Open Research Online

The Open University's repository of research publications
and other research outputs

Computer algebra techniques in object-oriented mathematical modelling.

Thesis

How to cite:

Mitic, Peter (1999). Computer algebra techniques in object-oriented mathematical modelling. PhD thesis. The Open University.

For guidance on citations see FAQs.

© 1998 Peter Mitic

Version: Version of Record

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online's data policy on reuse of materials please consult the policies page.

oro.open.ac.uk

UNRESTRICTED

Computer Algebra Techniques in Object-Oriented Mathematical Modelling

Peter Mitic

Thesis presented for the Degree of Doctor of Philosophy
to the Faculty of Mathematics and Computer Science,
Discipline of Mathematics
The Open University

May 1999

DATE OF SUBMISSION: 26 NOVEMBER 1998
DATE OF AWARD: 17 MAY 1999

ProQuest Number: 27696813

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 27696813

Published by ProQuest LLC (2019). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Acknowledgments

I would like to thank several people at the Open University who have made this thesis possible. My supervisors, Peter Thomas and Darrel Ince, guided it over a long eight year period. Karen Vines provided invaluable advice on the statistical aspects and Paula Cole did a great deal of administrative work.

Many other people have made their contributions. I am particularly grateful to Ian Nicol, who did the bulk of the final proof reading. Others have made comments, provided advice, given references, supplied clues, read extracts or have simply given encouragement. These small things are important: they show how small steps fit into an overall scheme. In particular, I am very grateful to everyone who appreciated this work for what it is, and did not encourage deviation. My wife has fed me, done the washing, mowed the lawn and generally organised things. She claims that an average of three children have been present for the duration, but this cannot be mathematically correct since the youngest is now five. Perhaps there were others... Motivations for a long-term project are sometimes hard to examine, but perhaps my daughter summed it up nicely. She said "If you want to do something, just do it": not bad for a five-year-old. She might be surprised if she reads this and understands the implications of that statement, but academic pursuit for its own sake seems to be unfashionable these days. I hope that things will change.

The Open University
28 JUL 1999
Library
DONATION
T 511 8
C

Contents

	Page
Acknowledgments	2
Contents	3
Abstract	5
0. Introduction and Overview of the Thesis	7
1. Mathematical Modelling Methodologies	13
2. A Review of Computer Algebra Algorithm Development and its impact on Mathematical Modelling	55
3. Teaching and learning Mathematics and Modelling by Computer	77
4. A Summary of Research problems arising from Literature Reviews	105
5. The Case for Object-Oriented Modelling	111
6. A new technique and methodology for Mathematical Modelling	141
7. *AMK*: Object-Oriented software for Particle Mechanics	187
8. Front-end Support for the modelling cycle	217
9. Validation	239
10. Conclusion and Further Research	265
Appendices	281
References	345

Thesis Abstract

This thesis proposes a rigorous object-oriented methodology, supported by computer algebra software, to generate and relate features in a mathematical model. Evidence shows that there is little heuristic or theoretical research into this problem from any of the three principal modelling methodologies: 'case study', 'scenario' and 'generic'. This thesis comprises two other major strands: applications of computer algebra software and the efficacy of symbolic computation in teaching and learning. Developing the principal algorithms in computer algebra has sometimes been done at the expense of ease of use. Developers have also not concentrated on integrating an algebra engine into other software. A thorough review of quantitative studies in teaching and learning mathematics highlights a serious difficulty in measuring the effect of using computer algebra. This arises because of the disparate nature of learning with and without a computer.

This research tackles relationship formulation by casting the problem domain into object-oriented terms and building an appropriate class hierarchy. This capitalises on the fact that specific problems are instances of generic problems involving prototype physical objects. The computer algebra facilitates calculus operations and algebraic manipulation. In conjunction, I develop an object-oriented design methodology applicable to small-scale mathematical modelling. An object model modifies the generic modelling cycle. This allows relationships between features in the mathematical model to be generated automatically. The software is validated by quantifying the benefits of using the object-oriented techniques, and the results are statistically significant.

The principal problem domain considered is Newtonian particle mechanics. Although the Newtonian axioms form a firm basis for a mathematical description of interactions between physical objects, applying them within a particular modelling context can cause problems. The goal is to produce an equation of motion. Applications to other contexts are also demonstrated.

