6.7 References

1. Brayton, R. K., Hachtel, and Sangiovanni-Vincentelli, A. L., (1981) "A Survey of Optimization Techniques for Integrated-Circuit Design," *Proceedings of the IEEE*, 69, 1334-1362.

2. Low, K. K. and Director, S. W., (1988) "An Efficient Macromodeling Approach for Statistical IC Process Design," in *IEEE Int. Conf. on Computer-Aided Design*, pp. 16-19, Santa Clara, CA.

3. Welch, W. J., Yu, Tat-Kuan, Kang, S. M., and Sacks, J., (1990) "Computer Experiments for Quality Control by Parameter Design," *Journal of Quality Technology*, 22, 15-22.

4. Yu, T. K., Kang, S. M., Sacks, J., and Welch, W. J., (1991) "Parametric Yield Optimization of CMOS Analogue Circuits by Quadratic Statistical Circuit Performance Models," *International Journal of Circuit Theory and Applications*, 19, 579-592.

5. Taguchi, G., (1986) *Introduction to Quality Engineering,* Asian Productivity Organization, Tokyo.

6. Box, G. E. P. and N. R. Draper, (1969) *Evolutionary Operation,* Wiley, New York.

7. Sacks, J., Schiller, S. B., and Welch, W. J., (1989) "Design for Computer Experiments," *Technometrics*, 31, 41-47.

8. Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., (1989) "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409-435.

9. Dehnad, K., (1989) *Quality Control, Robust Design and the Taguchi Method*, Wadsworth.

10. Phadke, M. S., (1989) *Quality Engineering Using Robust Design*, Prentice Hall, New Jersey.

- Chapter 7 -

Discussion of Statistical Methods of RED

7.1 Introduction

It is useful to separate the general strategy and philosophy of RED, where there is widespread agreement, from the tools and methods that have evolved to implement this philosophy. All the methods are concerned with the same subjects: model fitting, interactions, loss functions and optimization, noise factor distribution and the efficiency with which RED can be carried out. It is useful to discuss these topics separately rather than within the different methods. By subdividing the RED problem into fairly independent components it may be easier to build a consensus strategy. This may eliminate some of the confusion that is inevitable when there are several methods available for those attempting to apply RED in industry. The rest of the sections in this chapter discusses these subjects and other RED related topics.

7.2 Estimation and Noise Parameter Distribution

When computing the mean and variance of Y the expectation is really over the noise factors, $\mathbf{U}$, not Y since Ω is the true sample space. To obtain estimates of the mean and variance of Y the distribution of $\mathbf{U}$ needs to be defined or approximated in some way. One of the fundamental differences between the LM approach and the RM approach is how assumptions are made about the distribution of $\mathbf{U}$. The other fundamental difference in the two approaches is how they estimate $E(Y)$ and predict estimates of $E(Y)$ at new settings of the design factors.

The methods for estimating the mean and variance of the response in the LM approach and the RM approach are almost the reverse of each other. The RM approach uses the predictor $\hat{y}(\mathbf{c},\mathbf{U}) = \hat{y}(\mathbf{X})$ to estimate Y and

$$E_U(\hat{y}(\mathbf{c},\mathbf{U})) = \int_{\Omega_N} \hat{y}(\mathbf{c},\mathbf{U}) d\mathbf{U}$$

is used to estimate $E_U(Y)$. $E_U(\hat{y}(\mathbf{X}))$ in turn can be estimated by

$$\hat{E}_{\mathbf{U}}(\hat{y}(\mathbf{c},\mathbf{U})) = \sum_{\omega_N} \hat{y}(\mathbf{c},\mathbf{U}) d\mathbf{U},$$

where ω_N is a computer generated random sample from the same distribution as $\mathbf{U}$. The LM approach estimates $E(Y(s_D)) = \eta(\mathbf{s}_D)$ and $Var_U(Y(s_D)) = \sigma^2(\mathbf{s}_D)$ by

$$\hat{\eta}(\mathbf{s}_D) = \sum_{\mathbf{s}_i \in \mathbf{s}_N} y_i(\mathbf{s}_D)$$

and

$$\hat{\sigma}^2(\mathbf{s}_D) = \sum_{s_i \in S_N} (y_i(\mathbf{s}_D) - \hat{\eta}(\mathbf{s}_D))^2$$

respectively the mean and variance of Y over the outer array, S_N, at each design point, $\mathbf{s}_D$, of the inner array. The LM approach can then use the estimates $\hat{\eta}(\mathbf{s}_D)$ and $\hat{\sigma}^2(\mathbf{s}_D)$ to minimize variance and adjust to target or the estimates can be treated as response variables and modeled themselves, let the models be designated $\hat{\hat{\eta}}(\mathbf{c})$ and $\hat{\hat{\sigma}}^2(\mathbf{c})$ respectively.

There are two sets of assumptions needed for the estimation of $E_{\mathbf{U}}(Y)$ and $Var_{\mathbf{U}}(Y)$, the model fitting assumptions and the distribution assumptions of $\mathbf{U}$. For the RM approach all the input factors are treated like design factors, i.e. none of the input factors are random variables, when modeling $\hat{y}(\mathbf{X})$. The distribution of $\mathbf{U}$ is only needed when estimating the risk function. The LM approach models the mean over $\mathbf{U}$, so $\hat{\eta}(\mathbf{c})$ and $\hat{\sigma}(\mathbf{c})$ are functions of only the design factors. However, model *and* distribution assumptions about $\mathbf{U}$ are involved when estimating $\hat{\eta}(\mathbf{c})$ and $\hat{\sigma}(\mathbf{c})$ and so are involved in the estimation of $\hat{\hat{\eta}}(\mathbf{c})$ and $\hat{\hat{\sigma}}^2(\mathbf{c})$.

7.2.1 Distribution of Noise Parameter

Since the RM approach has no "replication" at the design factors it is impossible to get estimates of $E_{\mathbf{U}}(Y)$ and $Var_{\mathbf{U}}(Y)$ without explicitly stating the distribution of $\mathbf{U}$. This allows for a lot of flexibility in the choice of distributions, but also requires a degree of knowledge (or confidence) by the designer to write down a distribution. Common distributional assumptions are that $\mathbf{U}$ has a symmetric distribution with mean zero and variance σ^2. It is also common to assume that $\mathbf{U}$ is normally distributed. One benefit of the RM approach is that changes in the assumed distribution can be made and new estimates of $E(Y)$ and $Var(Y)$ using a new distribution can be computed just by generating a random sample of $\hat{Y}$ using the new distribution. No new observations need to be taken

from the simulation model. These random samples should be inexpensive to generate since the predictor is inexpensive to run.

As described in Chapter 1, the LM approach estimates the mean and variance by using the outer array to get sample means and variances. The settings of the noise factors in the outer array are fixed at given experimental design values. In this situation, the *proxy* distribution of **U** is a discrete distribution with probability $1/n_N$ for each run in the outer array. The outer array could also be thought of as a set of representative points of the distribution of the noise factors. The nature of the distribution, assuming that the points in the outer array are truly representative, will depend on the experimental design of the outer array. Many of the assumptions that are stated explicitly in the RM approach, plus some that are not, are made implicitly when creating the outer array. Hence, the designer is not eliminating the decisions about these assumptions, they are just being made indirectly.

A common choice of experimental design for the outer array is a two-level orthogonal array which assumes that there are a minimal number of interactions between noise factors. If the points of this design are to be "representative" of the distribution of **U**, the noise factors must be independent and the distribution be symmetric with mean zero and variance a function of the range of the noise factors. The latter assumptions were made in the response model approach, while the assumption of independence was not. These assumptions can not be changed without changing the design of the outer array or changing to the RM approach style of analysis.

7.2.2 Model Fitting Assumptions

The design factors are the usual focal point of model fitting for both the RM and LM approach. However, the noise factors also have a role to play in fitting the response for *both* approaches. In the RM approach the design and noise factors are treated in the same manner when building the model.

Since the RM approach uses $\hat{y}(\mathbf{X})$ to get estimates of the mean and variance, only distribution assumptions enter into the computation of $\hat{E}_{\mathbf{U}}\hat{y}(\mathbf{c})$. This creates a nice division between modeling and distributional assumptions. Given the distribution of **U**, errors in the model will effect the estimation of the mean and variance. Model errors could occur due to bad assumptions for either the design factors and the noise factors, but are restricted to the estimation of $\hat{y}(\mathbf{X})$.

The LM approach assumptions about model building are split into two parts, the design factors are used to model $\hat{\eta}(\mathbf{c})$ and *model* assumptions for the noise factors are also implied when choosing the outer array for *estimating* $\hat{\eta}(\mathbf{c})$.

When a two level experimental design is used for the outer array the requirement that these points be "representative" of the distribution of $\mathbf{U}$ forces some implicit assumptions about the effect of $\mathbf{U}$ on Y. Since it is only a two-level design it is necessary to assume the effect of $\mathbf{U}$ on Y is linear. So even though the LM approach appears to more "data-driven" than the "model-driven" RM approach this is not the case. Unlike the LM approach, the RM approach does not need experimental designs with small numbers of levels which gives the RM approach greater flexibility in choosing designs and models.

It is also worthwhile to note that

$$E(f(\mathbf{c},\mathbf{U})) \neq E(f(\mathbf{c},\mu_U)),$$

where $\mu_U = E(\mathbf{U})$, unless $f(\mathbf{c},\mathbf{U})$ is a first order linear function in the noise factors. The designer may be particularly prone to this error when the RSM method is used. For example, if

$$Y = \mathbf{g}'_1(\mathbf{c})\beta + \mathbf{g}'_2(\mathbf{U})\alpha + \mathbf{g}'_3(\mathbf{c},\mathbf{U})\gamma$$

and let $E(\mathbf{U}) = 0$, then $E(Y) = \mathbf{g}'(\mathbf{c})\beta$ only if $\mathbf{g}(\mathbf{U})$ is a first order linear function for all the noise factors.

The assumptions for the design factors are typical model fitting assumptions, e.g. interactions, additivity, linearity. If the assumed model does not fit then errors in prediction will occur. The stochastic process used in DACE makes fewer direct assumptions about the shape of the function, such as interactions and linearity, than the LM approach and RSM, but model assumptions need to be checked for all the methods discussed. An important reason why so many different PerMIA's are needed is that they are dependent on which model assumptions are made. There are many model checking tools available and these should be used to determine the accuracy of the model.

The design-noise factor interactions are the driving force behind RED. Without these interactions reductions in product variability would not be possible. So it is important to identify and accurately estimate the important interactions. The LM approach avoids identification decisions by making all design-noise factor interactions estimable with the negative consequence of large sample

sizes. Most of the degrees of freedom in the LM approach are used to estimate all the $D \times N$ interactions. It is unlikely that all these terms are significant. It is more efficient to choose which interactions to estimate including $D \times D$ and $N \times N$ interactions. The RM approach is flexible in choosing which interaction terms to estimate, helping to reduce sample size, but opens up the possibility of missing an important interaction. This trade-off is a fundamental difference between the two approaches.

7.2.3 Further Topics

It is important to remember why the designer is interested in the variance of the noise factor. Variance is used to estimate the range of the noise factors. If the noise factor has a normal distribution with mean zero and variance $\sigma^2 = 1/9$ then 99% of input values will be between [-1,1]. If the noise factor has a uniform distribution and $\sigma=1/3$ only 91% of input values will be in the range [-1,1]. If the range [-1,1] represents 95% of the input values given a normal distribution it will only cover 70% of values if the noise factor has a uniform distribution. These examples show that even if an accurate estimate of the variance of the noise factors is known, if the distribution is not known there can be a significant misrepresentation of the actual product variation.

In both the LM approach and RSM it is suggested that some screening experiments be carried out. Screening of design factors is useful in reducing the sample size of future experimental plans. These screening experiments frequently have only 2-level experimental plans. Both the LM approach and RSM rely on investigating one region of the input space then moving to another, possibly completely new, region in the search for optimal product factor settings. Screening results may become invalid when moving from one region to another. If the response has a complicated surface, design factors that were unimportant in the original space may be important factors in later regions. This is true especially if there has been a considerable shift in location of the input range. If these new regions are not rescreened important factors may be lost.

DACE starts by looking at a large design factor space. DACE will have a tendency to miss factors with smaller effects because the power of the experimental plan will not be strong enough to find them. Small effect factors will start to appear as the subregions become smaller so it is important not to screen out factors too readily when using this approach as well.

7.3 Loss Functions and Optimization

Finding the product factor settings that best achieve the design specifications under manufacturing and environmental variability is the goal of RED. The product response can usually be described in mathematical terms. Finding optimal solutions of a mathematical function is the goal of a large body of research. If the mathematical functions are known, and computing solutions to these functions is inexpensive, it is straightforward to use a numerical optimization algorithm to search for a solution. In RED the function is usually unknown (physical experiments) or the function is too expensive to put through a numerical optimization algorithm (computer experiments). If an accurate and cheap predictor can be found it can be used in place of the unknown or expensive function. This is the goal of the RM approach. It seems wasteful to ignore the large volume of work on optimization and not aim for the best solution as directly as possible by using these optimization techniques.

The response surface over a large range of the design and noise factors is likely to be a complicated function. Simple first or second order polynomials will often not fit the true response surface very accurately. This has led RSM to focus on a small area of the input range. The response surface is more likely to behave as a low order polynomial locally. The search for optimal solutions is then a matter of moving from one region to another until an optimal region is found. We will refer to this as small region exploration. DACE is capable of modeling more complicated surfaces than second order polynomials so the designer can attempt to model much larger regions. DACE then looks for subregions where likely solutions appear to be located. We will refer to this as large region exploration.

There are several drawbacks to using small region exploration. When using small region exploration it is necessary to have nominal settings available before trying to minimize product variability, otherwise it may not be possible to adjust to target for the range of inputs chosen. Once the nominal settings have been chosen, reducing variability only guarantees an optimal solution for that starting point. It is possible that another choice of initial nominal settings may lead to a better optimal solution. Also, optimization algorithms are hampered when using small region exploration because the algorithms can not search for global optima over the full input space.

To get a reasonable picture of the response surface with large region exploration it is important to have a fairly large experiment at the first stage, possibly larger than the first stage of RSM but still considerably less then the LM approach. The goal of the first stage is to be able to find general trends and relationships between variables. It will give enough information to choose a subregion where a new, smaller experiment will give a more accurate picture of the response surface. It is true that looking over a large design factor range will initially include a lot of space that does not remotely meet the response requirements. Some of this is due to taking observations on a rectangular space and even though the information may be "useless" for the problem at hand it could be valuable later. It also allows the designer to locate *all* feasible regions. This gives more flexibility in choosing the best solution possible. The focusing aspect of large region exploration gives designer a solid endpoint to the search. Once an accurate predictor has been established there is no need for further experimentation.

The performance measure in engineering design problems may be very complicated. Frequently the performance measure will involve many responses, i.e. multi-objective criteria. Besides the usual task of reducing variability some responses may need to satisfy some constraint. Sometimes specifications require a function of several responses to be constrained in some way. Trying to develop portmanteau performance measures, such as SN ratios and PerMIAs, to handle all situations is a daunting task. A performance measure can handle a much larger range of design specification problems other than just reducing variability when using predictors of the response. The task of developing performance measures is not difficult with the RM approach. The performance measure typically will closely mirror the given engineering design specifications. Performance measures based on real engineering objectives can be used with confidence since the predictor emulates the true behavior of the process under study. Multi-objective optimization already familiar in engineering design studies to find nominal settings would require only a small adjustment to include robustness requirements.

The use of a numerical optimization algorithm is simplified if the terms in the performance measure that reduce variability are mean squared error rather than variance. This expectation can be written as the sum of two terms, $(E(\hat{y}(x)) - T)^2$, or bias, and $Var(\hat{y})$. Bias will dominate this sum when starting

the numerical optimization from a random set of values for the input factors. This could lead to finding solutions that are only local optima. However, the nature of region selection in DACE provides a check against this occurring. When selecting a new region for investigation the designer can use main effect and interaction plots to identify dispersion factors and regions where variability is reduced. This helps to ensure the new region contains a solution that gives a real reduction in variability.

7.4 Sample Size

Computer simulation models typically have large numbers of variables available for investigation. This is because the functions are complicated and all the factors that may affect the output are known and readily available for experimentation. For computer experiments the product array becomes very costly for even modestly sized problems by computer experiment standards. For example, a simulation code with 20 control factors and 10 noise factors will require over 800 runs of the simulation code. If each run takes one minute of CPU then the full experiment will take over 13 hours of CPU time. The combined array reduces the size of the experimental plan, it will need only a small fraction of this number of runs for an identically sized problem. Also, remember that any screening experiments that are used to help reduce the size of the product array should be included when discussing the costs of factor design.

A single experiment using DACE typically will not yield an immediate solution. Further experimentation will be necessary to explore the chosen subregions. The cost of this sequential experimentation is a linear function. The example given in the first paragraph of this section would allow 5 experiments of 150 runs each before being equal to the number of runs in one product array experiment. Examples from this thesis and Bernardo, *et al.*[1] show that for fewer runs than in one step of the LM approach DACE will find a globally optimal solution. Since the RSM approach uses combined arrays the size of experimental plans will be similar to DACE. The difference in sampling costs between DACE and RSM will depend more on the number of experiments carried out in the sequential search for a solution. DACE is likely to need fewer experiments than the RSM approach because DACE is able to accurately model much larger regions than RSM.

7.5 Ease of Use

An important reason why Taguchi's methods have become so widespread is the ease with which they can be learned and incorporated into real engineering design problems. Recent work on LM approaches have done much to improve on Taguchi's initial method. However, the improvements have created new complications or pointed to overlooked ones. The designer now must be concerned about transformations, interactions, design selection, and selection of the performance measure. When selecting the performance measure the designer must either select the correct SN ratio or PerMIA if using "traditional" methods or model both the mean and variance. For multi-variate response RED problems this can quickly become cumbersome as it doubles the number of models that need to be estimated. After factor settings minimizing product variability have been found, location factors need to be adjusted so the response is on target. Decisions then need to be made whether a new experiment should be conducted to make further improvements. Many of these concerns apply to RSM as well.

DACE helps to eliminate some of these choices and works toward finding an optimal solution. By looking at each stage individually it is apparent how the number of decisions that the designer needs to make can be reduced.

1. Choose Design. The use of LHS in computer experiments makes design selection easy, no concern about interactions and aliasing. Only a decision about sample size is necessary and general guidelines are already available.
2. Analyze Results. The use of the stochastic process eliminates the need to decide on the functional form to be estimated as well as expanding the class of functions that can be modeled.
3. Plot Analysis and Optimize. The plots help to identify important factors and the nature of their effect on the responses. The plots, combined with optimization results, help to locate solution regions. The optimization can take advantage of prepackaged optimization code so the designer only needs to develop the performance measure.
4. If the model is not accurate enough, choose the next subregion. Once a likely area to look in is selected, choosing the next subregion is not difficult. For example, reducing the ranges of 10 out of 20 factors by half reduces the size of the new region to .001 of the previous region. This is one reason why the stochastic process can produce accurate models after

only a few stages. This allows the designer to be conservative in selecting the subregion and reduces the chance of missing the "correct" region.

4a. If satisfied with the model, confirm solution from 3. Confirmation is important for any statistical approach to RED. Once the results of DACE have been confirmed the designer can then go on to search other promising regions that may have been discovered from the initial stages of experimentation. When these regions have been investigated the designer can be confident that the best possible factor settings for that engineering design have been found.

The first three steps can be easily automated. The designer only needs to be concerned with understanding the effects plots and writing routines to calculate the performance statistic for the optimization code. Step 4 is more difficult to automate because for large design problems there may be several subregions worth investigating and choosing the size of the next region is a fairly subjective process. At this point engineering principles as well as model information may be applied to help with selecting the next subregion. Choosing when to stop building new subregions is also a subjective decision depending on the model accuracy needed by the designer.

7.6 System and Tolerance Design

Robust engineering design has been split into three major phases, system design, parameter design and tolerance design. This categorization is useful but it has also led to a division in the development of methods in RED. The work on methods for parameter design is a good example. The LM approach and RSM typically assume that some nominal setting of product factors is available, i.e. that system design has been completed. Once factor settings have been chosen to increase product robustness the task of tolerance design is left uncompleted.

Separating system design from parameter design has led most research on methods for parameter design to assume that some nominal setting of the control factors is available as a starting point. However, a significant part of the cost in engineering design is the time it takes to find initial settings. It would be useful to eliminate the need of finding an initial nominal setting, but modeling the variance and the mean separately makes this difficult. When minimizing variability without any concern for the mean there is a strong chance that adjustment to

target will be impossible without affecting product variability. The LM approach and RSM have avoided this difficulty by using the nominal settings as a base from which to start.

DACE used in computer experiments can easily combine parameter design with the part of system design that involves finding the nominal settings. The designer typically knows the factor settings which meet engineering design specifications will fall within some broad region. Instead of spending time finding the nominal settings, the designer can perform an experiment using these expanded ranges and generate estimates of the responses. The designer can use these predictors to search for regions where engineering design specifications are met. This gives the designer several advantages. The first advantage is that the last stages of system design may be skipped. Second, it allows systematic search in a much larger region to locate all feasible designs. This could lead to a design selection that may have been overlooked. Third, it gives the designer an early glimpse at how the product factors behave in changing the response.

Tolerance design can be incorporated into either the RM or the LM approach. The available factor plots can be used to determine the spread of the response due to a specific noise factor. If the noise factor has a small slope this would be a candidate for loosening the specifications. Alternatively, the discovery that a noise factor has a large influence on the response could lead to a tightening of the tolerance for that noise factor. This information is not readily available when using the LM approach because it is bound up in the performance measure. The advantage of the RM approach is that the noise factor distribution can be changed and the effects tested without further experimentation by using the predictor to estimate what would occur under the new noise distribution.

7.7 Physical RED vs. Computer RED

The ideas in DACE discussed in this thesis were developed from work on computer-aided design problems. Many of the ideas are also applicable to physical RED problems. There are a few points where difficulties may arise. These areas need to be investigated to see what changes might be made for use in physical RED problems. One aspect of physical RED problems that is not of concern in CAD/CAE RED problems is experimental error. Experimental error is the result of uncontrolled noise factors, noise factors whose variability is

measured by replication and are not part of Taguchi's outer array, and measurement error. The ideas of combined arrays and modeling the response are still valid for controllable noise factors. When experimental error exists it needs to be measured and cannot be assumed to be constant over the range of inputs. Replication will be necessary and a measure of variability will need to be taken at each point in the combined array. This measure will need to be modeled and incorporated into the performance measure. These are difficulties that affect all three strategies discussed.

In computer experiments changing the values of the input factors is a straightforward task. In physical experiments this is frequently not the case. It is unlikely that LHS can be used in its present form for physical experiments. The number of different settings can be reduced depending on the ease with which the factor setting can be altered in the physical experiment. An arrangement of run order could be found to reduce the number of changes. Work of a similar nature has already been done on orthogonal arrays,[2] and may be applicable to these adjusted Latin Hypercube Samples.

There is no theoretical reason why the stochastic process models used in DACE cannot be used in physical experiments. One way to incorporate experimental error is to use a covariance structure of the form $\sigma^2(R + \gamma I)$, where I is the identity matrix and $\sigma^2\gamma$ is a measure of experimental error. Further work is being carried out to see how effective this model is in physical experiments.

7.8 Conclusion

The LM approach is a popular and successful strategy in RED but many statistical weaknesses have become apparent. The LM approach builds simple linear models and makes assumptions about the noise factor distribution which are hard to assimilate with reality. The RM approaches build more complicated models but the assumptions involved with these models also occur, sometimes implicitly, in the LM approach. Checking model assumptions is equally important for the LM and RM approaches, but the extra flexibility in choosing experimental design in the RM approach gives the RM approach an advantage in checking the assumptions.

The RM approaches have more flexibility in the choice of noise factor distribution. This allows the designer to choose noise factor distributions which mimic real variability and not one that is forced upon the designer by the choice

of design when using the outer array of LM approach. Separating the distribution assumptions of the noise factors from the experimental design allows experimentation with different distributions without the need for further observations from the simulation model.

The use of a stochastic process as a predictor helps to reduce model error by interpolation through the design points. This works well with experimentation on computer simulation models. DACE has no hidden costs or hidden assumptions so the statistical difficulties can be approached directly. When using the stochastic process of DACE many of the modeling difficulties that arise when using linear models can be avoided, such as confounding, nonlinearity and additivity. DACE gives the designer a straightforward and efficient way to solve engineering design problems. It also leads directly to optimal solutions so more time can be spent by the designer developing entirely new engineering designs rather than making adjustments to old designs.

7.9 References

1. Bernardo, M. C., Buck, R., Liu, L., Nazaret, W. A., Sacks, J., and Welch, W. J., (1992) "Integrated Circuit Design Optimization Using a Sequential Strategy," *IEEE Transactions in Computer-Aided Design*, 11, 361-372.

2. Cheng, C.-S., (1985) "Run Order of Factorial Designs," in *Proceedings of Berkley Conference in Honor of Jerzy Neyman and Jack Kiefer*, vol. 2, pp. 619-633, Wadsworth Advanced Books and Software, Hayward, CA.

- Chapter 8 -

Conclusions and Further Research

The investigation of methods for robust engineering design has led to inquiries into a number of statistical topics as well as others, such as optimization. Two areas that were investigated more thoroughly because of their connection to computer experiments was spatial modeling and experimental design. Also, to help the development of robust engineering design methods a thorough investigation of the underlying probability model was necessary to identify weak points in the different methods outlined. The conclusions from this thesis can be roughly broken into spatial models, Latin hypercube sampling and robust engineering design.

Spatial Models

The spatial models described in Chapter 2 were used to predict the response of several different circuit simulation models. The predictions were accurate enough so that the predictor could be used to find optimal robust engineering designs. These examples, as well as others mentioned in the literature, are evidence that the predictor, particularly with the covariance function (2.3.1), can be used successfully to estimate computer simulation models. The examples also extend the used of spatial statistical models to higher dimensions than usually found in the literature.

The work in this thesis resulted in a software package for the design and analysis of computer experiments. The software is still fairly crude, but complete. There are still improvements that can be made in the efficiency of parts of the code, most notably in computing cross-validation results. While the research described in Chapter 3 led to maximum likelihood estimation methods for the model parameters at a reasonable cost in high dimension problems, work still can be done to improve the cost effectiveness of estimating the parameters for the spatial models described.

The study of spatial statistics is a fast growing field with many opportunities for further research. The covariance function used in this thesis has not been investigated thoroughly on a theoretical basis. Further study of the robustness of the parameters in the covariance function from both the prediction and MLE

perspective would be useful. An investigation of a variety of known functions would be useful to determine the range of functions that may be well estimated, e.g. the limits to roughness of a function or the effect of nonstationarity. The spatial model described in Chapter 2 could also be applied to problems with measurement error.

The theoretical basis for the ONETIME algorithm used to compute maximum likelihood estimates is not well developed. There are many avenues to explore in understanding why and when this algorithm works successfully. An important topic that is currently receiving some attention is the modality of the likelihood function. This is an intriguing area where our models do not quite fall into the families described in previous work. Further understanding of the parameter values and what they mean could also be helpful to gain a better intuitive feel to why the ONETIME algorithm works as well as it does.

Latin Hypercube Sampling

Latin hypercube sampling was developed for designing experiments for high dimensional computer experiments. Latin hypercube sampling is fast and simple to implement and previous work has shown that Latin hypercube sampling is asymptotically more efficient than simple random sampling or stratified random sampling and estimates from Latin hypercube sampling are asymptotically normally distributed.

Latin hypercube sampling together with the spatial models used for the research in this thesis appear to form a good basis for the design and analysis of computer experiments. This conclusion is based mainly on the nature of computer experiments, but asymptotic results from Chapter 4 show that $MSE(\hat{Y}) \rightarrow 0$ as $n \rightarrow \infty$ for Latin hypercube sampling, but not for simple random sampling or many deterministic designs when an interpolator is used for the statistical predictor.

Variability in the spatial location of design points in random designs is used as a measure of discrepency and comparisons were made between Latin hypercube sampling and simple random sampling. This measure of discrepency showed that Latin hypercube sampling is better, on average, in evenly spacing points throughout the input space.

The investigation of the space filling properties of Latin hypercube sampling is still at the initial stages. Further comparisons of the new discrepency

function with more experimental designs, in particular stratified random sampling, need to be examined. Also, values for other discrepency functions should be computed for Latin hypercube sampling for comparison to nonrandom designs.

Robust Engineering Design

Taguchi developed a simple experimental method that helped to make product performance robust to many types of variability, thus improving quality. The methods he used to find the product settings, while solid in principle, are not particularly efficient and have many theoretical problems from a statistical point of view. Examination of the underlying model in the RED problem show that the statistical assumptions about the type of function estimated and noise factor distribution are used for both the LM and RM approach, but the RM approach allows for more flexibility in choosing what assumptions are made. In particular, the RM approach has more flexibility in the selection of designs and statistical models available for estimation and prediction and more flexibility and opportunity for experimentation with different distributions for the noise factors.

DACE predicts the complicated response surface of simulation models better than regression polynomials used in RSM. DACE also can use a different approach to optimization. DACE can model a larger input space and still produce a reasonable accurate predictor, hence DACE does not need an initial set of nominal values before starting the optimization search. Investigation into using DACE on physical systems and manufacturing processes needs to be carried out to check what adjustments may be necessary to the procedures used in DACE. Also, optimization algorithms play a more important role when using DACE and further investigation is needed on choosing algorithms for these multivariate optimization problems.

The general strategy for experimentation in RED has largely been settled, however there are many statistical questions still worthy of further investigation. One area of study in RED that has largely been ignored is the estimation of the noise factor distribution. This would involve applying much of the work on density estimation and further research into estimating covariance matrices and multivariate density functions. This is an important area to pursue from an application point of view and has many opportunities in statistical research.

- Appendix 1 -

DACE Code and User's Guide

A1.1 User's Guide

The DACE software performs several tasks:

1. Computes parameter estimates of stochastic process using likelihood function in two ways:
   a. Full MLE, i.e. all parameters are free to vary.
   b. ONETIME algorithm, see Section 3.5.
2. Computes likelihood values (3.1.1) for any number of given parameter estimates.
3. Predicts response given parameter estimates, via (2.2.3).
4. Computes cross-validation for parameter estimates, via (3.4.1).
5. Computes values for "main effects" plots, see Section 2.4.

A1.1.1 Source and Header Files

There are 10 source files and 10 header files necessary to compile the DACE software. The source files end with the suffix .c and the header files end with the suffix .h and the file names will be written in bold type. The prefixes for the 10 pairs of source and header files are: **dace**,**rb_alloc**, **rb_covm**, **rb_effects**,**rb_like**, **rb_math**, **rb_matman**, **rb_opt**, **rb_predict**, and **rb_print**. The main function is in **dace** and contains the structure for choosing the user requested tasks. The main tasks for the rest of the source files are as follows:

1. **rb_alloc** - memory allocation routines.
2. **rb_covm** - various covariance computation functions.
3. **rb_effects** - functions for computing main effects.
4. **rb_like** - functions for computing likelihoods.
5. **rb_math** - miscellaneous mathematical functions.
6. **rb_matman** - some matrix manipulation functions.
7. **rb_opt** - functions involved with optimization.

8. **rb_predict** - functions for computing predictions.

9. **rb_print** - various input/output routines.

A1.1.2 Variable Definitions

All variables will be capitalized in this document for clarity. In the source code only global variables are capitalized. There are 20 global variables used in the DACE software they are identified below as part of their definition. All variables except a few obvious character strings are either integer or double scalars, vectors or matrices. Variable descriptions follow C programming language notation.

1. N - Number of runs in experiment. Number of rows in **S. Input by user. Global Variable.
2. NX - Number of input variables. Number of columns in **S. Input by user. Global Variable.
3. P - Number of parameters in linear model part. Typically P=1, a constant linear model. Input by user. Global Variable.
4. SEED - Seed for random number generator used for starting MLE or generating random inputs for prediction. Input by user.
5. NFILES - Number of data files where input variables and responses are stored. At this time looks for either all data in one file or input variables in one file and response variables in another file. Input by user.
6. NY - Number of response variables in input files. Input by user.
7. SELY - Response variable that user wants to analyze. Input by user.
8. **S - Input variables (columns) experimental run (rows) values. Input by user.
9. **TEMPY - Matrix with all response variables (columns) for respective experimental runs (rows). Input by user.
10. *KIN1,*KIN2 - Index vectors containing subscripts of variables used to compute linear model terms from **S. Some examples,
    a. if KIN1[1]=1 and KIN2[1]=0 then $f_1(\mathbf{x}_i)=x_{i1}$.
    b. if KIN1[1]=1 and KIN2[1]=1 then $f_1(\mathbf{x}_i)=x_{i1}^2$.
    c. if KIN1[1]=1 and KIN2[1]=2 then $f_1(\mathbf{x}_i)=x_{i1}x_{i2}$.

11. *Y - Vector with response variable selected for analysis by user. Global Variable.
12. NDEC - Used repeatedly as indice for decisions on whether tasks described above should be carried out. Typically 0 or 1 but for decision on MLE can be 0, 1 or 2. No MLE = 0, FULL MLE = 1, ONETIME algorithm = 2. Input by user.
13. **SDIFF - N*(N-1)/2 by NX matrix which contains |S[i][j] - S[k][j]| for i<k, k<N and j<NX. Global Variable.
14. **F - Contains linear model input values. Global Variable.
15. *THETA - Vector of covariance function parameters. See (2.3.1).
16. *POWER - Vector of covariance function parameters. See (2.3.1).
17. *BETA - Vector of linear model parameters. See (2.3.1).
18. **V - Correlation matrix.
19. GAMMA - "Nugget" parameter in covariance function, $(**V' = **V + \gamma I)$, where I is the identity matrix.
20. SIGMAZ - "Scale" parameter in covariance function.
21. READ_PARAM - Indicator variable. If MLE has not been carried out program prepares to a user given file to read in model parameters. The order of parameters in file is: SIGMAZ, GAMMA, -2*LN Likelihood, *BETA,*THETA,*POWER.

    The following variables are used only in likelihood computations.

22. NLOOP - For FULL MLE it is the number of random starts for computing MLE, suggested value ≤ 5. For CHEAP MLE it is the number of rounds through all variables, suggested value is 15. Input by user.
23. NLIKE - The number of user given sets of model parameters which will be used to compute their likelihoods. Input by user.
24. FUNC_DECISION - Determines the type of constraints on the covariance function during the computation of the MLE:
    a. FUNC_DECISION=0: FULL MLE, all parameters except GAMMA=0 free to vary.
    b. FUNC_DECISION=1: COMMON (θ, p), $\theta_1 = , \ldots , = \theta_d$ and $p_1 = , \ldots , = p_d$.

c. FUNC_DECISION=2: SINGLE (θ, p), optimize over (θ_i, p_i) only.

d. FUNC_DECISION=3: GAMMA ONLY, Fix (θ, p) and optimize over GAMMA only.

This is a global Variable.

25. NGAM - Indicator variable that indicates whether GAMMA is fixed = 0. Global Variable.

26. OLD_THETA - Contains value of θ_i when optimizing one at a time on x_i. Global Variable.

27. OLD_POWER - Contains value of p_i when optimizing one at a time on x_i. Global Variable.

28. NPK - The variable which is currently being optimized during one at a time MLE. Global Variable.

29. LIKE - The likelihood value.

30. OLD_LIKE - The likelihood value from previous stage.

The following variables are used only in prediction and cross-validation computations.

31. NUMPRED - The number of points at which user wants to predict. Input by user.

32. RAND_IND - Indicator variable states whether user would like set of random input points for prediction, yes = 1. Input by user.

33. TRUEY_IND - If user gives a set of points for prediction, TRUEY_IND is indicator variable stating whether true responses are available, yes = 1. Input by user.

34. **PREDX - Set of points which will be used for prediction. Input by user if RAND_IND = 0.

35. *TRUEY - Response values for prediction sites. Input by user if TRUEY_IND = 1.

36. *YHAT - Prediction estimates for PREDX.

37. *EMSE - Expected RMSE estimates for PREDX.

38. **SUBV - Used in Cross-Validation. **SUBV=$**V_{-i}$, where $**V_{-i}$ is **V with the i^{th} row and column deleted.

39. **SUBS - Used in Cross-Validation. **SUBS=$**S_{-i}$, where $**S_{-i}$ is **S with the i^{th} row deleted.

40. *SUBY - Used in Cross-Validation. *SUBY=$*Y_{-i}$,where $**Y_{-i}$ is **Y with the i^{th} row deleted.

41. **FPRED - Linear model terms for prediction sites.

42. *CVYHAT - Stores Cross-Validation estimates of response.

43. *CVMSE - Stores Cross-Validation estimates of Expected RMSE.

The following variables are used only in main effects computations.

44. INTLEV - The highest order of interaction levels user is interesting in computing. INTLEV=1 main effects only, INTLEV=2, Second order interactions. Input by user.

45. USESIG - Use "significance test" to determine which variables to include in main effects computations. The "significance test" is $H_0: \theta_i = 0$, significance level is $\Delta(-2ln\ L) > 6$. Input by user.

46. INT_IND - Indicator vector of length NX which states which variables are included for main effects computations, yes=1.

47. NUMPTS - The number of points at which to compute main effects. Currently hardwired into program at NUMPTS=21.

48. *MUIND - Vector which contains labels i of x_i the variables for which main effects are computerd

49. *X - Vector of points at which to compute main effects.

50. **MU - Results of main effects. For main effects columns are for different variables. For higher order interactions columns is results at X_{jk}.

51. MU0 - Overall mean.

52. *MU1 - Main effects.

53. NUMINTR - Number of interactions computed so far.

54 **INTR - Contains integration values for variables included for main effects computations.

55 *MULEXP - Product of integrals for variables not included for main effects computations.

56 *NULEXP - Product of integrals for variables included for main effects computations.

57 SSY - estimate of variability of estimated response surface from 1000 randomly selected points, $1/n_r \sum(\hat{y}_i - \mu_0)^2$.

58 *SS - Squared deviation of main effects from zero.

A1.1.3 Subroutines

Subroutines which can be found in the source files listed above except **rb_alloc.c** are briefly described below. For clarity they are written in this documentation as SUBROUTINE(). In the source code they are not capitalized.

1. void COVM() - Covariance function when computing full MLE. In **rb_covm.c**.
2. void COVM1() - Covariance function when computing MLE with common (θ,p) for all variables. In **rb_covm.c**.
3. void CHP_COVM() - Covariance function when computing MLE for ONE-TIME. Optimizing over one variable at a time, both θ and power. In **rb_covm.c**.
4. void SLC_COVM() - Covariance function when computing MLE for FORWARD. Optimizing over one variable at a time, but power fixed. In **rb_covm.c**.
5. double CALMU0() - Computes overall integrated mean. In **rb_effects.c**.
6. void CALMU1() - Computes main effects. In **rb_effects.c**.
7. void CALMU2() - Computes interaction effects. In **rb_effects.c**.
8. void INIT_ME() - Is the structure within which main effects, etc. are computed. In **rb_effects.c**.
9. double GETLIKE() - Computes likelihood for Gaussian stochastic process. In **rb_like.c**.
10. double COMPMLE() - Computes all parameter estimates for MLE. GETLIKE only computes estimates for θ, p, γ. In **rb_like.c**.
11. double CHPMLE() - Structure which computes CHEAP MLE. In **rb_like.c**.
12. void DET_SIG() - Determines "significance" of input variables. $H_0: \theta_i = 0$. Critical value is $\hat{\theta}_i \leq 6.0$. In **rb_like.c**.
13. void MULT_LIKE() - Structure which computes as many likelihoods as desired for user given parameter values. In **rb_like.c**.