This research is significant because it formalises feature and equation-generation in a novel way. A new modelling methodology ensures that this crucial stage in the modelling cycle is given priority and automated.

Chapter 0

Introduction and Overview of the Thesis

0.0 Introduction

The work behind this thesis covers a period during which there has been enormous change in computing. During the 1990s symbolic computation has moved from mainframe computers to micro-computers to hand held calculators. Many people have been reluctant to accept these changes, but they are here to stay. The task is therefore to discover ways of using the available tools to their best advantage, and this thesis must be seen in that context.

0.1 Previous Papers

Two papers, (Mitic 95A) and (Mitic 95B), cover aspects of original work for this thesis on object models and user interfaces.

0.2 Problem statement

The motivation behind this work lies in teaching a first course in mathematical modelling to undergraduate students, and in teaching mechanics to younger students. It appeared that, despite efforts to ease the modelling process, there were persistent problem-solving strategy difficulties. Of these, it seemed that a significant number could be classified as 'unable to start'. This was because, having completed preliminary stages, students could not easily identify the salient features in a mathematical model. Even if this stage was done successfully, students were sometimes unable to form relationships between the features to produce a model. The problem was therefore to devise a way of finding features, and of formulating a model by finding relationships between them. Although finding features and formulating relations between them are interlinked, this thesis concentrates more on formulating relations. The aim was to automate this process. Computer algebra software is necessarily involved in this process because algebraic computation is fundamental to mathematics (and hence to modelling) in a way that numeric computation is not. An algebraic model forces functional forms to be explicit, so that functional variation is equally explicit. Routine algebra and calculus operations are needed to do this.

0.3 Literature reviews

Chapters 1, 2, 3 and 5 contain literature reviews covering distinct areas.

Chapter 1 contains an analysis of three common mathematical modelling strategies, with a discussion of the strengths and weaknesses of each. The series of ICTMA[1] conferences is a principal source of information for two of these: 'scenario' and 'generic' modelling. These, and other sources, indicate that some basic modelling issues (as specified above) are not being addressed. Views on which methodology to use are somewhat polarised, and concentrate on theoretical issues, course content and presentation, presentation of results and other issues related to modelling. The main sources of literature for the third modelling strand, 'case-study' modelling, comes from standard texts on modelling. From these a surprisingly small number of modelling contexts can be identified, but there is, again, a lack of discussion on relationship formulation. The absence of extensive discussion of relationship formulation and identification of features indicates the gap in knowledge which this thesis fills.

The review in Chapter 2 traces the principal stages in algorithm development of computer algebra software from its introduction in the 1960s to the present. Design decisions for computer algebra software created barriers for use in mathematical education, and also in modelling. Tracing algorithm development explains why computer algebra was prominent in mathematical research long before educational and modelling applications were envisaged. Papers from the 1970s and 1980s show that user-friendly interfaces and easy manipulation of expressions were not a first priority of developers. There is therefore a gap in developing computer algebra tools for modelling, and the reviews in Chapter 1 show that there are only a few computer algebra applications in modelling.

Chapter 3 traces ideas suggested in Chapter 2 by presenting a summary of the general issues which can influence learning of mathematics, with particular emphasis on the use of computer algebra laboratories. Combining the results of other studies, I present the most comprehensive review to date of quantitative studies of the efficacy of learning mathematics. I challenge an informal view that use of computer algebra laboratories is beneficial, and show that computer laboratories have an overall neutral effect on learning. Reviews of the statistical results from research papers highlight a problem

which has a major bearing on those studies and on this thesis. This is that using computer algebra is so fundamentally different in approach to a traditional presentation that any direct comparison is ill-founded. It affects the validation technique used in Chapter 9.

Chapter 4 summarises the research problems identified from the previous three chapters, and identifies research tasks for this thesis. The main task is to find a formal way of linking features in a model, and to then find an appropriate validation method.