14. void DES_DIST() - Computes $|s_{ik} - s_{jk}|$ for all $i < j \leq N$ and $1 \leq k \leq NX$. In **rb_matman.c**.
15. void VEC_V() - Takes **V, the covariance matrix, and translates it to a vector. In **rb_matman.c**.
16. void MAT_V() - Takes the vector version of **V and translates it back to a matrix. In **rb_matman.c**.
17. void CD() - Computes Cholesky decompostion. Output is lower triangular matrix. In **rb_math.c**.
18. void INVMAT() - Inverts matrix. Result is returned in lower triangular part of input matrix. In **rb_math.c**.
19. double RB_ABS() - Computes absolute value of x. In **rb_math.c**.
20. void QR() - Computes QR decompostion. In **rb_math.c**.
21. void XTOF() - Takes design matrix **S and computes **F the input matrix for the linear model part of stochastic process. In **rb_math.c**.
22. void GET_RINVY() - Computes $R_S^{-1}y$ for main effects. In **rb_math.c**.
23. double NORMIN() - Computes integral of $exp(-\theta|s_{ik}-s_{jk}|^{p_k})$. In **rb_math.c**.
24. double FUNK() - Function used in optimization routine for computing MLE's. In FUNK, FUNC_DECISION determines which covariance function to use. In **rb_opt.c**.
25. int AMOEBA() - The optimization routine used to compute MLE's It has been taken from Nelder & Mead *Numerical Recipes* for FORTRAN and translated to C. In **rb_opt.c**.
26. double RUNOPT() - Function from which optimization routine is run. In **rb_opt.c**.
27. void INITIAL() - Computes some initial calculations needed for getting predictions. In **rb_predict.c**.
28. void PREDICT() - Computes $\hat{y}$. In **rb_predict.c**.
29. void GET_EMSE() - Computes $MSE(\hat{y})$. In **rb_predict.c**.
30. void GET_PRED() - Function from which prediction and MSE estimates are computed. In **rb_predict.c**.

31. void GET_CV() - Function from which Cross-validation results are computed. In **rb_predict.c**.

32. double MEPRED() - Computes predictions for estimate of overall estimate of response surface variability. In **rb_predict.c**.

33. void PRNT_LIKE() - Routine prints results of likelihood calculations. In **rb_print.c**.

34. void PRED_OUT() - Routine prints results of prediction part of program. In **rb_print.c**.

A1.1.4 Example of Input To Dace

```
Title          /* title for job */
50 20 1 4658    /* n,nx,p,seed for random number*/
1 1 1           /* nfiles, ny,sely   */
toya.des50y     /* design file and response file  */
0               /* kin1 */
0               /* kin2 */
0               /* gamma */
/*  if any following decisions is no (=0) then associated
    inputs need not be entered                          */
1               /* decision on mle (1,full,2=cheap) */
5               /* # random starts for full, # loops thru cheap mle */
like.dump       /* dump file for mle */
mle.out         /* mle results file     */

1               /* decision on multiple likelihood calculations */
30              /* # of likelihoods to calculate         */
param.in        /* file with parameters for likelihood calculations */
like.out        /* file name for likelihood output       */

1               /* decision on prediction            */
param.in        /* file with parameters to read if necessary */
100 0           /* # points to predict at, from random inputs (yes=1)  */
toya.ran100y    /* file name with pts to predict at          */
1               /* Do you have true responses for prediction?  */
pred.out        /* file name for prediction output           */
```

```
1           /* decision for cv calcualations          */
cv.out      /* file name for cv output                */

1           /* decision for main effects calculations */
2 1         /* interaction level to compute, use sign level */
toya        /* prefix for output files: .mu0, .me, .int, .med added */
```

A1.1.5 Description of Output

The output files are reasonably self explanatory except for the main effects output and to a lesser extent the computation of likelihoods. Except for the main effects output, the output for each type of computation is their own file. As mentioned the output for the main effects is spread over four output files. There is also a dump file for maximum likelihood estimates which contains information about the optimization process, but the file **mle.out** contains all pertainent information.

All output contains the job title the parameter estimates used for the computation and the name of the data file used for the job. For the main effects output this information is in the file with suffix **.med**. The output files for the likelihood computations do not contain the parameter estimates. The results for prediction, cross-validation and maximum likelihood calculations are labeled and should be self explanatory. The parameter estimates from the MLE calculations are in the same order as needed for **param.in** file to make the input file easier to construct. The prediction and cross-validation output has the same format. A line after the parameter estimates gives the empirical RMSE, the standard deviation of the empirical RSME and the maximum deviation at the predicted values. A list of true and predicted values with $\hat{RMSE}(Y)$ follows the summary statistics.

The main effects output is divided into four output files with the suffices described above. They will be refered to here as **output.med**, **output.me**, **output.int**, and **output.sf**. The output file **output.med** contains the parameter estimates, etc. and the ANOVA-like sums of squares for main effects and higher order interactions. The output file **output.mu0** contains the value of $\int \hat{y}(\mathbf{x}) d\mathbf{x}$, β, KIN1 and KIN2. The main effects computations assume that a constant model has been used. The value of μ_0 in (2.4.1) is obtained by adding $\int \hat{y}(\mathbf{x}) d\mathbf{x}$ and β. The main effects in (2.4.2) are found in the output file **output.me**. The first row of the output file contains the number of factors which the main effects

were computed for and the indices for those factors. The main effects values start in the second row of the output file and the columns are the results for the different factors. The interaction effects in (2.4.3) and joint effects are constructed the same way and are in the files **output.int** and **output.sf** respectively. The joint effects are

$$\eta_{ij}(x_i, x_j) = \int y(\mathbf{x}) \prod_{k \neq i,j} dx_k .$$

The interaction effects are for all pairs of variables i, j, $i < j$, found in the output file **output.me**. The data in the respective output files are the interaction or joint effects values over a grid of points with the second dimension changing fastest and the data being read row by row. The matrices for the effects are also ordered with the second variable increasing fastest.

```
/***************This is dace.h*****************************/
int N;
int NX;
int P;
double **S;
double *Y;
double **V;
double **F;
double **SDIFF;

int NOPT;
int NGAM;
int NPK;
int FUNC_DECISION;
int NFCALLS;
double OLD_THETA;
double OLD_POWER;
double OLD_GAMMA;
double **TV;

char DUMP_FILE[40];
char MLE_FILE[40];
char PRED_FILE[40];
/*********************************************************************/
/*              This is dace.c                                       */
/*This is the main function for dace and calls all other routines*/
/*-------------------------------------------------------------*/
#include<stdio.h>
#include<string.h>
#include<time.h>
#include "dace.h"
#include "rb_alloc.h"
#include "rb_covm.h"
#include "rb_effects.h"
#include "rb_like.h"
#include "rb_matman.h"
#include "rb_predict.h"
#include "rb_dace_prnt.h"
#define RAND_MAX (pow(2,31)-1)
/*-------------------------------------------------------------*/
main()
{
  int i,j,k,nfiles,ny,sely,*kin1,*kin2,ndec,seed,read_param=1;
  int numpred,truey_ind,rand_ind;
  int nloop,itmax,nlike;
  int intlev,usesig,*int_ind;
  double **s;
```

```
double *theta,*power,gamma,*beta,sigmaz,like;
double **predx,*truey,ran_num,*yhat,*emse;
double tol,**tempy;
long temp_ran;
char temp_name[100],*fname,*oname,*data_file,*data_file2,*title;
char temp_title[100],datetime[30];
long clock;

FILE *in1;
FILE *in2;
FILE *out;
FILE *out1;

/* Read in job title and dimensions of the problem          */

 clock=time(0);
 strcpy(datetime,ctime(&clock));

 title=gets(temp_title,100);

scanf("%d %d %d %d",&N,&NX,&P,&seed);
scanf("%d %d %d",&nfiles,&ny,&sely);

 scanf("%s",temp_name);
 data_file=AllocChar(strlen(temp_name));
 strcpy(data_file,temp_name);

 if(nfiles==2)
 {
   scanf("%s",temp_name);
   data_file2=AllocChar(strlen(temp_name));
   strcpy(data_file2,temp_name);
   in2=fopen(data_file2,"r");
 }

/*  Allocate space to standard variables                    */

s=AllocDouble2(N,NX);
SDIFF=AllocDouble2(N*(N-1)/2,NX);
Y=AllocDouble(N);
theta=AllocDouble(NX);
power=AllocDouble(NX);
beta=AllocDouble(P);
kin1=AllocInt(P);
kin2=AllocInt(P);
F=AllocDouble2(N,P);
V=AllocDouble2(N,N);
```

```
  int_ind=AllocInt(NX);
  tempy=AllocDouble2(N,ny);

/*  Read in data                                */

  in1=fopen(data_file,"r");

  for(j=0;j<N;j++)
  {
    for(i=0;i<NX;i++) fscanf(in1,"%lf",&s[j][i]);
    for(k=0;k<ny;k++)
    {
      if(nfiles==1)fscanf(in1,"%lf",&tempy[j][k]);
      if(nfiles==2)fscanf(in2,"%lf",&tempy[j][k]);
    }
    Y[j]=tempy[j][sely-1];
  }

  FreeDouble2(N,ny,tempy);
  fclose(in1);
  if(nfiles==2)fclose(in2);

  for(i=0;i<P;i++) scanf("%d",&kin1[i]);
  for(i=0;i<P;i++) scanf("%d",&kin2[i]);
  scanf("%lf",&gamma);

  NGAM= (gamma) ? 1 : gamma ;

  srandom(seed);

  scanf("%d",&ndec);

/*  This section computes MLEs using one at a time likelihood */
/*  estimation techniques              */

  if(ndec)
  {
    read_param=0;
    itmax=100;
    tol=0.0001;

    scanf("%d",&nloop);

    scanf("%s",temp_name);
    strcpy(DUMP_FILE,temp_name);

    scanf("%s",temp_name);
```

```
strcpy(MLE_FILE,temp_name);

out=fopen(MLE_FILE,"w");
out1=fopen(DUMP_FILE,"w");

xtof(s,N,P,kin1,kin2,F);
des_dist(N,NX,s,SDIFF);

fputs(title,out);
fputs(title,out1);
fprintf(out,"\n%s\n",datetime);
fprintf(out1,"\n%s\n",datetime);

if(ndec==1)
{
  fprintf(out,"\n");
  fprintf(out1,"\n");
  fprintf(out,"RESULTS FROM FULL MLE\n");
  fprintf(out1,"RESULTS FROM FULL MLE\n");
  fprintf(out,"\n");
  fprintf(out1,"\n");
  itmax=itmax*50;
 like=mle(out,out1,N,NX,P,SDIFF,F,Y,theta,power,&gamma,beta,&sigmaz,tol,\e
 nloop,itmax);
}

if(ndec==2)
{
  fprintf(out,"\n");
  fprintf(out1,"\n");
  fprintf(out,"RESULTS FROM ONE AT A TIME MLE\n");
  fprintf(out1,"RESULTS FROM ONE AT A TIME MLE\n");
  fprintf(out,"\n");
  fprintf(out1,"\n");

 like=chpmle(out,out1,N,NX,P,SDIFF,F,Y,theta,power,&gamma,beta,&sigmaz,tol,\e
 nloop,itmax);
}

fprintf(out,"\n");
fprintf(out,"THE DATA FOR THIS RUN IS IN THE FILE %s",data_file);
if(nfiles==2)fprintf(out," AND %s",data_file2);
fprintf(out,"\n");
fprintf(out1,"\n");
fprintf(out1,"THE DATA FOR THIS RUN IS IN THE FILE %s",data_file);
if(nfiles==2)fprintf(out1," AND %s",data_file2);
fprintf(out1,"\n");
```

```
    fclose(out);
    fclose(out1);


  }

  scanf("%d",&ndec);

/*  This section computes nlike likelihood values for     */
/*    nlike sets of theta and power  */

  if(ndec)
  {
    read_param=0;
    scanf("%d",&nlike);
    scanf("%s",temp_name);
    fname=AllocChar(strlen(temp_name));
    strcpy(fname,temp_name);
    in1=fopen(fname,"r");
    FreeChar(strlen(temp_name),fname);

    scanf("%s",temp_name);
    fname=AllocChar(strlen(temp_name));
    strcpy(fname,temp_name);
    out=fopen(fname,"w");
    FreeChar(strlen(temp_name),fname);

    fputs(title,out);
    fprintf(out,"\n%s\n",datetime);
    fprintf(out,"\n");
    fprintf(out,"LIKELIHOOD RESULTS\n");
    fprintf(out,"\n");

    xtof(s,N,P,kin1,kin2,F);
    des_dist(N,NX,s,SDIFF);

    for(j=0;j<nlike;j++)
    {
      for(i=0;i<NX;i++) fscanf(in1,"%lf",&theta[i]);
      for(i=0;i<NX;i++) fscanf(in1,"%lf",&power[i]);

      covm(N,NX,SDIFF,theta,power,gamma,V);
      like=compmle(N,P,Y,F,V,&sigmaz,beta);

      fprintf(out,"RESULTS FOR PARAMETER SET %d\n",j+1);
      fprintf(out,"-2*LN LIKELIHOOD= %lf\n",like);
      fprintf(out,"SIGMAZ= %lf\n",sigmaz);
```

```
      for(i=0;i<P;i++) fprintf(out,"BETA= %lf\n",beta[i]);
    }

    fprintf(out,"\n");
    fprintf(out,"THE DATA FOR THIS RUN IS IN THE FILE %s",data_file);
    if(nfiles==2)fprintf(out," AND %s",data_file2);
    fprintf(out,"\n");

      fclose(in1);
      fclose(out);

  }

  scanf("%d",&ndec);

/*  This section computes predicted values for given parameters and */
/*    given or randomly selected prediction points   */

  if(ndec)
  {

    if(read_param)
    {
      scanf("%s",temp_name);
      fname=AllocChar(strlen(temp_name));
      strcpy(fname,temp_name);
      in1=fopen(fname,"r");
      FreeChar(strlen(temp_name),fname);

      fscanf(in1,"%lf %lf %lf",&sigmaz,&gamma,&like);
      for(i=0;i<P;i++) fscanf(in1,"%lf",&beta[i]);
      for(i=0;i<NX;i++) fscanf(in1,"%lf",&theta[i]);
      for(i=0;i<NX;i++) fscanf(in1,"%lf",&power[i]);
      read_param=0;
    }

    scanf("%d %d",&numpred,&rand_ind);
    scanf("%d %d %d",&nfiles,&ny,&sely);
    predx=AllocDouble2(numpred,NX);

/*   computing random inputs for prediction if required   */

    if(rand_ind)
    {
     for(i=0;i<numpred;i++)
       for(j=0;j<NX;j++)
       {
```

```
        temp_ran=random();
        ran_num=temp_ran;
        predx[i][j]=ran_num/RAND_MAX-0.5;
      }
    }
    else
    {

/*    or reading inputs in from file                    */

      scanf("%s",temp_name);
      data_file=AllocChar(strlen(temp_name));
      strcpy(data_file,temp_name);

      if(nfiles==2)
      {
        scanf("%s",temp_name);
        data_file2=AllocChar(strlen(temp_name));
        strcpy(data_file2,temp_name);
        in2=fopen(data_file2,"r");
      }

      in1=fopen(data_file,"r");
      tempy=AllocDouble2(numpred,ny);
      truey=AllocDouble(numpred);

      for(j=0;j<numpred;j++)
      {
        for(i=0;i<NX;i++) fscanf(in1,"%lf",&predx[j][i]);
        for(k=0;k<ny;k++)
        {
          if(nfiles==1)fscanf(in1,"%lf",&tempy[j][k]);
          if(nfiles==2)fscanf(in2,"%lf",&tempy[j][k]);
        }
        truey[j]=tempy[j][sely-1];
      }

      FreeDouble2(numpred,ny,tempy);
      fclose(in1);
      if(nfiles==2)fclose(in2);
    }
    scanf("%s",temp_name);
    fname=AllocChar(strlen(temp_name));
    strcpy(fname,temp_name);

    out=fopen(fname,"w");
    FreeChar(strlen(temp_name),fname);
```

```
    yhat=AllocDouble(numpred);
    emse=AllocDouble(numpred);

    fputs(title,out);
    fprintf(out,"\n%s\n",datetime);
    fprintf(out,"\n");
    fprintf(out,"PREDICTION RESULTS\n");
    prnt_like(out,N,NX,P,theta,power,gamma,beta,sigmaz,like);
    fprintf(out,"\n");

    get_pred(N,NX,P,kin1,kin2,s,Y,theta,power,gamma,sigmaz,numpred,predx,yhat,emse);
    pred_out(out,numpred,truey_ind,truey,predx,yhat,emse);

    fprintf(out,"\n");
    fprintf(out,"THE DATA FOR THIS RUN IS IN THE FILE %s",data_file);
    if(nfiles==2)fprintf(out," AND %s",data_file2);
    fprintf(out,"\n");

    fclose(out);

  }

/*  This section computes cross-validation results for given parameters */

  scanf("%d",&ndec);

  if(ndec)
  {

    if(read_param)
    {
      scanf("%s",temp_name);
      fname=AllocChar(strlen(temp_name));
      strcpy(fname,temp_name);
      in1=fopen(fname,"r");
      FreeChar(strlen(temp_name),fname);

      fscanf(in1,"%lf %lf %lf",&sigmaz,&gamma,&like);
      for(i=0;i<P;i++) fscanf(in1,"%lf",&beta[i]);
      for(i=0;i<NX;i++) fscanf(in1,"%lf",&theta[i]);
      for(i=0;i<NX;i++) fscanf(in1,"%lf",&power[i]);
      read_param=0;
    }

    scanf("%s",temp_name);
    fname=AllocChar(strlen(temp_name));
    strcpy(fname,temp_name);
```

```
    out=fopen(fname,"w");
    FreeChar(strlen(temp_name),fname);

    fputs(title,out);
    fprintf(out,"\n%s\n",datetime);
    fprintf(out,"\n");
    fprintf(out,"CROSS-VALIDATION RESULTS\n");
    prnt_like(out,N,NX,P,theta,power,gamma,beta,sigmaz,like);
    fprintf(out,"\n");

    get_cv(out,N,NX,P,s,Y,theta,power,gamma,sigmaz,kin1,kin2);

    fprintf(out,"\n");
    fprintf(out,"THE DATA FOR THIS RUN IS IN THE FILE %s",data_file);
    if(nfiles==2)fprintf(out," AND %s",data_file2);
    fprintf(out,"\n");

    fclose(out);

  }

/*  This section computes main effects results for selected variables */

  scanf("%d",&ndec);

  if(ndec)
  {

   if(read_param)
   {
    scanf("%s",temp_name);
    fname=AllocChar(strlen(temp_name));
    strcpy(fname,temp_name);
    in1=fopen(fname,"r");
    FreeChar(strlen(temp_name),fname);

    fscanf(in1,"%lf %lf %lf",&sigmaz,&gamma,&like);
    for(i=0;i<P;i++) fscanf(in1,"%lf",&beta[i]);
    for(i=0;i<NX;i++) fscanf(in1,"%lf",&theta[i]);
    for(i=0;i<NX;i++) fscanf(in1,"%lf",&power[i]);
    read_param=0;
   }

   scanf("%d %d",&intlev,&usesig);

/*  either use significance levels to choose factors to compute */
/*  main effects for or input factors manually                  */
```

```
    if(usesig)
      det_sig(N,NX,P,s,Y,theta,power,gamma,kin1,kin2,int_ind);
    else
      for(i=0;i<NX;i++)scanf("%d",&int_ind[i]);

/*  preparation of output file names                    */

    scanf("%s",temp_name);
    oname=AllocChar(strlen(temp_name));
    strcpy(oname,temp_name);

    fname=AllocChar(strlen(oname)+4);
    strcpy(fname,oname);
    fname=strcat(fname,".med");
    out=fopen(fname,"w");
    FreeChar(strlen(oname)+4,fname);

    fputs(title,out);
    fprintf(out,"\n%s\n",datetime);
    fprintf(out,"\n");
    fprintf(out,"MAIN EFFECTS RESULTS\n\n");

    prnt_like(out,N,NX,P,theta,power,gamma,beta,sigmaz,like);
    fprintf(out,"\n");

    init_me(out,oname,N,NX,P,s,Y,theta,power,gamma,kin1,kin2,int_ind,intlev);

    fprintf(out,"\n");
    fprintf(out,"THE DATA FOR THIS RUN IS IN THE FILE %s",data_file);
    if(nfiles==2)fprintf(out," AND %s",data_file2);
    fprintf(out,"\n");

    fclose(out);

  }

  FreeDouble2(N,N,V);
  FreeDouble2(N,NX,s);
  FreeDouble2(N,P,F);
  FreeDouble(N,Y);
  FreeDouble(NX,theta);
  FreeDouble(NX,power);
  FreeDouble(P,beta);
  FreeDouble(numpred,yhat);
  FreeDouble(numpred,emse);
  FreeInt(P,kin1);
  FreeInt(P,kin2);
```

```
    FreeInt(NX,int_ind);

    return;
}
/*-------------------------------------------------------------*/
```

```
/***************This is rb_like.h*****************************/
extern double getlike();
extern double compmle();
extern double mle();
extern double chpmle();
extern void det_sig();
extern void mult_like();
/*****************************************************************/
/*              This is rb_like.c                                */
/*  These routines compute likelihood values                     */
/*---------------------------------------------------------------*/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include "rb_alloc.h"
#include "rb_covm.h"
#include "rb_math.h"
#include "rb_opt.h"
#include "rb_dace_prnt.h"
#include "dace.h"
/*---------------------------------------------------------------*/
double getlike(n,p,y,f,v,sigmaz,r,c)
  int n,p;
  double *y,**f,**v,**r,*c,*sigmaz;
{
  int i,j,k;
  double fcn,det,**ftilda,tempf,tempy,*z,*ytilda,ss,*resid,**q;
  ftilda=AllocDouble2(n,p+1);
  ytilda=AllocDouble(n);
  resid=AllocDouble(n);
  q=AllocDouble2(n,p);
  z=AllocDouble(n);

  cd(n,v,z);
  det=0.;
  for(i=0;i<n;i++) det=det-log(z[i]);
  det=2*det;

  for(k=0;k<n;k++)
  {
    for(i=0;i<p;i++)
    {
      tempf=f[k][i];
      for(j=0;j<k;j++) tempf=tempf-v[k][j]*ftilda[j][i];
      ftilda[k][i]=tempf*z[k];
    }
  }
```

```
  for(k=0;k<n;k++)
  {
    tempy=y[k];
    for(j=0;j<k;j++) tempy=tempy-v[k][j]*ytilda[j];
    ytilda[k]=tempy*z[k];
  }

  if(p==0)
  {
    ss=0.;
    for(i=0;i<n;i++) ss=ss+ytilda[i]*ytilda[i];
    *sigmaz=ss/n;

    fcn=n*log(*sigmaz) + det ;
    fcn=fcn/100.;
    return(fcn);
  }

  for(i=0;i<n;i++) ftilda[i][p]=ytilda[i];
  qr(ftilda,n,p,p+1,q,r,ytilda,c);

  for(i=0;i<n;i++)
  {
    resid[i]=0.;
    for(j=0;j<p;j++) resid[i]=resid[i]+q[i][j]*c[j];
    resid[i]=ytilda[i]-resid[i];
  }

  ss=0.;
  for(i=0;i<n;i++) ss=ss+resid[i]*resid[i];
  *sigmaz=ss/n;

  fcn=n*log(*sigmaz) + det;
  fcn=fcn/100.;

  FreeDouble2(n,p+1,ftilda);
  FreeDouble2(n,p,q);
  FreeDouble(n,ytilda);
  FreeDouble(n,resid);
  FreeDouble(n,z);

  return(fcn);
}
/*-----------------------------------------------------------------*/
double compmle(n,p,y,f,v,sigmaz,beta)
  int n,p;
  double *y,**f,**v,*sigmaz,*beta;
```

```
{
  int i,j;
  double **r,*c,fcn;
  c=AllocDouble(n);
  r=AllocDouble2(p,p);

  fcn=getlike(n,p,y,f,v,sigmaz,r,c);
  fcn=fcn*100.;
  for(i=0;i<p;i++)
  {
    beta[i]=c[i];
    for(j=p-1;j>i;j--) beta[i]=beta[i]-r[i][j]*beta[j];
    beta[i]=beta[i]/r[i][i];
  }

  FreeDouble(n,c);
  FreeDouble2(p,p,r);

  return(fcn);
}
/*-------------------------------------------------------------*/
double mle(out,out1,n,nx,p,sdiff,f,y,theta,power,gamma,beta,sigmaz,tol,nloop,itmax)
  int n,nx,p,nloop,itmax;
  double **sdiff,**f,*y,*theta,*power,*gamma,*beta,*sigmaz,tol;
  FILE *out,*out1;
{
  int i,j,k,iter,nopt;
  double like,t_like,*dx,ran_num,tempgam,RAND_MAX,*gtheta,*gpower,ggamma;
  double old_like=10000000;
  long temp_num;

  nopt= (NGAM) ? 2*nx+1 : 2*nx;
  dx=AllocDouble(nopt);
  gtheta=AllocDouble(nx);
  gpower=AllocDouble(nx);

  RAND_MAX=2147483647;
  for(j=0;j<nloop;j++)
  {
    for(i=0;i<nx;i++)
    {
      temp_num=random();
      ran_num=temp_num;
      dx[i]=log(ran_num/RAND_MAX+0.000000001);
      ran_num=random();
      dx[nx+i]=asin(2*(ran_num/RAND_MAX-0.5));
    }
```

```
  ran_num=rand();
  if (NGAM) dx[2*nx]=log(ran_num/RAND_MAX+0.0000000001);

  NFCALLS=0;
  FUNC_DECISION=0;
  like=runopt(dx,nopt,tol,&iter,itmax);

  fprintf(out1,"NFCALLS= %d # ITERS= %d MAX ITERS ALLOWED= %d\n",\
  NFCALLS,iter,itmax);

  for(i=0;i<nx;i++)
  {
    theta[i]=exp(dx[i]);
    power[i]=sin(dx[nx+i])/2+1.5;
  }
  *gamma= (NGAM) ? exp(dx[2*nx]) : 0;

  covm(n,nx,sdiff,theta,power,*gamma,V);
  t_like=compmle(n,p,y,f,V,sigmaz,beta);

  if(like<old_like)
  {
    for(i=0;i<nx;i++)
    {
      gtheta[i]=exp(dx[i]);
      gpower[i]=sin(dx[nx+i])/2+1.5;
    }
    ggamma= (NGAM) ? exp(dx[2*nx]) : 0;
    old_like=t_like;
  }

  prnt_like(out1,n,nx,p,theta,power,*gamma,beta,*sigmaz,t_like);
}

for(i=0;i<nx;i++)
{
  theta[i]=gtheta[i];
  power[i]=gpower[i];
}
*gamma=ggamma;
covm(n,nx,sdiff,theta,power,*gamma,V);
t_like=compmle(n,p,y,f,V,sigmaz,beta);

prnt_like(out,n,nx,p,theta,power,*gamma,beta,*sigmaz,t_like);

FreeDouble(nopt,dx);
FreeDouble(nx,gtheta);
```

```
  FreeDouble(nx,gpower);

  return(old_like);
}
/*-------------------------------------------------------------*/
double chpmle(out,out1,n,nx,p,sdiff,f,y,theta,power,gamma,beta,sigmaz,tol,nloop,itmax)
  int n,nx,p,nloop,itmax;
  double **sdiff,**f,*y,*theta,*power,*gamma,*beta,*sigmaz,tol;
  FILE *out,*out1;
{
  int i,j,k,iter,nopt;
  double like,old_like,t_like,*dx,ran_num,tempgam,RAND_MAX;
  long temp_num;

/*   FILE *out;
  FILE *out1;
  out=fopen(MLE_FILE,"w");
  out1=fopen(DUMP_FILE,"w"); */

  nopt= (NGAM) ? 3 : 2;
  dx=AllocDouble(nopt);

  RAND_MAX=2147483647;
  temp_num=random();
  ran_num=temp_num;
  dx[0]=log(ran_num/RAND_MAX+0.000000001);
  ran_num=random();
  dx[1]=asin(2*(ran_num/RAND_MAX-0.5));
  ran_num=rand();
  if (NGAM) dx[2]=log(ran_num/RAND_MAX+0.0000000001);

  NFCALLS=0;
  FUNC_DECISION=1;
  old_like=runopt(dx,nopt,tol,&iter,itmax);

  for(i=0;i<nx;i++)
  {
    theta[i]=exp(dx[0]);
    power[i]=sin(dx[1])/2+1.5;
  }
  *gamma= (NGAM) ? exp(dx[2]) : 0;

        covm(n,nx,sdiff,theta,power,*gamma,V);
        t_like=compmle(n,p,y,f,V,sigmaz,beta);

  prnt_like(out1,n,nx,p,theta,power,*gamma,beta,*sigmaz,old_like);
  prnt_like(out,n,nx,p,theta,power,*gamma,beta,*sigmaz,old_like);
```

```
  covm1(n,nx,sdiff,theta,power,*gamma,V);

  FUNC_DECISION=2;

  for(j=0;j<nloop;j++)
  {
    for(i=0;i<nx;i++)
    {
      OLD_THETA=theta[i];
      OLD_POWER=power[i];
      dx[0]=log(theta[i]+0.000000001);
      dx[1]=asin(2*(power[i]-1.5));
      if(NGAM) dx[2]=log(*gamma+0.0000000001);
      NPK=i;
      NFCALLS=0;

      like=runopt(dx,nopt,tol,&iter,itmax);

    fprintf(out1,"NFCALLS= %d # ITERS= %d MAX ITERS ALLOWED= %d\n",\
    NFCALLS,iter,itmax);

      if(like<old_like)
      {
        theta[i]=exp(dx[0]);
        power[i]=sin(dx[1])/2+1.5;
        *gamma= (NGAM) ? exp(dx[2]) : 0;
        covm(n,nx,sdiff,theta,power,*gamma,V);
        t_like=compmle(n,p,y,f,V,sigmaz,beta);
        old_like=t_like;
      }
    }

    fprintf(out,"FOR LOOP %d -2*LN LIKELIHOOD= %10.4le\n",j+1,old_like);
    prnt_like(out1,n,nx,p,theta,power,*gamma,beta,*sigmaz,old_like);
  }

  prnt_like(out,n,nx,p,theta,power,*gamma,beta,*sigmaz,old_like);
/*  fclose(out);
  fclose(out1); */

  FreeDouble(nopt,dx);

  return(old_like);
}
/*-----------------------------------------------------------------*/
void det_sig(n,nx,p,s,y,theta,power,gamma,kin1,kin2,int_ind)