Chapter 5 states the case for applying an object-oriented solution to the problems posed in this thesis. I aim to show that an object-oriented solution is feasible and desirable, and point out that this type of consideration is rarely justified in many contexts. I then identify what 'new' concepts are important for an object-oriented analysis of a mathematical modelling domain. A review of the principal object-oriented design methodologies currently in use indicates that none are wholly suitable for the task of producing an object model for a mathematical modelling domain. It is possible to identify some useful ideas from them, and these are enhanced in the next chapter.

0.4 Solution by Object-Oriented Analysis and Design

In Chapter 6 I develop an object-oriented analysis methodology which is more appropriate for mathematical modelling contexts. It capitalises on standard practice in mathematical modelling, and uses specific mathematical processes and constructions. The principal task is to identify physical objects, which are abstracted as 'nouns with properties'. I suggest two ways of doing this. The first uses a diagram, which would be drawn as a normal part of the modelling process, and abstracts objects from the diagram. In doing so, I develop a modelling heuristic, the *Principle of Adjacency*, which plays a significant role in determining which elements in the problem domain can be sensibly related. A diagram alone is often insufficient to define a complete model. It is therefore backed up by considering the axioms and heuristics of the problem domain, and ideas are imported from established modelling methodologies to do this. These two strands give the *Diagram-Axiom* methodology, which is my new approach to modelling. Outline examples show how the *Diagram-Axiom* methodology may be applied to

[1] International Conference on the Teaching of Mathematical modelling and Applications

several modelling contexts, pointing out cases where it is not satisfactory, as well as more successful cases.

Chapter 7 contains a more rigorous treatment of the ideas presented so far by applying object-oriented modelling to the context of Newtonian Particle mechanics. The Mathematica software which supports this stresses throughout how objects in the problem domain interact with each other. This software is the concrete manifestation of the solution to the problem posed by this thesis. The software combines two important features. The first is the object model. The second is a methodology for using the object model. The software contains procedures for implementing a modelling cycle that involves constructing objects, linking them and invoking class methods until an equation of motion results. I call this the *Construct→Link→Invoke_Methods* process. Several examples of how models are developed illustrate the power and scope of the methodology. Some programming proved to be tricky, and I discuss successes, failures and some alternative strategies. Some difficulties arise when using the software, and these are addressed in the next chapter. I also briefly mention extending the principles developed in this chapter to other modelling contexts, and demonstrate that this can be done.

In Chapter 8 I attempt to solve two problems which originate from the software described in Chapter 7. The first is that there is nothing to force the user to follow a rigid *Construct→Link→Invoke_Methods* process. This diminishes its value. The second is that composing the Mathematica inputs is syntactically complex. This chapter illustrates how a front-end for the Mathematica software constrains the user to use the methodology rigorously. In doing so it supplies an algorithm for the modelling process, automates production of links between objects, and hides the awkward syntax required. A second interface makes direct use of the *Principle of Adjacency* and its converse by building a model by manipulating a diagram on screen.

Chapter 9 serves two purposes. First it provides empirical justification, using modelling assignments from 61 students, for the claim that relating features in a model presents a problem. An analysis of errors made shows that 35% of those errors are in relationship formulation. Of these, 19% were unable to start the process, and a further 71% made material errors in relationship formulation. Furthermore, evidence shows that there is a lack of research into assessment of the model formulation stage in modelling. The

second task of this chapter is to address the problem highlighted in Chapter 3. The nature of field trials would invalidate a statistical analysis for validating the modelling methodology and software of this thesis. The strategy adopted here is to give an account of how an object-oriented approach would have been beneficial in alleviating difficulties in the 61 student assignments. The criteria for judging the effect of using object-oriented principles are stringent, but the results are still statistically significant.

The last chapter assesses what further research could result from the ideas in this thesis. Some of these relate to developments of the *Diagram-Axiom* methodology: how to create a class library for a new modelling context quickly and easily, and how to cast the components of the methodology into a workable algorithm. Others relate to the software components: developing the icon-driven front-end software, and finding efficient implementations of the object model. I also reiterate the difficulties of evaluating software in use and comparing it with 'traditional' methods. Lastly, I suggest developing some variations on object-oriented themes in different ways, using alternative software.