  int n,nx,p,*kin1,*kin2,*int_ind;
```

```
  double **s,*y,*theta,*power,gamma;
{
  int i,j;
  double tlike,like,sigmaz;
  double **sdiff,**v,*ntheta,**r,*c,fcn,**f;

  sdiff=AllocDouble2(n*(n-1)/2,nx);
  v=AllocDouble2(n,n);
  f=AllocDouble2(n,p);
  r=AllocDouble2(p,p);
  c=AllocDouble(n);
  ntheta=AllocDouble(nx);

  des_dist(n,nx,s,sdiff);
  xtof(s,n,p,kin1,kin2,f);

  covm(n,nx,sdiff,theta,power,gamma,v);
  tlike=getlike(n,p,y,f,v,&sigmaz,r,c);
  tlike=tlike*100.;

  for(i=0;i<nx;i++)
  {
    int_ind[i]=0;
    for(j=0;j<nx;j++) ntheta[j]=theta[j];
    ntheta[i]=0;

    covm(n,nx,sdiff,ntheta,power,gamma,v);
    like=getlike(n,p,y,f,v,&sigmaz,r,c);
    like=like*100.;

    if(like-tlike>6)int_ind[i]=1;
  }

  FreeDouble2(n*(n-1)/2,nx,sdiff);
  FreeDouble2(n,n,v);
  FreeDouble2(n,p,f);
  FreeDouble2(p,p,r);
  FreeDouble(n,c);
  FreeDouble(nx,ntheta);

  return;
}
/*-------------------------------------------------------------
void mult_like(n,nx,p,kin1,kin2,s,y,theta,power,gamma,nlike)
   int n,nx,p,*kin1,*kin2,nlike;
   double **s,*y,*theta,*power,gamma;
{
```

```
  int i,j;
  double **sdiff,**v,**f,*beta,like,sigmaz;
  sdiff=AllocDouble2(n*(n-1)/2,nx);
  f=AllocDouble2(n,p);
  v=AllocDouble2(n,n);

    xtof(s,n,p,kin1,kin2,f);
    des_dist(n,nx,s,sdiff);

    for(j=0;j<nlike;j++)
    {
      for(i=0;i<nx;i++) fscanf(in,"%lf",&theta[i]);
      for(i=0;i<nx;i++) fscanf(in,"%lf",&power[i]);
      fclose(in);

      covm(n,nx,sdiff,theta,power,gamma,v);
      like=compmle(n,p,y,f,v,&sigmaz,beta);

      fprintf(out,"RESULTS FOR PARAMETER SET %d:\n",j);
      fprintf(out,"-2*LN LIKELIHOOD= %lf\n",like);
      fprintf(out,"SIGMAZ= %lf\n",sigmaz);
      for(i=0;i<P;i++) fprintf(out,"BETA= %lf\n",beta[i]);
    }
  return;
}
/*-----------------------------------------------------------------*/
```

```
/***************This is rb_predict.h*****************************/
extern void initial();
extern void predict();
extern void get_emse();
extern void get_pred();
extern void get_cv();
extern double mepred();
/*************************************************************************/
/*              This is rb_predict.c                                     */
/*  These routines are used for prediction and cross-validation    */
/*-----------------------------------------------------------------*/
#include<math.h>
#include<stdio.h>
#include "rb_math.h"
#include "rb_alloc.h"
#include "rb_matman.h"
#include "rb_covm.h"
/*-----------------------------------------------------------------*/
void initial(n,p,f,v,vec1,vec2,fvf)
  int n,p;
  double **f,**v,**vec1,**vec2,**fvf;
{
  int i,j,k;
  double **a;
  a=AllocDouble2(p,n);

  invmat(n,v);

  for(i=0;i<n;i++)
    for(j=i+1;j<n;j++) v[i][j]=v[j][i];

  for(i=0;i<p;i++)
    for(j=0;j<n;j++)
      for(k=0;k<n;k++)  a[i][j]=a[i][j]+f[k][i]*v[k][j];

  for(i=0;i<p;i++)
    for(j=0;j<p;j++)
      for(k=0;k<n;k++)  fvf[i][j]=fvf[i][j]+a[i][k]*f[k][j];

  if(p==1) fvf[0][0]=1./fvf[0][0];

  if(p!=1) invmat(p,fvf);

  for(i=0;i<p;i++)
    for(j=i+1;j<p;j++)  fvf[i][j]=fvf[j][i];

  for(i=0;i<p;i++)
```

```
    for(j=0;j<n;j++)
    {
      vec1[i][j]=0;
     for(k=0;k<p;k++)
       vec1[i][j]=vec1[i][j]+fvf[i][k]*a[k][j];
    }

  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
    {
     vec2[i][j]=-1.0*v[i][j];
     for(k=0;k<p;k++)  vec2[i][j]=vec2[i][j]+a[k][i]*vec1[k][j];
    }

  FreeDouble2(p,n,a);

  return;
}
/*-------------------------------------------------------------*/
void predict(n,nx,p,s,y,theta,power,numpred,x,fpred,vec1,vec2,yhat)
 int n,nx,p,numpred;
 double **s,*y,*theta,*power,**x,**fpred,**vec1,**vec2,*yhat;
{
 int i,j,k;
 double *r,*v1,*v2;
 r=AllocDouble(n);
 v1=AllocDouble(p);
 v2=AllocDouble(n);

 for(i=0;i<p;i++)
   for(j=0;j<n;j++)  v1[i]=v1[i]+vec1[i][j]*y[j];

 for(i=0;i<n;i++)
   for(j=0;j<n;j++)  v2[i]=v2[i]-vec2[i][j]*y[j];

 for(i=0;i<numpred;i++)
 {
   for(j=0;j<n;j++)
   {
     r[j]=0;
     for(k=0;k<nx;k++)  r[j]=r[j]-theta[k]*pow(rb_abs(s[j][k]-x[i][k]),power[k]);
     r[j]=exp(r[j]);
   }

   yhat[i]=0;
   for(j=0;j<n;j++) yhat[i]=yhat[i]+r[j]*v2[j];
```

```
    for(j=0;j<p;j++) yhat[i]=yhat[i]+fpred[i][j]*v1[j];
  }

  FreeDouble(n,r);
  FreeDouble(p,v1);
  FreeDouble(n,v2);

  return;
}
/*-----------------------------------------------------------------*/
void get_emse(n,nx,p,s,theta,power,sigmaz,numpred,x,fpred,vec1,vec2,fvf,emse)
  int n,nx,p,numpred;
  double **s,*theta,*power,sigmaz,**x,**fpred,**vec1,**vec2,**fvf,*emse;
{
  int i,j,k;
  double *r,d1;
  r=AllocDouble(n);

  for(i=0;i<numpred;i++)
  {
    for(j=0;j<n;j++)
    {
      r[j]=0;
      for(k=0;k<nx;k++)  r[j]=r[j]-theta[k]*pow(rb_abs(s[j][k]-x[i][k]),power[k]);
      r[j]=exp(r[j]);
    }

    emse[i]=1;

    for(j=0;j<p;j++)
    {
      d1=0;
      for(k=0;k<p;k++)  d1=d1+fvf[j][k]*fpred[i][k];
      emse[i]=emse[i]+fpred[i][j]*d1;
    }

    for(j=0;j<p;j++)
    {
      d1=0;
      for(k=0;k<n;k++)  d1=d1+vec1[j][k]*r[k];
      emse[i]=emse[i]-2*fpred[i][j]*d1;
    }

    for(j=0;j<n;j++)
    {
      d1=0;
      for(k=0;k<n;k++)  d1=d1+vec2[j][k]*r[k];
```

```
      emse[i]=emse[i]+r[j]*d1;
    }

    emse[i]=sigmaz*emse[i];
  }

  FreeDouble(n,r);

  return;
}
/*-------------------------------------------------------------------------*/
void get_pred(n,nx,p,kin1,kin2,s,y,theta,power,gamma,sigmaz,numpred,predx,yhat,emse)
  int n,nx,p,*kin1,*kin2,numpred;
  double **s,*y,*theta,*power,gamma,sigmaz,**predx,*yhat,*emse;
{
  int i,j;
  double **f,**sdiff,**v,**fpred,**vec1,**vec2,**fvf;

  f=AllocDouble2(n,p);
  v=AllocDouble2(n,n);
  sdiff=AllocDouble2(n*(n-1)/2,nx);
  fpred=AllocDouble2(numpred,p);
  vec1=AllocDouble2(p,n);
  vec2=AllocDouble2(n,n);
  fvf=AllocDouble2(p,p);

  des_dist(n,nx,s,sdiff);
  xtof(s,n,p,kin1,kin2,f);
  covm(n,nx,sdiff,theta,power,gamma,v);
  xtof(predx,numpred,p,kin1,kin2,fpred);
  initial(n,p,f,v,vec1,vec2,fvf);
  predict(n,nx,p,s,y,theta,power,numpred,predx,fpred,vec1,vec2,yhat);
  get_emse(n,nx,p,s,theta,power,sigmaz,numpred,predx,fpred,vec1,vec2,fvf,emse);

  FreeDouble2(n,p,f);
  FreeDouble2(n,n,v);
  FreeDouble2(n*(n-1)/2,nx,sdiff);
  FreeDouble2(numpred,p,fpred);
  FreeDouble2(p,n,vec1);
  FreeDouble2(n,n,vec2);
  FreeDouble2(p,p,fvf);

  return;
}
/*-------------------------------------------------------------------------*/
void get_cv(out,n,nx,p,s,y,theta,power,gamma,sigmaz,kin1,kin2)
  int n,nx,p,*kin1,*kin2;
```

```
  double **s,*y,*theta,*power,gamma,sigmaz;
  FILE *out;
{
  int i,j,k,m;
  double **sdiff,**v,**subv,**subs,*suby,**f,**predx,*yhat,*emse;
  double **vec1,**vec2,**fvf,**fpred,*cvyhat,*cvmse;

  v=AllocDouble2(n,n);
  subv=AllocDouble2(n-1,n-1);
  sdiff=AllocDouble2(n*(n-1)/2,nx);
  subs=AllocDouble2(n-1,nx);
  suby=AllocDouble(n-1);
  f=AllocDouble2(n-1,p);
  fpred=AllocDouble2(1,p);
  predx=AllocDouble2(1,nx);
  vec1=AllocDouble2(p,n);
  vec2=AllocDouble2(n,n);
  fvf=AllocDouble2(p,p);
  yhat=AllocDouble(1);
  emse=AllocDouble(1);
  cvyhat=AllocDouble(n);
  cvmse=AllocDouble(n);

  des_dist(n,nx,s,sdiff);
  covm(n,nx,sdiff,theta,power,gamma,v);

  for(i=0;i<n;i++)
  {
    m=0;
    for(j=0;j<n;j++)
    {
      if(i==j)
      {
        for(k=0;k<nx;k++) predx[0][k]=s[j][k];
      }
      else
      {
        suby[m]=y[j];
        for(k=0;k<nx;k++) subs[m][k]=s[j][k];
        for(k=0;k<n;k++)
        {
          if(k<i)subv[m][k]=v[j][k];
          if(k>i)subv[m][k-1]=v[j][k];
        }
        m++;
      }
    }
```

```
    xtof(subs,n-1,p,kin1,kin2,f);
    xtof(predx,1,p,kin1,kin2,fpred);
    initial(n-1,p,f,subv,vec1,vec2,fvf);
    predict(n-1,nx,p,subs,suby,theta,power,1,predx,fpred,vec1,vec2,yhat);
    get_emse(n-1,nx,p,subs,theta,power,sigmaz,1,predx,fpred,vec1,vec2,fvf,emse);

    cvyhat[i]=yhat[0];
    cvmse[i]=emse[0];
  }

  pred_out(out,n,1,y,s,cvyhat,cvmse);

  FreeDouble2(n,n,v);
  FreeDouble2(n-1,n-1,subv);
  FreeDouble2(n*(n-1)/2,nx,sdiff);
  FreeDouble2(1,p,fpred);
  FreeDouble2(n-1,nx,subs);
  FreeDouble(n-1,suby);
  FreeDouble2(n-1,p,f);
  FreeDouble2(1,nx,predx);
  FreeDouble2(p,n,vec1);
  FreeDouble2(n,n,vec2);
  FreeDouble2(p,p,fvf);
  FreeDouble(1,yhat);
  FreeDouble(1,emse);
  FreeDouble(n,cvyhat);
  FreeDouble(n,cvmse);

  return;
}
/*-----------------------------------------------------------------*/
double mepred(n,nx,p,s,y,theta,power,f,v,kin1,kin2)
  int n,nx,p,*kin1,*kin2;
  double **s,*y,*theta,*power,**f,**v;
{
  int i,j,numpred=1000;
  double **vec1,**vec2,**fvf,**fpred,**predx,*yhat;
  double yavg=0,ssy=0,RAND_MAX=2147483647;

  fpred=AllocDouble2(numpred,p);
  predx=AllocDouble2(numpred,nx);
  vec1=AllocDouble2(p,n);
  vec2=AllocDouble2(n,n);
  fvf=AllocDouble2(p,p);
  yhat=AllocDouble(numpred);

  for(i=0;i<numpred;i++)
```

```
    for(j=0;j<nx;j++) predx[i][j]=(random()/RAND_MAX)-0.5;

  xtof(predx,numpred,p,kin1,kin2,fpred);
  initial(n,p,f,v,vec1,vec2,fvf);
  predict(n,nx,p,s,y,theta,power,numpred,predx,fpred,vec1,vec2,yhat);

  for(i=0;i<numpred;i++) yavg=yavg+yhat[i];
  yavg=yavg/numpred;

  for(i=0;i<numpred;i++) ssy=ssy+(yhat[i]-yavg)*(yhat[i]-yavg);
  ssy=ssy/numpred;

  FreeDouble2(numpred,p,fpred);
  FreeDouble2(p,n,vec1);
  FreeDouble2(n,n,vec2);
  FreeDouble2(p,p,fvf);
  FreeDouble(numpred,yhat);

  return(ssy);
}
/*-------------------------------------------------------------------*/
```

```
/***************This is rb_effects.h*****************************/
extern double calmu0();
extern void calmu1();
extern void calmu2();
extern void init_me();
/*************************************************************************/
/*              This is rb_effects.c                                     */
/*  These routines compute main effects and interactions                 */
/*-----------------------------------------------------------------------*/
#include<stdio.h>
#include<string.h>
#include<math.h>
#include "rb_alloc.h"
#include "rb_covm.h"
#include "rb_like.h"
#include "rb_math.h"
#include "rb_matman.h"
#include "rb_predict.h"
/*-----------------------------------------------------------------------*/
double calmu0(n,rinvy,mulexp,numintr,intr,nulexp)
  int n,numintr;
  double *rinvy,*mulexp,**intr,*nulexp;
{
  int i,j;
  double result=0;

  for(i=0;i<n;i++) nulexp[i]=mulexp[i];

  for(i=0;i<numintr;i++)
    for(j=0;j<n;j++) nulexp[j]=nulexp[j]*intr[j][i];

  for(i=0;i<n;i++) result=result+nulexp[i]*rinvy[i];

  return(result);
}
/*-----------------------------------------------------------------------*/
void calmu1(out1,fname,n,nx,numpts,int_ind,s,theta,power,ssy,rinvy,numintr,\
intr,nulexp,mu0,mu)
  int n,nx,numpts,*int_ind,numintr;
  double ssy,*nulexp,**intr,**s,*theta,*power,*rinvy,mu0,**mu;
  char *fname;
  FILE *out1;
{
  int i,j,k,*muind;
  int cnt=0;
  double *x,*bulexp,etox,*ss;
```

```
FILE *out;

bulexp=AllocDouble(n);
x=AllocDouble(numpts);
muind=AllocInt(numintr);
ss=AllocDouble(numintr);

x[0]=-0.5;
for(i=1;i<numpts;i++) x[i]=x[i-1]+1./(numpts-1);

for(i=0;i<nx;i++) if(int_ind[i]) muind[cnt++]=i;

for(i=0;i<numintr;i++)
  for(k=0;k<numpts;k++)
  {
    mu[k][i]=0;
    for(j=0;j<n;j++)
    {
      etox=pow(rb_abs(s[j][muind[i]]-x[k]),power[muind[i]]);
      etox=exp(-1.*theta[muind[i]]*etox);
      bulexp[j]=nulexp[j]*etox/intr[j][i];
      mu[k][i]=mu[k][i]+bulexp[j]*rinvy[j];
    }
  }

for(k=0;k<numpts;k++)
  for(i=0;i<numintr;i++) mu[k][i]=mu[k][i]-mu0;

out=fopen(fname,"w");
fprintf(out," %d",numintr);
for(i=0;i<numintr;i++) fprintf(out," %d",muind[i]+1);
  fprintf(out,"\n");
for(k=0;k<numpts;k++)
{
  for(i=0;i<numintr;i++)
  {
    fprintf(out,"%8.3lf",mu[k][i]);
    ss[i]=ss[i]+mu[k][i]*mu[k][i];
  }
  fprintf(out,"\n");
}

for(i=0;i<numintr;i++)
  fprintf(out1,"       Var %2d MSE= %12.4le    MSE/VAR(Y)= %8.4lf\n",muind[i]+1,\
  ss[i]/numpts,(ss[i]/numpts)/ssy);

fclose(out);
```

```
  FreeDouble(n,bulexp);
  FreeDouble(numpts,x);
  FreeInt(numintr,muind);
  FreeDouble(numintr,ss);

  return;
}
/*-------------------------------------------------------------------*/
void calmu2(out2,oname,n,nx,numpts,int_ind,s,theta,power,beta,ssy,rinvy,numintr,\
intr,nulexp,mu0,mu1)
  int n,nx,numpts,*int_ind,numintr;
  double ssy,*nulexp,**intr,**s,*theta,*power,*beta,*rinvy,mu0,**mu1;
  char *oname;
  FILE *out2;
{
  int i,j,k,l,m,*muind;
  int cnt=0,cntint=0;
  double *x,*bulexp,temp,etox,etoy,**mu,*ss;
  char *fname,*fname1;

  FILE *out;
  FILE *out1;

  fname=AllocChar(strlen(oname)+4);
  strcpy(fname,oname);
  fname=strcat(fname,".int");

  fname1=AllocChar(strlen(oname)+3);
  strcpy(fname1,oname);
  fname1=strcat(fname1,".sf");

  out=fopen(fname,"w");
  out1=fopen(fname1,"w");

  bulexp=AllocDouble(n);
  x=AllocDouble(numpts);
  mu=AllocDouble2(numpts,numpts);
  muind=AllocInt(numintr);

  x[0]=-0.5;
  for(i=1;i<numpts;i++) x[i]=x[i-1]+1./(numpts-1);

  for(i=0;i<nx;i++) if(int_ind[i]) muind[cnt++]=i;

  ss=AllocDouble(numintr*(numintr-1)/2);

  for(i=0;i<numintr;i++)
```

```
for(m=i+1;m<numintr;m++)
{
 for(k=0;k<numpts;k++)
  for(l=0;l<numpts;l++)
  {
    mu[k][l]=0;
    for(j=0;j<n;j++)
    {
      temp=pow(rb_abs(s[j][muind[i]]-x[k]),power[muind[i]]);
      etox=exp(-1.*theta[muind[i]]*temp);
      temp=pow(rb_abs(s[j][muind[m]]-x[l]),power[muind[m]]);
      etoy=exp(-1.*theta[muind[m]]*temp);
      bulexp[j]=nulexp[j]*etox*etoy/(intr[j][i]*intr[j][m]);
      mu[k][l]=mu[k][l]+bulexp[j]*rinvy[j];
    }
  }

 for(k=0;k<numpts;k++)
 {
   for(j=0;j<numpts;j++) fprintf(out1," %8.3lf",mu[k][j]+beta[0]);
   fprintf(out1,"\n");
 }

 for(k=0;k<numpts;k++)
   for(j=0;j<numpts;j++)
   {
     mu[k][j]=mu[k][j]-mu1[k][i]-mu1[j][m]-mu0;
     ss[cntint]=ss[cntint]+mu[k][j]*mu[k][j];
   }

 ss[cntint]=ss[cntint]/(numpts*numpts);
 fprintf(out2,"Var %2d Var %2d MSE= %12.4le     MSE/VAR(Y)= %8.4lf\n",\
 muind[i]+1,muind[m]+1,ss[cntint],ss[cntint]/ssy);
 cntint++;

 for(k=0;k<numpts;k++)
 {
   for(j=0;j<numpts;j++) fprintf(out," %8.3lf",mu[k][j]);
   fprintf(out,"\n");
 }

}

fclose(out);
fclose(out1);

FreeChar(strlen(oname)+3,fname1);
```

```
  FreeChar(strlen(oname)+4,fname);
  FreeDouble(n,bulexp);
  FreeDouble(numpts,x);
  FreeDouble(numintr*(numintr-1)/2,ss);
  FreeDouble2(numpts,numpts,mu);
  FreeInt(numintr,muind);

  return;
}
/*-----------------------------------------------------------------*/
void init_me(out2,oname,n,nx,p,s,y,theta,power,gamma,kin1,kin2,int_ind,intlev)
  int n,nx,p,*kin1,*kin2,intlev,*int_ind;
  double **s,*y,*theta,*power,gamma;
  char *oname;
  FILE *out2;
{
  int i,j,k;
  int numintr=0,numpts=21;
  double **f,**sdiff,**v,*beta,like,sigmaz;
  double *rinvy,*mulexp,*nulexp,**intr,mu0,**mu1,ssy;
  char *fname,*fname1;

  FILE *out;

  beta=AllocDouble(p);
  rinvy=AllocDouble(n);
  mulexp=AllocDouble(n);
  nulexp=AllocDouble(n);
  intr=AllocDouble2(n,nx);
  sdiff=AllocDouble2(n*(n-1)/2,nx);
  v=AllocDouble2(n,n);
  f=AllocDouble2(n,p);

  des_dist(n,nx,s,sdiff);
  xtof(s,n,p,kin1,kin2,f);
  covm(n,nx,sdiff,theta,power,gamma,v);
  like=compmle(n,p,y,f,v,&sigmaz,beta);

  covm(n,nx,sdiff,theta,power,gamma,v);

  ssy=mepred(n,nx,p,s,y,theta,power,f,v,kin1,kin2);
  fprintf(out2,"  Variance of Y= %12.5le\n\n",ssy);

  get_rinvy(n,p,f,v,y,beta,rinvy);

  for(i=0;i<n;i++) mulexp[i]=1;
```

```
for(i=0;i<nx;i++)
{
  if(int_ind[i])
  {
    for(j=0;j<n;j++) intr[j][numintr]= normin(theta[i],power[i],s[j][i]);
    numintr++;
  }
  else
  {
    for(j=0;j<n;j++) mulexp[j]=mulexp[j]*normin(theta[i],power[i],s[j][i]);
  }
}

mu0=calmu0(n,rinvy,mulexp,numintr,intr,nulexp);

mu1=AllocDouble2(numpts,numintr);

fname=AllocChar(strlen(oname)+4);
strcpy(fname,oname);
fname=strcat(fname,".mu0");
out=fopen(fname,"w");

fprintf(out,"%12.4le\n",mu0);
for(i=0;i<p;i++) fprintf(out," %12.4le",beta[i]);
fprintf(out,"\n");
for(i=0;i<p;i++) fprintf(out," %d",kin1[i]);
fprintf(out,"\n");
for(i=0;i<p;i++) fprintf(out," %d",kin2[i]);
fprintf(out,"\n");
fclose(out);

FreeChar(strlen(oname)+4,fname);

if(intlev<=2)
{
  fname=AllocChar(strlen(oname)+3);
  strcpy(fname,oname);
  fname=strcat(fname,".me");

  calmu1(out2,fname,n,nx,numpts,int_ind,s,theta,power,ssy,rinvy,numintr,\
  intr,nulexp,mu0,mu1);

  FreeChar(strlen(oname)+3,fname);
}

if(intlev==2)
  calmu2(out2,oname,n,nx,numpts,int_ind,s,theta,power,beta,ssy,rinvy,numintr,\
```

```
    intr,nulexp,mu0,mu1);

  FreeChar(strlen(oname)+3,fname1);
  FreeDouble(n,rinvy);
  FreeDouble(n,mulexp);
  FreeDouble(n,nulexp);
  FreeDouble2(n,nx,intr);
  FreeDouble2(n*(n-1)/2,nx,sdiff);
  FreeDouble2(n,n,v);
  FreeDouble2(n,p,f);
  FreeDouble2(numpts,numintr,mu1);

  return;
}
/*-----------------------------------------------------------------*/
```

```
/***************This is rb_covm.h*****************************/
extern void covm();
extern void covm1();
extern void chp_covm();
extern void slc_covm();
/*************************************************************/
/*              This is rb_covm.c                            */
/* These routines compute covariance matrices                */
/* from the n*(n-1)/2 by nx matrix **sdiff                   */
/*-----------------------------------------------------------*/
#include<math.h>
#include<stdio.h>
#include "rb_math.h"
#include "rb_alloc.h"
#include "rb_matman.h"
/*-----------------------------------------------------------*/
void covm(n,nx,sdiff,theta,power,gamma,v)
  int n,nx;
  double **sdiff,*theta,*power,gamma,**v;
{
  int i,j,k;
  double *vdiff;
  vdiff=AllocDouble(n*(n-1)/2);

  for(i=0;i<n*(n-1)/2;i++)
  {
    for(k=0;k<nx;k++)
      vdiff[i]=vdiff[i]-theta[k]*pow(sdiff[i][k],power[k]);
    vdiff[i]=exp(vdiff[i]);
  }

  mat_v(n,vdiff,v);
  for(i=0;i<n;i++) v[i][i]=1.0+gamma;

  FreeDouble(n*(n-1)/2,vdiff);

  return;
}
/*-------------------------------------------------------------*/
void covm1(n,nx,sdiff,theta,power,gamma,v)
  int n,nx;
  double **sdiff,*theta,*power,gamma,**v;
{
  int i,j,k;
  double *vdiff;
  vdiff=AllocDouble(n*(n-1)/2);
```

```
  for(i=0;i<n*(n-1)/2;i++)
  {
    for(k=0;k<nx;k++)
      vdiff[i]=vdiff[i]-pow(sdiff[i][k],power[0]);
    vdiff[i]=exp(theta[0]*vdiff[i]);
  }

  mat_v(n,vdiff,v);
  for(i=0;i<n;i++) v[i][i]=1.0+gamma;

  FreeDouble(n*(n-1)/2,vdiff);

  return;
}
/*-----------------------------------------------------------------*/
void chp_covm(npk,n,sdiff,theta,power,gamma,ngam,old_theta,old_power,v)
  int n,npk,ngam;
  double **sdiff,*theta,*power,gamma,old_theta,old_power,**v;
{
  int i,j,k;
  double *vdiff;
  vdiff=AllocDouble(n*(n-1)/2);

  vec_v(n,v,vdiff);

  for(i=0;i<n*(n-1)/2;i++)
  vdiff[i]=vdiff[i]*exp(old_theta*pow(sdiff[i][npk],old_power)-\
  theta[0]*pow(sdiff[i][npk],power[0]));

  mat_v(n,vdiff,v);
  if(ngam=1)
    for(i=0;i<n;i++) v[i][i]=v[i][i]+gamma;

  FreeDouble(n*(n-1)/2,vdiff);

  return;
}
/*-----------------------------------------------------------------*/
void slc_covm(npk,n,sdiff,theta,old_theta,old_power,v)
  int n,npk;
  double **sdiff,*theta,old_power,old_theta,**v;
{
  int i,j,k;
  double *vdiff;
  vdiff=AllocDouble(n*(n-1)/2);

  vec_v(n,v,vdiff);
```

```
    for(i=0;i<n*(n-1)/2;i++)
      vdiff[i]=vdiff[i]*exp((old_theta-theta[0])*pow(sdiff[i][npk],old_power));

    mat_v(n,vdiff,v);

    FreeDouble(n*(n-1)/2,vdiff);

    return;
}
/*-----------------------------------------------------------------*/
```

```
/***************This is rb_opt.h*****************************/
extern double funk();
extern int amoeba();
extern double runopt();
/************************************************************************/
/*                This is rb_opt.c                              */
/*These routines are used for the optimization algorithms AMOEBA*/
/*----------------------------------------------------------------*/
#include<math.h>
#include "rb_alloc.h"
#include "rb_math.h"
#include "rb_like.h"
#include "dace.h"
/*----------------------------------------------------------------*/
double funk(dx)
  double *dx;
{
  int i,j,k;
  double **r,*c,*theta,*power,gamma,fcn,sigmaz;
  double **tv;

  tv=AllocDouble2(N,N);
  r=AllocDouble2(P,P);
  c=AllocDouble(N);

  NFCALLS++;

  if(FUNC_DECISION==0)
  {
/*   MLE - OPT 1 THETA,POWER FOR EACH VARIABLE  */

    theta=AllocDouble(NX);
    power=AllocDouble(NX);

    for(i=0;i<NX;i++)
    {
      theta[i]=exp(dx[i]);
      power[i]=sin(dx[NX+i])/2+1.5;
    }
      gamma= (NGAM) ? exp(dx[2*NX]) : 0;
      covm(N,NX,SDIFF,theta,power,gamma,tv);

    FreeDouble(NX,theta);
    FreeDouble(NX,power);
  }
  else if(FUNC_DECISION==1)
  {
```

```
/*   CHPMLE - OPT 1 THETA,POWER FOR ALL VARIABLES  */

    theta=AllocDouble(1);
    power=AllocDouble(1);

    theta[0]=exp(dx[0]);
    power[0]=sin(dx[1])/2+1.5;
    gamma= (NGAM) ? exp(dx[2]) : 0;
    covm1(N,NX,SDIFF,theta,power,gamma,tv);

    FreeDouble(1,theta);
    FreeDouble(1,power);

  }
  else
  {
   for(i=0;i<N;i++)
     for(j=i;j<N;j++)
     {
        tv[i][j]=V[i][j];
        tv[j][i]=tv[i][j];
     }
   switch(FUNC_DECISION)
   {
   case 2 :
   {
/*   CHPMLE - OPT 1 THETA,POWER AT TIME FOR EACH VARIABLE */

     theta=AllocDouble(1);
     power=AllocDouble(1);

     theta[0]=exp(dx[0]);
     power[0]=sin(dx[1])/2+1.5;
     gamma= (NGAM) ? exp(dx[2]) : 0;
     chp_covm(NPK,N,SDIFF,theta,power,gamma,NGAM,OLD_THETA,OLD_POWER,tv);

     FreeDouble(1,theta);
     FreeDouble(1,power);
     break;
   }
   case 3 :
   {
/*   OPT OVER GAMMA ONLY  */

     gamma=exp(dx[0]);
     tv[i][i]=tv[i][i]-OLD_GAMMA+gamma;
     break;
```

```
    }
    case 4 :
    {
/*    SLICE - OPT 1 THETA AT TIME POWER FIXED */

      theta=AllocDouble(1);

      theta[0]=exp(dx[0]);
      slc_covm(NPK,N,SDIFF,theta,OLD_THETA,OLD_POWER,tv);

      FreeDouble(1,theta);
      break;
    }
    }
  }

  fcn=getlike(N,P,Y,F,tv,&sigmaz,r,c);
  fcn=fcn*100;

  FreeDouble2(N,N,tv);
  FreeDouble2(P,P,r);
  FreeDouble(N,c);

  return(fcn);
}
/*-----------------------------------------------------------------*/
int amoeba(xi,fxi,nopt,ftol,itmax)
  int nopt,itmax;
  double *fxi,**xi,ftol;
{
  int i,j,k,ihi,inhi;
  int iter=0,ilo=1;
  double *pr,*prr,*pbar,rtol,ypr,yprr;
  double alpha=1.0,beta=0.5,gamma=2.0;

  pr=AllocDouble(nopt);
  prr=AllocDouble(nopt);
  pbar=AllocDouble(nopt);

  ihi= (fxi[0]>fxi[1]) ? 1 : 2;
  inhi= (fxi[0]<fxi[1]) ? 1 : 2;

  for(i=0;i<nopt+1;i++)
  {
    if(fxi[i]<fxi[ilo])ilo=i;
    if(fxi[i]>fxi[ihi])
    {
```

```
    inhi=ihi;
    ihi=i;
  }
  else if(fxi[i]>fxi[inhi]) if(i!=ihi)inhi=i;
}
rtol=2*rb_abs(fxi[ihi]-fxi[ilo])/(rb_abs(fxi[ihi])+rb_abs(fxi[ilo]));

while(rtol>ftol && iter<itmax)
{
 iter++;

 for(j=0;j<nopt;j++) pbar[j]=0;
 for(i=0;i<nopt+1;i++)
   if(i!=ihi) for(j=0;j<nopt;j++) pbar[j]=pbar[j]+xi[i][j];

 for(j=0;j<nopt;j++)
 {
   pbar[j]=pbar[j]/nopt;
   pr[j]=(1+alpha)*pbar[j]-alpha*xi[ihi][j];
 }
 ypr=funk(pr);

 if(ypr<=fxi[ilo])
 {
   for(j=0;j<nopt;j++)
     prr[j]=gamma*pr[j]+(1-gamma)*pbar[j];
   yprr=funk(prr);

   if(yprr<fxi[ilo])
   {
     for(j=0;j<nopt;j++) xi[ihi][j]=prr[j];
     fxi[ihi]=yprr;
   }
   else
   {
     for(j=0;j<nopt;j++) xi[ihi][j]=pr[j];
     fxi[ihi]=ypr;
   }
 }
 else if(ypr>=fxi[inhi])
 {

   if(ypr<fxi[ihi])
   {
     for(j=0;j<nopt;j++) xi[ihi][j]=pr[j];
     fxi[ihi]=ypr;
   }
```

```
  for(j=0;j<nopt;j++) prr[j]=beta*xi[ihi][j]+(1-beta)*pbar[j];
  yprr=funk(prr);
  if(yprr<fxi[ihi])
  {
    for(j=0;j<nopt;j++) xi[ihi][j]=prr[j];
    fxi[ihi]=yprr;
  }

  else
  {
    for(i=0;i<nopt+1;i++)
    {
      if(i!=ilo)
      {
        for(j=0;j<nopt;j++)
        {
          pr[j]=0.5*(xi[i][j]+xi[ilo][j]);
          xi[i][j]=pr[j];
        }
        fxi[i]=funk(pr);
      }
    }
  }
}
else
{
  for(j=0;j<nopt;j++) xi[ihi][j]=pr[j];
  fxi[ihi]=ypr;
}

ilo=1;

ihi= (fxi[0]>fxi[1]) ? 1 : 2;
inhi= (fxi[0]<fxi[1]) ? 1 : 2;

for(i=0;i<nopt+1;i++)
{
  if(fxi[i]<fxi[ilo])ilo=i;
  if(fxi[i]>fxi[ihi])
  {
    inhi=ihi;
    ihi=i;
  }
  else if(fxi[i]>fxi[inhi])
    if(i!=ihi)inhi=i;
}
rtol=2*rb_abs(fxi[ihi]-fxi[ilo])/(rb_abs(fxi[ihi])+rb_abs(fxi[ilo]));
```

```
  }

  FreeDouble(nopt,pr);
  FreeDouble(nopt,prr);
  FreeDouble(nopt,pbar);

  return(iter);
}
/*-----------------------------------------------------------------*/
double runopt(dx,nopt,tol,iter,itmax)
  int nopt,*iter,itmax;
  double *dx,tol;
{
  int i,j,k,nbest=0;
  double **xi,*ptxi,*fxi,fcn;

  xi=AllocDouble2(nopt+1,nopt);
  ptxi=AllocDouble(nopt);
  fxi=AllocDouble(nopt+1);

  for(j=0;j<nopt;j++) xi[0][j]=dx[j];
  for(k=0;k<nopt;k++)
    for(j=0;j<nopt;j++)  xi[k+1][j]= (k==j) ? dx[j]+1 : dx[j];

  for(k=0;k<nopt+1;k++)
  {
    for(j=0;j<nopt;j++) ptxi[j]=xi[k][j];
    fxi[k]=funk(ptxi);
  }

  *iter=amoeba(xi,fxi,nopt,tol,itmax);

  /*  write output results were here */

  fcn=fxi[0];
  for(k=1;k<nopt+1;k++)
    if(fcn>fxi[k])
    {
      nbest=k;
      fcn=fxi[k];
    }

  for(i=0;i<nopt;i++) dx[i]=xi[nbest][i];

  FreeDouble2(nopt+1,nopt,xi);
  FreeDouble(nopt,ptxi);
  FreeDouble(nopt+1,fxi);
```

```
    return(fcn);
}
/*-------------------------------------------------------------------*/
```

```
/**********************This is rb_print.h**********************/
/*                                                            */
extern void prnt_like();
extern void pred_out();
/************************************************************/
/*                    This is rb_print.c                      */
/* These routine print output for the different tasks         */
/************************************************************/
#include<stdio.h>
#include<math.h>
#include "rb_alloc.h"
/*-----------------------------------------------------------*/
/*  prnt_like prints parameter values and likelihood values  */
/*-----------------------------------------------------------*/
void prnt_like(out,n,nx,p,theta,power,gamma,beta,sigmaz,like)
  int n,nx,p;
  double *theta,*power,gamma,*beta,sigmaz,like;
/*  char file_name[40];*/
  FILE *out;
{
  int i;

/*  FILE *out;
  out=fopen(file_name,"a");*/

  fprintf(out,"\n");
  fprintf(out,"N= %d NX= %d\n",n,nx);
  fprintf(out,"SIGMAZ= %10.4le GAMMA= %8.4lf -2*LN LIKELIHOOD= %10.4le\n",\
  sigmaz,gamma,like);
  fprintf(out,"BETA=");
  for(i=0;i<p;i++)
  {
    fprintf(out," %10.4le",beta[i]);
    if((i+1)%5==0)fprintf(out,"\n");
  }
  if(p%5) fprintf(out,"\nTHETA= ");
  if(p%5==0) fprintf(out,"THETA= ");
  for(i=0;i<nx;i++)
  {
    fprintf(out," %10.4le",theta[i]);
    if((i+1)%5==0 && i+1<nx)fprintf(out,"\nTHETA= ");
  }
  if(p%5) fprintf(out,"\nPOWER= ");
  if(p%5==0) fprintf(out,"POWER= ");
  for(i=0;i<nx;i++)
  {
    fprintf(out," %10.4le",power[i]);
```

```
    if((i+1)%5==0 && i+1<nx)fprintf(out,"\nPOWER= ");
  }
  fprintf(out,"\n");
/*  fclose(out); */

}
/*-------------------------------------------------------------------------*/
/*   pred_out calculates summary statistics and prints prediction results   */
/*-------------------------------------------------------------------------*/
void pred_out(out,numpred,truey_ind,truey,predx,yhat,emse)
   int numpred,truey_ind;
   double *truey,**predx,*yhat,*emse;
   FILE *out;
{
   int i,j;
   double *diff,err=0,ss=0,diffmax=-1;
   diff=AllocDouble(numpred);

   if(truey_ind)
   {
     for(i=0;i<numpred;i++)
     {
       diff[i]=(truey[i]-yhat[i])*(truey[i]-yhat[i]);
       if(diff[i]>diffmax)  diffmax=diff[i];
       err=err+diff[i];
     }

     err=err/numpred;

     for(i=0;i<numpred;i++)  ss=ss+(diff[i]-err)*(diff[i]-err);

     ss=ss/numpred;

     fprintf(out,"  ERMSE      STD.ERR(ERMSE)     MAX|ERR|\n");
     fprintf(out,"%12.4le %12.4le     %12.4le\n\n",sqrt(err),sqrt(ss),\
     sqrt(diffmax));

     fprintf(out,"CASE          Y          YHAT        RMSE(YHAT)\n");
     for(i=0;i<numpred;i++)
     fprintf(out,"%4d %12.4le %12.4le %12.4le\n",i+1,truey[i],yhat[i],\
     sqrt(emse[i]));
   }
   else
   {
     fprintf(out,"CASE          YHAT        RMSE(YHAT)\n");
     for(i=0;i<numpred;i++)
     fprintf(out,"%4d  %12.4le %12.4le\n",i,yhat[i],sqrt(emse[i]));
```

```
    }

    FreeDouble(numpred,diff);

    return;
}
/*-------------------------------------------------------------*/
```

```
/***************This is rb_math.h*****************************/
extern void cd();
extern void invmat();
extern double rb_abs();
extern void qr();
extern double normin();
extern void xtof();
extern void get_rinvy();
/*********************************************************************/
/*              This is rb_math.c                                    */
/*  these are various math routines                                  */
/*-------------------------------------------------------------------*/
#include<math.h>
#include "rb_alloc.h"
/*-------------------------------------------------------------------*/
void cd(n,v,z)
   int n;
   double **v,*z;
{
   int i,j,k;
   double *vd;
   vd=AllocDouble(n);

   for(i=0;i<n;i++) vd[i]=v[i][i];

   for(i=0;i<n;i++)
   {
     z[i]=1./sqrt(v[i][i]);
     for(j=i+1;j<n;j++) v[j][i]=v[j][i]*z[i];

     for(j=i+1;j<n;j++)
        for(k=j;k<n;k++) v[k][j]=v[k][j]-v[j][i]*v[k][i];
   }

   for(i=0;i<n;i++) v[i][i]=vd[i];
   FreeDouble(n,vd);
   return;
}
/*-------------------------------------------------------------*/
void invmat(n,v)
    int n;
    double **v;
{
    int i,j,k,temp;
    double *z,*w,sum;
    w=AllocDouble(n);
    z=AllocDouble(n);
```

```
    cd(n,v,z);

    for(i=0;i<n;i++) v[i][i]=1./z[i];

    for(i=n-1;i>=0;i--)
    {
      v[i][i]=z[i];
      for(k=i;k>=0;k--)
      {
        sum=0;
        for(j=k+1;j<=i;j++)
        {
          sum=sum+v[j][i]*v[j][k];
          v[k][i]=(-sum*z[k]);
        }
      }
    }

    for(i=0;i<n;i++)
    {
      w[i]=0;
      for(j=i;j<n;j++) w[i]=w[i]+v[i][j]*v[i][j];
      for(j=i+1;j<n;j++)
      {
        v[j][i]=0;
        for(k=j;k<n;k++) v[j][i]=v[j][i]+v[i][k]*v[j][k];
      }
    }

    for(i=0;i<n;i++) v[i][i]=w[i];

    FreeDouble(n,z);
    FreeDouble(n,w);
    return;
}
/*-------------------------------------------------------------*/
double rb_abs(x)
  double x;
{
  if(x<0) x=-1*x;
  return(x);
}
/*-------------------------------------------------------*/
void qr(x,n,p,ncol,q,r,y,c)
  int n,p,ncol;
  double **x,*y,**q,**r,*c;
{
```

```
  int i,j,k;

  for(j=0;j<p;j++)
  {
    r[j][j]=0.;
    for(i=0;i<n;i++) r[j][j]=r[j][j]+x[i][j]*x[i][j];
    r[j][j]=sqrt(r[j][j]);

    for(i=0;i<n;i++) q[i][j]=x[i][j]/r[j][j];

    for(k=j+1;k<p;k++) r[j][k]=0.;
    for(i=0;i<n;i++)
      for(k=j+1;k<p;k++) r[j][k]=r[j][k]+x[i][k]*q[i][j];

    c[j]=0;
    for(i=0;i<n;i++) c[j]=c[j]+y[i]*q[i][j];

    for(i=0;i<n;i++)
      for(k=j+1;k<ncol;k++) x[i][k]=x[i][k]-q[i][j]*r[j][k];
  }

  return;
}
/*---------------------------------------------------------------------*/
double normin(theta,power,x)
  double theta,power,x;
{
  int i;
  double result;
  double u=-0.5,du=0.0001,sum=0;

  for(i=0;i<=10000;i++)
  {
    sum=sum+exp(-1.*theta*pow(rb_abs(x-u),power));
    u=u+du;
  }

  result=sum/10001;

  return(result);
}
/*---------------------------------------------------------------------*/
void xtof(s,n,p,kin1,kin2,f)
  int n,p,*kin1,*kin2;
  double **s,**f;
{
  int i,j;
```

```
  double ss;


  for(i=0;i<p;i++)
  {
      if(kin2[i]==0)

        if(kin1[i]==0)
         for(j=0;j<n;j++) f[j][i]=1;
        else
         for(j=0;j<n;j++) f[j][i]=s[j][kin1[i]-1];

      else
      {
       if(kin2[i]==kin1[i])
       {
        ss=0;
        for(j=0;j<n;j++) ss=ss+s[j][kin2[i]-1]*s[j][kin1[i]-1];
        for(j=0;j<n;j++) f[j][i]=1-n*s[j][kin1[i]-1]*s[j][kin2[i]-1]/ss;
       }
       else
         for(j=0;j<n;j++) f[j][i]=s[j][kin1[i]-1]*s[j][kin2[i]-1];
      }
  }

  return;
}
/*-----------------------------------------------------------------*/
void get_rinvy(n,p,f,v,y,beta,rinvy)
 int n,p;
 double **f,**v,*y,*beta,*rinvy;
{
 int i,j;
 double yhat,*resid;
 resid=AllocDouble(n);

 for(i=0;i<n;i++)
 {
  yhat=0;
  for(j=0;j<p;j++) yhat=yhat+f[i][j]*beta[j];
  resid[i]=y[i]-yhat;
 }

 for(i=0;i<n;i++)
 {
  rinvy[i]=0;
  for(j=0;j<n;j++)  rinvy[i]=rinvy[i]+v[i][j]*resid[j];
```

```
  }

  FreeDouble(n,resid);

  return;
}
/*-----------------------------------------------------------------*/
```

```
/***************This is rb_matman.h************************/
extern void des_dist();
extern void vec_v();
extern void mat_v();
/*************************************************************/
/*              This is rb_matman.c                          */
/*  These routines perform various matrix manipulations   */
/*-----------------------------------------------------------*/
#include "rb_math.h"
/*-----------------------------------------------------------*/
void des_dist(n,nx,s,sdiff)
  int n,nx;
  double **s,**sdiff;
{
  int i,j,k,m=0;

  for(i=0;i<nx;i++)
  {
    m=0;
    for(j=0;j<n;j++)
      for(k=j+1;k<n;k++)
        sdiff[m++][i]=rb_abs(s[j][i]-s[k][i]);
  }

  return;
}
/*-----------------------------------------------------------*/
void vec_v(n,mat,vec)
  int n;
  double **mat,*vec;
{
  int i,j,m=0;

  for(i=0;i<n;i++)
    for(j=i+1;j<n;j++)
      vec[m++]=mat[i][j];

  return;
}
/*-----------------------------------------------------------*/
void mat_v(n,vec,mat)
  int n;
  double **mat,*vec;
{
  int i,j,m=0;

  for(i=0;i<n;i++)
```

```
    for(j=i+1;j<n;j++)
    {
      mat[i][j]=vec[m++];
      mat[j][i]=mat[i][j];
    }

  return;
}
/*-----------------------------------------------------------------*/
```

```
/****************** This is rb_alloc.h ************************/
extern char **AllocChar2();
extern char *AllocChar();
extern void FreeChar();
extern long *AllocLong();
extern void FreeLong();
extern int *AllocInt();
extern int **AllocInt2();
extern void FreeInt();
extern void FreeInt2();
extern float *AllocFloat();
extern float **AllocFloat2();
extern void FreeFloat();
extern void FreeFloat2();
extern double *AllocDouble();
extern double **AllocDouble2();
extern void FreeDouble();
extern void FreeDouble2();
/*********************************************************************/
/*              This is rb_alloc.c                                   */
/* Routines used for allocating and freeing memory in DACE code*/
/*-------------------------------------------------------------*/
#include<malloc.h>
char **AllocChar2(n)
    int n;
{
    char **B;
    B = ( char **) calloc(n,sizeof(char *));
    return B;
}
/*-------------------------------------------------------------*/
char *AllocChar(n)
    int n;
{
    char *B;
    B = ( char *) calloc(n,sizeof(char));
    return B;
}
/*-------------------------------------------------------------*/
void FreeChar(n,B)
    int n;
    char *B;
{
    cfree(B, n, sizeof(char));
    return;
}
/*-------------------------------------------------------------*/
```

```
long *AllocLong(n)
    int n;
{
    long *B;
    B = ( long *) calloc(n,sizeof(long));
    return B;
}
/*-------------------------------------------------------------*/
void FreeLong(n,A)
     int n;
     long *A;
{
     cfree(A, n, sizeof(long));
}
/*-------------------------------------------------------------*/
int *AllocInt(n)
    int n;
{
    int *B;
    B = ( int *) calloc(n,sizeof(int));
    return B;
}
/*-------------------------------------------------------------*/
int **AllocInt2( n, p)
    int n,p;
{
    int i;
    int **A;
    A = (int **) calloc(n , sizeof(int *));
    for(i=0;i<n;i++) A[i]=AllocInt(p);
    return A;
}
/*-------------------------------------------------------------*/
void FreeInt(n,A)
     int n;
     int *A;
{
     cfree(A, n, sizeof(int));
}
/*-------------------------------------------------------------*/
void FreeInt2( n, p , A)
     int n,p;
     int **A;
{
     int i;
     for(i=(n-1);i>=0;i--) cfree(A[i], p, sizeof(int) );
     cfree(A, n , sizeof(int *));
```

```
}
/*-------------------------------------------------------------*/
float *AllocFloat(n)
    int n;
{
    float *B;
    B = ( float *) calloc(n,sizeof(float));
    return B;
}
/*-------------------------------------------------------------*/
float **AllocFloat2( n, p)
    int n,p;
{
    int i;
    float **A;
    A = (float **) calloc(n , sizeof(float *));
    for(i=0;i<n;i++) A[i]=AllocFloat(p);
    return A;
}
/*-------------------------------------------------------------*/
void FreeFloat(n,A)
     int n;
     float *A;
{
     cfree(A, n, sizeof(float));
}
/*-------------------------------------------------------------*/
void FreeFloat2( n, p , A)
     int n,p;
     float **A;
{
     int i;
     for(i=(n-1);i>=0;i--) cfree(A[i], p, sizeof(float) );
     cfree(A, n , sizeof(float *));
}
/*-------------------------------------------------------------*/
double *AllocDouble(n)
    int n;
{
    double *B;
    B = ( double *) calloc(n,sizeof(double));
    return B;
}
/*-------------------------------------------------------------*/
double **AllocDouble2( n, p)
    int n,p;
{
```

```
    int i;
    double **A;
    A = (double **) calloc(n , sizeof(double *));
    for(i=0;i<n;i++) A[i]=AllocDouble(p);
    return A;
}
/*---------------------------------------------------------------*/
void FreeDouble(n,A)
    int n;
    double *A;
{
    cfree(A, n, sizeof(double));
}
/*---------------------------------------------------------------*/
void FreeDouble2( n, p , A)
    int n,p;
    double **A;
{
    int i;
    for(i=(n-1);i>=0;i--) cfree(A[i], p, sizeof(double) );
    cfree(A, n , sizeof(double *));
}
/*---------------------------------------------------------------*/
```