Chapter 1

Mathematical Modelling Methodologies

1.0 Abstract

This chapter presents an analysis of three common mathematical modelling strategies, and discusses advantages and disadvantages of each. The first is modelling by using specific examples, in which a model based on a given context is presented as a set of related equations, which are then solved. This approach can make it hard for the novice to see where the equations originated from and encourages reuse of inappropriate old models. The idea of a fixed number of context types, from which such models originate, is supported by standard texts, which subdivide "applied mathematics" into a small number of categories, each with standard models. A summary of these cases is given. The second category is "scenario modelling", in which a model is constructed around a particular mathematical technique or process. This method also discourages generalisation and is limited in scope. The third category, "generic" modelling, allows a much wider class of scenarios to be modelled. It is based on a modelling cycle in which a problem is analysed and goals set, and important features, peripheral features and variables are identified. These steps are relatively easy and the method allows some progress to be made by nearly all students. The stage where relations are established between the variables is harder, and this thesis presents empirical evidence to support the claim that a "generic" modelling cycle is insufficient to address the problem of how to form relations between variables. Furthermore, a method for quantifying and assessing the efficacy of generic model development has not yet been developed, and this makes it hard to assess preliminary steps in the cycle. The same evidence also shows that a disproportionate effort is directed towards peripheral activities in the cycle. Formation of relations between variables is a crucial step which is addressed in this thesis by an object-oriented technique which necessarily involves symbolic computation.

1.1 The purpose and scope of mathematical modelling

In this section I present some general ideas of what constitutes mathematical modelling, and identify three main methodologies for mathematical modelling.

1.1.1 Mathematical modelling: background, aims and definitions

Papers from the ICTMA (International Conference on the Teaching of Mathematical modelling of Applications - the principal mathematical modelling forum) indicate a move towards teaching mathematics through applications. In this chapter I discuss the motivations for this. Two papers in ICTMA-3 (Sloyer 89 and Blum 91) serve as general introductions to mathematical modelling in the late 1980s, and are still relevant today. The first of these is a very short introduction to topics which were then of interest to mathematicians, and hence which could usefully benefit from mathematical modelling. Sloyer makes the general point that the reason why students should learn mathematical modelling is to "be better able to understand their world". This definition would seem to be too general to be able to allow further progress as far as mathematics and modelling are concerned. Sloyer lists eleven areas of interest in modelling, eight of which have statistical origins. These may reflect his interests and are not typical of modelling topics presented in texts on modelling (discussed later in this chapter). Sloyer's paper omits topics in applied mathematics and mechanics. However, the list indicates a need to concentrate on the precise nature and purpose of the modelling process.

The article by Blum (Blum 91) is a review paper which discusses general modelling issues in much more depth. He states that mathematical modelling means applying mathematical techniques to a "real problem situation", but does not make much headway in defining terms rigorously. To attempt a rigorous definition for mathematical modelling is not particularly productive. However, the purpose of the exercise becomes clear when one considers the 4-step modelling process described in Blum's paper. The steps are:

1. produce a simplified (abstract) situation from a real situation;
2. translate data, concepts, relations, conditions and assumptions into mathematics ("mathematise");
3. use appropriate mathematical techniques to solve equations.;
4. interpret the solution and validate the model.

The idea of associating a set of equations with a set of physical objects, and also with solution methods for those equations is missing from (Blum 91), and is also missing from current mathematical modelling methodologies. Such an association introduces the idea of an object, which is central to this thesis.

Blum mentions the three principal mathematical modelling methodologies which I discuss in this thesis. He first hints at a *generic modelling methodology* which will be discussed in detail later in this chapter. Second, he uses the term *mathematise*: the process of producing a mathematical model from a real situation. This is central to the methodology which I term *scenario modelling* in this thesis. Third, he mentions the idea that an existing model may be applied to a new or different situation. This constitutes the third methodology: using specific *case studies*. Blum isolates the following as important modelling examples: income tax, elections, traffic flow, and shot putting. He also emphasises that all the examples he mentions are suitable for analysis by an iterative problem solving process, and that problem choice "is essentially a matter of looking at the examples and of preparing them". Blum does not explain this comment.