Bibliography

1. Adams, B. M. and Woodall, W. H., (1989) "An Analysis of Taguchi's On-Line Process Control Procedure Under a Random Walk Model," *Technometrics*, 31, 401-414.
2. Aitkin, M., (1987) "Modeling Variance Heterogeneity in Normal Regression Using GLIM," *Applied Statistics*, 36, 332-339.
3. Alvarez, A. R., Abdi, B. L., Young, D. L., Weed, H. D., Teplik, J., and Herald, E. R., (1988) "Application of Statistical Design and Response Surface Methods to Computer-Aided VLSI Device Design," *IEEE Transactions on Computer-Aided Design*, 7, 272-288.
4. Andronikou, A. M., Bekey, G. A., and Masri, S. F., (1982) "Identification of Nonlinear Hysteretic Systems using Random Search," *Proceedings 6th IFAC Symposium on Identification and System Parameter Estimation*, 1, 263-268.
5. Atkinson, A. C., (1982) "Developments in the Design of Experiments," *International Statistical Review*, 50, 161-177.
6. Bailey, R. A., (1977) "Patterns of Confounding in Factorial Designs," *Biometrika*, 64, 597-603.
7. Bailey, R. A., (1982) "The Decomposition of Treatment Degress of Freedom in Quantitative Factorial Experiments," *Journal of the Royal Statistical Society - Series B*, 44, 63-70.
8. Bartlett, M. S. and Kendall, D. G., (1946) "The Statistical Analysis of Variance Heterogeneity and the Logarithmic Transformation," *Journal of the Royal Statistical Society - Series B*, 8, 128-150.
9. Beckman, R. J. and McKay, M. D., (1987) "Monte Carlo Estimation Under Different Distributions Using the Same Simulation," *Technometrics*, 29, 153-160.
10. Bekey, G. A. and Masri, S. F., (1983) "Random Search Techniques for Optimization of Nonlinear Systems with Many Parameters," *Math. Comput. Simulation*, 25, 210-213.
11. Berk, K. N. and Picard, R. R. , (1991) "Significance Tests for Saturated Orthogonal Arrays," *Journal of Quality Technology*, 23, 79-89.

12. Bernardo, M. C., Buck, R., Liu, L., Nazaret, W. A., Sacks, J., and Welch, W. J., (1992) "Integrated Circuit Design Optimization Using a Sequential Strategy," *IEEE Transactions in Computer-Aided Design*, 11, 361-372.

13. Bissel, A. F., (1989) "Interpreting Mean Squares in Saturated Fractional Designs," *Journal of Applied Statistics*, 16, 7-18.

14. Bissell, A. F., (1992) "Mean Squares in Saturated Fractional Designs Revisited," *Journal of Applied Statistics*, 19, 351-366.

15. Bjornstad, J. F., (1990) "Predictive Likelihood: A Review," *Statistical Science*, 5, 242-265.

16. Box, G. E. P. and Wilson, K. B., (1951) "On the Experimental Attainment of Optimum Conditions (with discussion)," *Journal of the Royal Statistical Society - Series B*, 13, 1-45.

17. Box, G. E. P. and Youle, P. V., (1955) "The Exploration and Exploitation of Response Surfaces: An Example of the Link Between the Fitted Surface and the Basic Mechanism of the System," *Biometrics*, 11, 287-323.

18. Box, G. E. P. and Hunter, J. S., (1957) "Multifactor Experimental Designs for Exploring Response Surfaces," *The Annals of Mathematical Statistics*, 28, 195-241.

19. Box, G. E. P. and Draper, N. R., (1959) "A Basis for the Selection of a Response Surface Design," *Journal of the American Statistical Association*, 54, 622-654.

20. Box, G. E. P. and Behnken, D. W., (1960) "Some New Three Level Designs for the Study of Quantitative Variables," *Technometrics*, 2, 455-475.

21. Box, G. E. P., (1963) "The Effects of Errors in the Factor Levels and Experimental Design," *Technometrics*, 5, 247-262.

22. Box, G. E. P. and Draper, N. R., (1963) "The Choice of a Second Order Rotatable Design," *Biometrika*, 50, 335-352.

23. Box, G. E. P. and N. R. Draper, (1969) *Evolutionary Operation,* Wiley, New York.

24. Box, G. E. P. and Draper, N. R., (1975) "Robust Designs," *Biometrika*, 62, 347-352.

25. Box, G. E. P., Hunter, W. G., and Hunter, J. S., (1978) *Statistics for Experimenters* , John Wiley , New York .

26. Box, G. E. P., (1982) ''Choice of Response Surface Design and Alphabetic Optimality,'' *Utilitas Mathematica*, 21B, 11-55.

27. Box, G. E. P. and Draper, N. R., (1982) ''Measures of Lack of Fit for Response Surface Designs and Predictor Variable Transformations,'' *Technometrics*, 23, 1-8.

28. Box, G. E. P. and Meyer, R. D., (1986) ''Dispersion Effects from Fractional Designs,'' *Technometrics*, 28, 19-27.

29. Box, G. E. P. and Meyer, R. D., (1986) ''An Analysis of Unreplicated Fractional Factorials,'' *Technometrics*, 28, 11-18.

30. Box, G. E. P. and Draper, N. R., (1987) *Empirical Model Building and Response Surfaces* , John Wiley & Sons , New York, N.Y. .

31. Box, G. E. P., (1988) ''Signal to Noise Ratios, Performance Criteria and Transformations (with discussion),'' *Technometrics*, 30, 1-40.

32. Box, G. E. P., Bisgaard, S., and Fung, C., (1988) ''An Explanation and Critique of Taguchi's Contributions to Quality Engineering,'' *Quality and Reliability Engineering International*, 4, 123-131.

33. Box, G. E. P. and Jones, G., (1992) ''Split-plot Designs for Robust Product Experimentation,'' *Journal of Applied Statistics*, 19, 3-26.

34. Boyles, R. A., (1991) ''The Taguchi Capability Index,'' *Journal of Quality Technology*, 23, 17-26.

35. Brayton, R. K., Hachtel, and Sangiovanni-Vincentelli, A. L., (1981) ''A Survey of Optimization Techniques for Integrated-Circuit Design,'' *Proceedings of the IEEE*, 69, 1334-1362.

36. Bullington, K. E., Hool, J. N., and Maghsoodloo, S., (1990) ''A Simple Method for Obtaining Resolution IV Designs for Use with Taguchi's Orthogonal Arrays,'' *Journal of Quality Technology*, 22, 260-264.

37. Buxton, J. R., (1992) ''Some Comments on the Application of Diagnostic Techiques in Empirical Modelling,'' *Journal of Applied Statistics*, 19, 211-222.

38. Chen, J. and Wu, C. F. J., (1991) ''Some Results on s^{n-k} Fractional Factorial Designs with Minimum Aberration or Optimal Moments,'' *Annals of*

Statistics, 19, 1028-1041.

39. Cheng, C.-S., (1985) "Run Order of Factorial Designs," in *Proceedings of Berkley Conference in Honor of Jerzy Neyman and Jack Kiefer*, vol. 2, pp. 619-633, Wadsworth Advanced Books and Software, Hayward, CA.

40. Cheng, C.-S., (1989) "Some Orthogonal Main Effects Plans for Asymmetrical Factorials," *Technometrics*, 31, 475-478.

41. Christensen, R., Johnson, W., and Pearson, L. M., (1992) "Prediction Diagnostics for Spatial Linear Models," *Biometrika*, 79, 583-592.

42. Cook, R. D. and Nachtsheim, C. J., (1980) "A Comparison of Algorithms for Constructing Exact D-Optimal Designs," *Technometrics*, 22, 315-324.

43. Cook, R. D. and Nachtsheim, C. J., (1989) "Computer-Aided Blocking of Factorial and Response Surface Designs," *Technometrics*, 31, 339-346.

44. Cox, D. R., (1990) "Role of Models in Statistical Analysis," *Statistical Science*, 5, 169-174.

45. Cressie, N., (1991) *Statistics for Spatial Data,* John Wiley & Sons , New York, N.Y. .

46. Currin, C., Mitchell, T., Morris, M., and Ylvisaker, D., (1991) "Bayesian Prediction of Deterministic Functions, with Application to the Design and Analysis of Computer Experiments," *Journal of the American Statistical Association*, 86, 953-963.

47. D'Errico, J. R. and Zaino, N. A., (1988) "Statistical Tolerancing Using a Modification of Taguchi's Method," *Technometrics*, 30, 397-405.

48. Daniel, C., (1959) "Use of Half-Normal Plots in Interpreting Factorial Two-Level Experiements," *Technometrics*, 1, 311-341.

49. Davidian, M. and Carroll, R. J., (1987) "Variance Function Estimation," *Journal of the American Statistical Association*, 82, 1079-1091.

50. Davis, P. J. and Rabinowitz, P., (1984) *Methods of Numerical Integration,* 2nd ed., Academic, Orlando, Fla..

51. Defeo, P. and Myers, R. H., (1992) "A New Look at Experimental Design Robustness," *Biometrika*, 79, 375-381.

52. Dehnad, K., (1989) *Quality Control, Robust Design and the Taguchi Method,* Wadsworth.

53. Deming, S. N. and Morgan, S. L., (1983) "Teaching the Fundamentals of Experimental Design," *Analytica Chimica Acta*, 150, 183-198.

54. Derringer, G. and Suich, R., (1980) "Simultaneous Optimization of Several Response Variables," *Journal of Quality Technology*, 12, 214-219.

55. Dietrich, C. R., (1991) "Modality of the Restricted Maximum Likelihood for Spatial Gaussian Random Fields," *Biometrika*, 78, 833-840.

56. Dietrich, C. R. and Osborne, M. R., (1991) "Estimation of Covariance Parameters in Kriging via Restricted Maximum Likelihood," *Mathematical Geology*, 23, 119-135.

57. Downing, D. J., Gardner, R. H., and Hoffman, F. O., (1985) "An Examination of Response Surface Methodologies for Uncertainty Analysis in Assessment Models," *Technometrics*, 27, 151-163.

58. Dozenaksay, N., Falten, F. W., and Tucker, W. T., (1991) "Identification of Out of Control Quality Characteristics in a Multivariate Manufacturing Environment," *Communications in Statistics - Theory and Methods*, 20, 2775-2790.

59. Draper, N. R., (1985) "Small Composite Designs," *Technometrics*, 27, 173-180.

60. Draper, N. R. and Lin, D. K. J., (1990) "Small Response Surface Designs," *Technometrics*, 32, 187-194.

61. Draper, N. R. and Lin, D. K. J., (1990) "Connections Between 2-Level Designs of Resolution III^* and V," *Technometrics*, 32, 283-288.

62. Draper, N. R., (1992) "Applied Regression Analysis Bibliography Update 1990-1991," *Communication in Statistics: Theory and Methods*, 21, 2415-2438.

63. Easterling, R. G., Johnson, M. E., Bement, T. R., and Nachtsheim, Christopher J., (1991) "Statistical Tolerancing Based on Consumer's Risk Considerations," *Journal of Quality Technology*, 23, 1-11.

64. Engel, J., (1992) "Modelling Variation in Industrial Experiments," *Applied Statistics*, 41, 579-594.

65. Fearn, T., (1992) "Box-Cox Transformations and the Taguchi Method: an Alternative Analysis of a Taguchi Case Study," *Applied Statistics*, 41, 553-562.

66. Fienberg, S. E., (1992) "A Brief History of Statistics in Three and One-Half Chapters: A Review Essay," *Statistical Science*, 7, 208-225.

67. Ford, I., Kitsos, C. P., and Titterington D. M., (1989) "Recent Advances in Nonlinear Experimental Design," *Technometrics*, 31, 49-60.

68. Franklin, M. F. and Bailey, R. A., (1977) "Selection of Defining Contrasts and Confounded Effects in Two-Level Experiments," *Journal of the Royal Statistical Society-C*, 26, 321-326.

69. Franklin, M. F., (1985) "Selecting Defining Contrasts and Confounded Effects in p^{n-m} Factorial Experiments," *Technometrics*, 27, 165-172.

70. Freeny, A. E. and Nair, V. N., (1992) "Robust Parameter Design with Uncontrolled Noise Variables," *Statistics Sinica*, July.

71. Fries, A. and Hunter, W. G., (1980) "Minimum Aberration 2^{k-p} Designs," *Technometrics*, 22, 601-608.

72. Gelfand, S. B. and Mitter, S. K., (1991) "Recursive Stochastic Algorithms for Global Optimization in R^d," *SIAM Journal on Control and Optimization*, 29, 999-1018.

73. Gill, P. E., Hammarling, S. J., Murray, W., Saunders, M. A., and Wright, M. H., (1986) "User's Guide to LSSOL (Version 1.0)," Dept. of Operations Research, Stanford University (Report SOL 86-1), Stanford University.

74. Giovannitti-Jensen, A. and Myers, R. H., (1989) "Graphical Assessment of the Prediction Capability of Response Surface Designs," *Technometrics*, 31, 159-171.

75. Grondona, M. O. and Cressie, N., (1991) "Using Spatial Considerations in the Analysis of Experiments," *Technometrics*, 33, 381-392.

76. Grove, D. M. and Davis, T. P., "Taguchi's Idle Column Method," *Technometrics*, 33, 349-354.

77. Hackl, P. and Ledolter, J., (1992) "A New Nonparametric Quality Control Technique," *Communications in Statistics - Simulations and Computation*, 21, 423-444.

78. Hahn, G. J. and Dershowitz, A. F., (1974) "Evolutionary Operation Today-Some Survey Results and Observations," *Applied Statistics*, 23, 214-218.

79. Hamada, M. and Wu, C. F. J., (1990) ''A Critical Look at Accumulation Analysis and Related Methods (with discussion),'' *Technometrics*, 32, 119-130.

80. Hamada, M., (1992) ''An Explanation and Criticism of Minute Accumulating Analysis: Taguchi's Method for Life Test Data,'' *Journal of Quality Technology*, 24, 70-77.

81. Hamada, M. and Wu, C. F. J., (1992) ''Analysis of Designed Experiments with Complex Aliasing,'' *Journal of Quality Technology*, 24, 130-137.

82. Handcock, M. S., (1991) ''On Cascading Latin Hypercube Designs and Additive Models for Experiments,'' *Communications in Statistics-Theory and Methods*, 20, 417-440.

83. Hardin, J. M. and Kurkjian, B. M., (1989) ''The Calculus for Factorial Arrangements: A Review and Bibliography,'' *Communications in Statistics: Theory and Methods*, 18, 1251-1277.

84. Harville, D. A., (1977) ''Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems,'' *Journal of the American Statistical Association*, 72, 320-340.

85. Harville, D. A. and Jeske, D. R., (1992) ''Mean Squared Error of Estimation or Prediction under a General Linear Model,'' *Journal of the American Statistical Association*, 87, 724-731.

86. Haslett, J., (1992) ''Spatial Data Analysis - Challenges,'' *The Statistician*, 41, 271-284.

87. Hill, W. J. and Hunter, W. G., (1966) ''A Review of Response Surface Methodology: A Literature Review,'' *Technometrics*, 8, 571-590.

88. Hill, W. J. and Wiles, R. A., (1975) ''Plant Experimentation (PLEX),'' *Journal of Quality Technology*, 7, 115-122.

89. Hunter, J. S., (1985) ''Statistical Design Applied to Product Design,'' *Journal of Quality Technology*, 17, 210-221.

90. Hunter, W. G. and Kittrell, J. R., (1966) ''Evolutionary Operation: A Review,'' *Technometrics*, 8, 389-397.

91. Hunter, W. G., (1981) ''The Practice of Statistics: The Real World is an Idea Whose Time Has Come,'' *The American Statistician*, 35, 72-76.

92. Ilumoka, A. and Spence, R., (1988) "Parameter Tolerance Design for Electrical Circuits," *Quality and Reliability Engineering International*, 4, 87-94.

93. Iman, R. L. and Conover, W. J., (1980) "Small Sample Sensitivity Analysis Techniques for Computer Models, With Application to Risk Assessment (with discussion)," *Communications in Statistics - Theory and Methods*, A9, 1749-1874.

94. Iman, R. L. and Conover, W. J., (1982) "A Distribution-Free Approach to Inducing Rank Correlation Among Input Variables," *Communications in Statistics- Simulation and Computation*, 11, 311-334.

95. Iman, R. L. and Helton, J. C., (1988) "An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models," *Risk Analysis*, 8, 71-90.

96. Jebb, A. and Wynn, H. P., (1989) "Robust Engineering Design Post-Taguchi," *Phil. Trans. R. Soc. London A*, 327, 605-616.

97. Jensen, S. T., Johansen, S., and Lauritzen, S. L., (1991) "Globally Convergent Algorithms for Maximizing a Likelihood," *Biometrika*, 78, 867-878.

98. Juan, J. and Pena, D., (1992) "A Simple Method to Identify Significant Effects in Unreplicated Two-Level Factorial Designs," *Communications in Statistics - Theory and Methods*, 21, 1383-1404.

99. Juran, J. M., *Quality Control Handbook*, Third Edition, McGraw Hill, New York.

100. Juran, J. M. and Gryna, F. M., *Quality Planning and Analysis*.

101. Kacker, R. N., (1985) "Off-line Quality Control, Parameter Design and the Taguchi Method (with discussion)," *Journal of Quality Technology*, 17, 176-209.

102. Kacker, R. N. and Shoemaker, A. C., (1986) "Robust Design: A Cost-Effective Method for Improving Maunfacturing Processes," *AT&T Technical Journal*, 65, 39-50.

103. Kacker, R. N. and Tsui, K.-L., (1990) "Interaction Graphs: Graphical Aids for Planning Experiments," *Journal of Quality Technology*, 22, 1-14.

104. Kacker, R. N., Lagergren, E. S., and Filliben, J. J., (1991) "Taguchi's Fixed-Element Arrays are Fractional Factorials," *Journal of Quality*

Technology, 23, 107-116.

105. Khuri, A. I. and Cornell, J. A., (1987) *Response Surfaces: Design and Analysis*, Marcel Dekker, New York, N.Y..

106. Kiefer, J., (1959) "Optimum Experimental Designs (with discussion)," *Journal of the Royal Statistical Society - Series B*, 21, 272-319.

107. Kitanidis, P. K., (1985) "Minimum-Variance Unbiased Quadratic Estimation of Covariance of Regionalized Variables," *Mathematical Geology*, 17, 195-208.

108. Kusaba, I., (1988) "Statistical Methods in Japanese Quality Control," *Societas Qualitatis*, 2, 1-3.

109. Lehmann, E. L., (1990) "Model Specification: The Views of Fisher and Neyman, and Later Developments," *Statistical Science*, 5, 160-168.

110. Lenth, R. V., (1989) "Quick and Easy Analysis of Unreplicated Factorials," *Technometrics*, 31, 469-474.

111. León, R. V., Shoemaker, A. C., and Kacker, R. N., (1987) "Performance Measures Independent of Adjustment," *Technometrics*, 29, 253-265.

112. Lewis, S., (1982) "Generators for Factorial Experiments," *Journal of Statistical Planning and Inference*, 6, 59-64.

113. Lin, D. K. J. and Draper, N. R. , (1992) "Projection Properties of Placett and Burman Designs," *Technometrics*, 34, 423-428.

114. Liu, W., (1992) "Predictive Screening," *Communications in Statistics-Theory & Methods*, 21, 2349-2366.

115. Logothetis, N., (1987) "Off-line Quality Control with Initial Exploration of Data," *GEC Journal of Research*, 5, 40-48.

116. Logothetis, N., (1990) "Box-Cox Transformations and the Taguchi Method," *Applied Statistics*, 39, 31-48.

117. Lotwick, H. W. and Silverman, B. W., (1982) "Methods for Analysing Spatial Processes of Several Types of Points," *Journal of the Royal Statistical Society - Series B*, 44, 406-413.

118. Low, K. K. and Director, S. W., (1988) "An Efficient Macromodeling Approach for Statistical IC Process Design," in *IEEE Int. Conf. on Computer-Aided Design*, pp. 16-19, Santa Clara, CA.

119. Lowe, C. W., (1974) "Evolutionary Operation in Action," *Applied Statistics*, 23, 218-226.

120. Mager, P. P., (1982) "Multivariate Response Surface Optimization," *Pharmizie*, 37, 658-660.

121. Mak, T. K., (1992) "Estimation of Parameters in Heteroscedastic Linear Models," *Journal of the Royal Statistical Society - Series B*, 54, 649-655.

122. Mardia, K. V., (1980) "Some Statistical Inference Problems in Kriging II," in *Proceedings of the 26th International Geological Congress, Sciences de la Terre, Serie "Informatique Ge´ologique*, vol. 15, pp. 113-131.

123. Mardia, K. V. and Marshall, R. J. , (1984) "Maximum Likelihood Estimation of Models for Residual Covariance in Spatial Regression," *Biometrika*, 71, 135-146.

124. Mardia, K. V. and Watkins, A. J., (1989) "On Multimodality of the Likelihood in the Spatial Linear Model," *Biometrika*, 76, 289-295.

125. Mardia, K. V. and Dryden, I. L., (1989) "The Statistical Analysis of Shape Data," *Biometrika*, 76, 271-281.

126. Marshall, R. J. and Mardia, K. V., (1985) "Minimum Norm Quadratic Estimation of Components of Spatial Covariance," *Mathematical Geology*, 17, 517-525.

127. Masri, S. F., Bekey, G. A., and Safford, F. B., (1976) "An Adaptive Random Search Method for Identification of Large-Scale Nonlinear Systems," *Proceedings 4th IFAC Symposium on Identification and System Parameter Estimation*, 246-255.

128. Matheron, G., (1963) "Principles of Geostatistics," *Economic Geology*, 58, 1246-1266.

129. McKay, M. D., Conover, W. J., and Beckman, R. J., (1979) "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, 21, 239-245.

130. McPherson, G., (1989) "The Scientists View of Statistics - A Neglected Area (with comments)," *Journal of the Royal Statistical Society-A*, 152, 221-240.

131. Mead, R., (1990) "The Non-Orthongonal Design of Experiments (with discussion)," *Journal of the Royal Statistical Society-A*, 153, 151-202.

132. Mee, R. W., (1990) "An Improved Procedure for Screening Based on a Correlated, Normally Distributed Variable," *Technometrics*, 32, 331-338.

133. Mitchell, T. J., (1974) "An Algorithm for the Construction of D-optimal Experimental Designs," *Technometrics*, 16, 203-210.

134. Mitchell, T. J., (1974) "Computer Construction of 'D-optimal' First-order Designs," *Technometrics*, 16, 211-220.

135. Mitchell, T. J. and Bayne, C. K., (1978) "D-optimal Fractions of Three-level Factorial Designs," *Technometrics*, 20, 369-380.

136. Monrod, H. and Bailey, R. A., (1992) "Pseudofactors Normal Use to Improve Design and Facilitate Analysis," *Applied Statistics*, 41, 317-336.

137. Montgomery, D. C., (1991) "Using Fractional Factorial Designs for Robust Design Process Development," *Quality Engineering*, 3, 193-205.

138. Morris, M. D. and Mitchell, T. J., (1983) "Two-level Multifactor Designs for Detecting the Presences of Interaction," *Technometrics*, 25, 345-355.

139. Morris, M. D., (1987) "Two-Stage Factor Screening Procedures Using Multiple Grouping Assignments," *Communications in Statistics - Theory and Methods*, A16, 3051-3067.

140. Morris, M. D., (1991) "Factorial Sampling Plans for Preliminary Computational Experiments," *Technometrics*, 33, 161-174.

141. Morris, M. D., Mitchell, T. J., and Ylvisaker, D., (1991) "Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction," ORNL Technical Report (ORNL/TM-11699).

142. Morrison, S. J., (1957) "The Study of Variability in Engineering Design," *Applied Statistics*, 6, 133-138.

143. Myers, R. H. and Carter, W. H. Jr., (1973) "Response Surface Techniques for Dual Response Systems," *Technometrics*, 15, 301-317.

144. Myers, R. H., Khuri, A. I., and Carter, W. H., (1989) "Response Surface Methodology: 1966-1988," *Technometrics*, 31, 137-157.

145. Myers, R. H., (1991) "Response Surface Methodology in Quality Improvement," *Communications in Statistics-Theory and Methods*, 20, 457-476.

146. Myers, R. H., Vining, G. G., Giovannitti-Jensen, A., and Myers, Sharon L., (1992) "Variance Dispersion Properties of Second-Order Response Surface Designs," *Journal of Quality Technology*, 24, 1-11.

147. Myers, R. H., Khuri, A. I., and Vining, G., (1992) "Response Surface Alternatives to the Taguchi Robust Parameter Design Approach," *The American Statistician*, 46, 131-139.

148. Nair, V. N. and Pregibon, D., (1986) "A Data Analysis Strategy for Quality Engineering Experiments," *AT&T Technical Journal*, 65, 73-84.

149. Nair, V. N. and Pregibon, D., (1988) "Analyzing Dispersion Effects from Replicated Factorial Experiments," *Technometrics*, 30, 247-257.

150. Nair, V. N., (1992) "Taguchi's Parameter Design: A Panel Discussion," *Technometrics*, 34, 127-161.

151. Nazaret, W. and Liu, L., (1990) "Computer Aided Design for Quality(CADQ)," *AT&T Technical Journal*, 69, 46-60.

152. Nelder, J. A. and Mead, R., (1965) "A Simplex Method for Function Minimization," *Computer Journal*, 7, 308-313.

153. Nelder, J. A. and Pregibon, D., (1987) "An Extended Quasi-likelihood Function," *Biometrika*, 74, 221-232.

154. Nelder, J. A. and Lee, Y., (1991) "Generalized Linear Models for the Analysis of Taguchi-type Experiments," *Applied Stochastic Models and Data Analysis*, 7, 107-120.

155. Nelder, J. A. and Lee, Y., (1992) "Likelihood, Quasi-likelihood and Psuedo-likelihood: Some Comparisons," *Journal of the Royal Statistical Society - Series B*, 54, 273-284.

156. O'Hagan, A., (1978) "Curve Fitting and Optimal Design for Prediction (with discussion)," *Journal of the Royal Statistical Society - Series B*, 40, 1-41.

157. Owen, A. B., (1991) "Orthogonal Arrays for Computer Experiments, Integration and Visualization," *Technical Report, University of Stanford.*

158. Owen, A. B., (1992) "Lattice Sampling Revisited: Monte Carlo Variance of Means Over Randomized Orthogonal Arrays," Stanford University Technical Report.

159. Owen, A. B., (1992) "A Central Limit Theorem for Latin Hypercube Sampling," *Journal of the Royal Statistical Society - Series B*, 54, 541-552.

160. Patterson, H. D., (1954) "The Errors of Lattice Sampling," *Journal of the Royal Statistical Society - Series B*, 16, 140-149.

161. Patterson, H. D., (1976) "Generation of Factorial Designs," *Journal of the Royal Statistical Society - Series B*, 38, 175-179.

162. Phadke, M. S., (1982) "Quality Engineering Using Design of Experiments," *Proceedings of the section on Statistical Education, ASA*, 11-20.

163. Phadke, M. S., (1986) "Design Optimization Case Studies," *AT&T Technical Journal*, 65, 51-68.

164. Phadke, M. S. and Dehnad, K., (1988) "Optimization of Product and Process Design for Quality and Cost," *Quality and Reliability Engineering International*, 4, 105-112.

165. Phadke, M. S., (1989) *Quality Engineering Using Robust Design,* Prentice Hall, New Jersey.

166. Plackett, R. L. and Burman, J. P., (1946) "The Design of Optimum Multifactorial Experiments," *Biometrika*, 33, 305-325.

167. Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., (1986) *Numerical Recipes,* Cambridge University Press, Cambridge.

168. Pronzato, L., Walter, E., Venot, A., and Lebruchec, J., (1984) "A General-Purpose Global Optimizer: Implementation and Applications," *Mathematics and Computers in Simulation*, 26, 412-422.

169. Ripley, B. D., (1977) "Modelling Spatial Pattern (with discussion)," *Journal of the Royal Statistical Society - Series B*, 39, 172-212.

170. Rosenblatt, A. and Watson, G. F., (1991) "Concurrent Engineering," *IEEE Spectrum*, July, 22-37.

171. Sacks, J. and Schiller, S., "Spatial Designs," in *Statistical Decision Theory and Related Topics IV*, ed. Berger, J. O., vol. 2, pp. 385-399, Springer, New York.

172. Sacks, J. and Ylvisaker, D., (1970) "Statistical Designs and Integral Approximation," in *Proc. 12th Bien. Sem. Canad. Math. Congress*, ed. R. Pyke, pp. 115-136, Canadian Mathematical Congress, Montreal.

173. Sacks, J. and Ylvisaker, D., (1978) "Linear Estimation for Approximately Linear Models," *Annals of Statistics*, 6, 1122-1137.

174. Sacks, J., Schiller, S. B., and Welch, W. J., (1989) "Design for Computer Experiments," *Technometrics*, 31, 41-47.

175. Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P., (1989) "Design and Analysis of Computer Experiments," *Statistical Science*, 4, 409-435.

176. Saunders, I. W. and Eccleston, J. A., (1992) "Experimental Design for Continuous Processes," *Australian Journal of Statistics*, 34, 77-90.

177. Schneider, H. and Tang, K., (1990) "Adaptive Procedures for the 2-Stage Group Testing Problem Based on Prior Distributions and Costs," *Technometrics*, 32, 397-406.

178. Shilling, M. F., (1992) "Spatial Design When the Observations are Correlated," *Communications in Statistics - Simulation and Computation*, 21, 243-268.

179. Shoemaker, A. C. and Kackar, R. N., (1988) "A Methodology for Planning Experiments in Robust Product and Process Design," *Quality and Reliability Engineering International*, 4, 95-103.

180. Shoemaker, A. C., Tsui, K.-L., and Wu, C. F. J., (1991) "Economical Experimentation Methods for Robust Design," *Technometrics*, 33, 415-428.

181. Smith, D. M., (1991) "AS268-All Possible Subset Regressions Using the QR Decomposition," *Applied Statistics*, 40, 502-513.

182. Smyth, G. K., (1989) "Generalized Linear Models with Varying Dispersion," *Journal of the Royal Statistical Society - Series B*, 51, 47-60.

183. Stein, A. and Corsten, L. C. A., (1991) "Universal Kriging and Cokriging as a Regression Procedure," *Biometrics*, 47, 575-588.

184. Stein, M., (1987) "Large Sample Properties of Simulations Using Latin Hypercube Sampling," *Technometrics*, 29, 143-151.

185. Stein, M., (1989) "Asymptotic Distribution of Minimum Norm Quadratic Estimators of the Covariance Function of a Gaussian Random Field," *Annals of Statistics*, 17, 980-1000.

186. Stein, M., (1990) "Uniform Asymptotic Optimality of Linear Predictions of a Random Field using an Incorrect Second-order Structure," *Annals of Statistics*, 18, 850-872.

187. Stein, M., (1990) "Bounds on the Efficiency of Linear Predictions using an Incorrect Covariance Function," *Annals of Statistics*, 18, 1116-1138.

188. Stein, M., (1990) "A Comparison of Generalized Cross Validation and Modified Maximum Likelihood for Estimating the Parameters of a

Stochastic Process," *Annals of Statistics*, 18, 1139-1157.

189. Stein, M., (1991) "A Kernel Approximation to the Kriging Predictor of a Spatial Process," *Annals of the Institute of Statistical Mathematics*, 43, 61-75.

190. Steinberg, D. M. and Hunter, W. G., (1984) "Experimental Design: Review and Comment (with discussion)," *Technometrics*, 26, 71-130.

191. Steinberg, D. M., (1985) "Model Robust Response Surface Designs: Scaling Two-level Factorials," *Biometrika*, 72, 513-526.

192. Taguchi, G. and Y. I. Wu, (1979) *Introduction to Off-Line Quality Control,* Central Japan Quality Control Association.

193. Taguchi, G., (1986) *Introduction to Quality Engineering*. Asian Productivity Organization, Tokyo.

194. Taguchi, G., (1989) "Experimental Design for Dynamic Characteristics," *ASI Journal*, 2, 7-24.

195. Tjur, T., (1991) "Block Designs and Electrical Networks," *Annals of Statistics*, 19, 1010-1027.

196. Tribus, M. and Szonyi, G., (1989) "An Alternative View of the Taguchi Approach," *Quality Progress*, 22, 46-52.

197. Tsui, K.-L., (1988) "Strategies for Planning Experiments Using Orthogonal Array and Confounding Tables," *Quality and Reliability Engineering International*, 4, 113-122.

198. Turiel, T. P., (1988) "A FORTRAN Program to Generate Fractional Factorial Experiments," *Journal of Quality Technology*, 20, 63-72.

199. Vecchia, A. V., (1988) "Estimation and Model Identification for Continuous Spatial Process," *Journal of the Royal Statistical Society - Series B*, 50, 297-312.

200. Vecchia, A. V., (1992) "A New Method of Prediction for Spatial Regression Models," *Journal of the Royal Statistical Society - Series B*, 54, 813-830.

201. Vining, G. G. and R. H. Myers, (1990) "Combining Taguchi and Response Surface Philosophies: A Dual Response Approach," *Journal of Quality Technology*, 22, 38-45.

202. Vining, G. G. and Myers, R. H., (1991) "A Graphical Approach for Evaluating Response Surface Designs in Terms of the Mean Squared Error of Prediction," *Technometrics*, 33, 315-326.

203. Voss, D. T. and Dean, A. M., (1987) "Equivalence of Methods of Confounding in Asymmetrical Single Replicate Factorial Designs," *Annals of Statistics*, 15, 376-384.

204. Wang, J. C. and Wu, C. F. J., (1991) "An Approach to the Construction of Asymmetrical Orthogonal Arrays," *Journal of the American Statistical Association*, 86, 450-456.

205. Wang, J. C. and Wu, C. F. J., (1992) "Nearly Orthogonal Arrays with Mixed Levels and Small Runs," *Technometrics*, 34, 409-422.

206. Wang, W. and Lawson, J., (1988) "Estimating the Error of an Effect in Unreplicated 2-Level Fractional Factorial Designs," *Quality and Reliability Engineering International*, 4, 189-192.

207. Warnes, J. J. and Ripley, B. D., (1987) "Problems with Likelihood Estimation of Covariance Functions of Spatial Gaussian Processes," *Biometrika*, 74, 640-642.

208. Watkins, A. J. and Mardia, K. V., (1992) "Maximum Likelihood Estimation and Prediction Mean Square Error in the Spatial Linear Model," *Journal of Applied Statistics*, 19, 49-60.

209. Welch, W. J., (1982) "Branch-and-bound Search for Experimental Designs Based on D Optimality and Other Criteria," *Technometrics*, 24, 41-48.

210. Welch, W. J., (1983) "A Mean Squared Error Criterion for the Design of Experiments," *Biometrika*, 70, 205-213.

211. Welch, W. J., Yu, Tat-Kuan, Kang, S. M., and Sacks, J., (1990) "Computer Experiments for Quality Control by Parameter Design," *Journal of Quality Technology*, 22, 15-22.

212. Welch, W. J. and Sacks, J., (1991) "A System for Quality Improvement Via Computer Experiments," *Communications in Statistics-Theory and Methods*, 20, 477-496.

213. Welch, W. J., Buck, R. J., Sacks, J., Wynn, H. P., Mitchell, T. J., and Morris, M. D., (1992) "Screening, Predicting, and Computer Experiments," *Technometrics*, 34, 15-25.

214. Wong, W.-K., (1992) "A Unified Approach to the Construction of Minimax Designs," *Biometrika*, 79, 611-620.

215. Wu, C. F. J., Mao, S. S., and Ma, F. S., (1979) "SEL: A Search Method Based on Orthogonal Arrays," in *Statistical Design and Analysis of Industrial Experiments*, ed. S. Ghosh, pp. 279-310, Marcel Dekker.

216. Wu, C. F. J. and Chen, Youyi, (1992) "A Graph-Aided Method for Planning Two-Level Experiments when Certain Interactions are Important.," *Technometrics*, 34, 162-175.

217. Wynn, H. P., Sivaloganathan, S., Buck, R. J., and Lewis, S. M., (1992) "Generate: An Algorithm for the Computer Generation of Orthogonal Arrays with Specific Alias Structure," Engineering Design and Quality Centre Technical Report #30.

218. Yakowitz, S. J. and Szidarovsky, F., (1985) "A Comparison of Kriging and Nonparametric Regression Methods," *Journal of Multivariate Analysis*, 16, 21-53.

219. Yang, J. and Kushner, (1991) "A Monte Carlo Method for Sensitivity Analysis and Parameter Optimization of Nonlinear Stochastic Systems," *SIAM Journal on Control and Optimization*, 29, 1216-1249.

220. Ylvisaker, D., (1975) "Designs on Random Fields," in *A Survey of Statistical Design and Linear Models*, ed. J. N. Srivastava, pp. 593-607, North-Holland, Amsterdam.

221. Ylvisaker, D., (1987) "Prediction and Design," *Annals of Statistics*, 15, 1-19.

222. Young, D. L., Teplik, J., Weed, H. D., Tracht, N. T., and Alvarez, A. R., (1991) "Application of Statistical Design and Response Surface Methods to Computer-Aided VLSI Device Design II: Desirability Functions and Taguchi Methods," *IEEE Transactions on Computer-aided Design*, 10, 103-115.

223. Yu, T. K., Kang, S. M., Sacks, J., and Welch, W. J., (1991) "Parametric Yield Optimization of CMOS Analogue Circuits by Quadratic Statistical Circuit Performance Models," *International Journal of Circuit Theory and Applications*, 19, 579-592.

224. Zacks, S., (1991) "Inference Based on Taguchi's Saturated Designs," *Communications in Statistics-Theory and Methods*, 20, 497-510.

225. Zahn, D. A., (1975) ''Modifications of and Revised Critical Values for the Half-Normal Plot,'' *Technometrics*, 17, 189-200.

226. Zahn, D. A., (1975) ''An Empirical Study of the Half-Normal Plot,'' *Technometrics*, 17, 201-211.

227. Zahn, D. A. and Isenberg, D. J., (1983) ''Nonstatistical Aspects of Statistical Consulting,'' *The American Statistician*, 37, 297-302.

228. Zhigljavsky, A. A., (1991) *Theory of Global Random Search*. Kluwer Academic Publishers.

229. Zimmerman, D. L., (1989) ''Computationally Exploitable Structure of Covariance Matrices and Generalized Covariance Matrices in Spatial Models,'' *J. Statist. Comput. Simul.*, 32, 1-15.

230. Zimmerman, D. L. and Cressie, N., (1991) ''Mean Squared Prediction Error in the Spatial Linear Model with Estimated Covariance Parameters,'' *Annals of the Institute of Statistical Mathematics*, 43, 27-43.

231. Zimmerman, D. L. and Zimmerman, M. B., (1991) ''A Comparison of Spatial Semivariogram Estimators and Corresponding Ordinary Kriging Predictors,'' *Technometrics*, 33, 77-92.

232. Zimmerman, D. L. and Harville, D. A., (1991) ''A Random Field Approach to the Analysis of Field-Plot Experiments and Other Spatial Experiments,'' *Biometrics*, 47, 223-240.