Blum's discussion of general aims for mathematics teaching and modelling is consistent with the idea of *applicable mathematics*. These aims are to describe "extra-mathematical" areas, to develop problem solving skills and analysis (although there is insufficient evidence to justify transference of mathematical skills to problem solving in general), and for its own sake, as part of a cultural heritance. Modelling skills have to be learned, and Blum suggests three ways to teach them.

1. The first is to create "meta-knowledge" by teaching a technique through a specific example.
2. The second is to develop a comprehensive understanding of concepts, although he does not say how this should be done. There is evidence from research involving computer algebra laboratories for calculus to show that problems involving understanding rather than actual computation do, indeed, aid understanding. Mayes (Mayes 97) gives a summary of many of these studies, and I discuss them in detail in Chapter 3. However, I also demonstrate in Chapter 3 that the way in which the studies have been done render the origin of improved understanding unclear.

3. The third is to use "iterative ideas of reality to prove mathematical response rigorously". This seems to amount to using intuition, which can be misleading.

In parallel with Blum's article, he and Niss have written extensively about the modelling process in (Blum 91A). They give a definition of mathematical modelling, in terms of a very simple relational meta-model. Thus, a *mathematical model* is defined as a triple (S, M, R) comprising a real problem situation S, a collection of mathematical entities M, and a mapping R from S to M. The latter constitutes the *mathematisation* process. It must be a two-way mapping because there must also be a map back to S from M. This definition is not particularly useful because all it does is to allocate symbols to any standard diagram in a modelling cycle. It says nothing about the contents of S and M and in particular says nothing about the nature of R. This very basic theme is expanded upon by Warzel (Warzel 89) in a discussion of a more sophisticated meta-model for mathematical modelling. Warzel's meta-model is also impractical because it does not provide ways of producing a mathematical model from a real situation. The software and techniques described in Chapters 6, 7 and 8 of this thesis fill this gap with a new modelling methodology and a software implementation of it.

Blum (Blum 91A) distinguishes between models which have an axiomatic basis and models which do not. He calls the former *descriptive* models and the latter *normative* models. The essential distinction between them is that normative models require value judgements when formulating features, relationships and assumptions. Descriptive models (which normally relate to physical phenomena through Newton's laws or similar) already contain the basis of a model in the form of established relationships and features (which constitute axioms). This distinction has been made by other authors (see, for example Crighton 95). It is a key feature of the modelling methodology developed in Chapter 6 of this thesis.

1.1.2 Thought processes which affect modelling

Research on cognitive processes in mathematical modelling (Lambert 89 and Lamon 97), provides evidence that domain knowledge and relationship generation are significant factors in modelling. These factors are discussed in the sections on *Generic Modelling*, and this section provides some background for that discussion.

Lambert and others (Lambert 89) provide evidence for a need for domain specific knowledge as part of a mental model. This is a significant factor that supports the research in this thesis. They develop a common conceptual model in which *beliefs* (knowledge, assumptions) relate to *metacognition* (monitoring, evaluation, control), which in turn relates to the mathematical domain and the problem domain. The latter only interact via the *metacognition* process. They conclude that such models often consist of simple, personal, rules of thumb, and that a mental model is distinct and different from any mathematical process model. They also report that difficulties have been experienced in trying to teach the metacognitive processes that contribute to modelling performance. There is little discussion in Lambert's analysis on ways to facilitate the production of the necessary mental model, other than experience.

A need for familiarity with a problem domain is supported by Lamon (Lamon 97), who proposes the idea of a Conceptual model (C-model), which consists of a system of mathematical ideas and associated processes. Her discussion proceeds by example, so that it is difficult to generalise, but she concludes that a stable C-model can be easily used and adapted in more sophisticated models, and that it is less subject to rote use than a non-stable C-model. Lamon does not address problems of how any conceptual model is derived in the first place.

1.1.3 Axiomatic and Heuristic models

In this section I assess the distinction between a model based on axiomatic principles and one based on heuristics. Chapter 6 contains examples of both types of model, formulated using the methodology which I develop in Chapter 6. The significance of the axiomatic/heuristic distinction is important in applying that methodology.

Any model based on Newton's Laws is an example of a model based on axiomatic principles. Heuristic models encapsulate any data model (for example, in economics). Crighton stresses this fundamental difference (Crighton 95), saying that many models are suspect because they have no underlying axiomatic basis. He does not mention a further problem: data models are often over-fitted and rarely have any provision for ensuring that they will perform in the future. Booss-Bavnbek (Booss-Bavnbek 91) makes the same point, but clouds the issue by arguing that Newton's Laws and similar laws (such as the Coulomb law in electrodynamics) are fundamental laws of nature and are not empirical. This is not quite true. They are based on assumptions made about the physical world, which form the axioms of the model. Experiments confirm that such models are accurate to a high degree. He also takes an extreme (and incorrect) view in arguing that numeric methods in computational fluid dynamics are inappropriate because they are fundamental to models. I view them as no more than numerical techniques.

Having established a rationale for doing mathematical modelling, I now discuss each of the current mathematical modelling methodologies in detail.

1.2 Scenario modelling

The first mathematical modelling methodology has been termed *Mathematisation* or *Scenario Modelling*. It encapsulates teaching mathematics through applications. In this section I summarise the development of and current work on this methodology. Some examples and a discussion of its advantages and disadvantages follow.

1.2.1 Scenario Modelling: historical development

Schupp (Schupp 89) gives an explanation of the historical background to mathematical modelling and mathematics education in Germany from the 19th century to the present. During this time views have swayed between a wish to include practical applications in teaching and a desire to do exactly the opposite. For teaching in the Gymnasiums, there were swings in favour of applications in 1905 (the Meraner Pläne: Meran directive) and 1922 (the revised Meran directive). Pure mathematics was favoured in the intervening periods, and neither view was supported by evidence. Mathematics in the Hauptschule from the 1950s onwards was embedded in applications, for that was their role. Schupp provides the scheme in Figure 1.1 to represent a modelling methodology in these schools. It is transcribed from Ochl's 1965 book *Rechenunterricht in der Hauptschule*. I have made it more explicit by adding the dotted polygon. This separates the 'real world' inside it from the 'mathematical world', outside it.

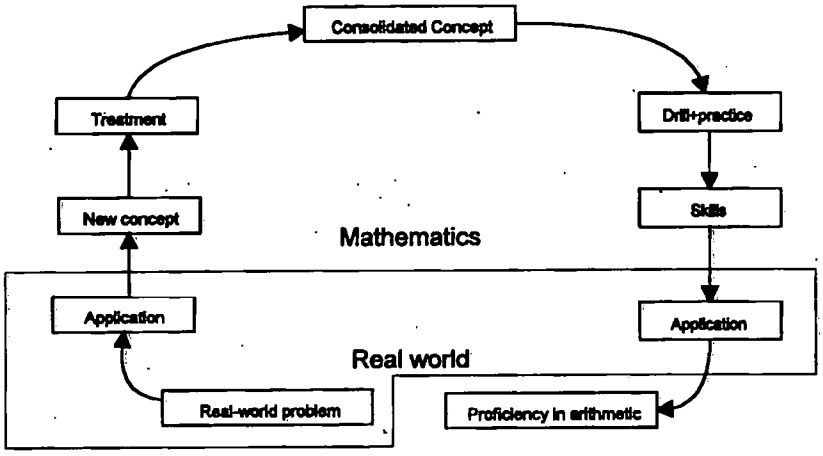


Figure 1.1

Figure 1.1 shows no indication of how a model is to be constructed. It is also curious that a real world situation appears to result in proficiency in arithmetic.

Schupp considers that this way of teaching modelling remained static in Germany for some years because of doubt about future usefulness, about what should be considered as 'normal', and a wish to have applications which were accessible to all students. Applications were therefore very uniform in style. Schupp reports two shifts away from applications, neither supported by evidence. His first claim is that during the 1960s there was a shift from arithmetic to a "set-function-structure" paradigm with the introduction of "new math", in response to scientific developments. His second claim is that in the early 1970s, a significant statistical component was introduced into the curriculum. Neither promoted applied mathematics teaching. Schupp quotes four informal studies on applied mathematics teaching, for which there was some testing in the class, although he only implies that this testing amounted to "use" with no disastrous consequences reported. Figure 1.2 shows a 1980s version of Figure 1.1. It shows little advance because it does not address the problem of how the model should be derived from the real situation.

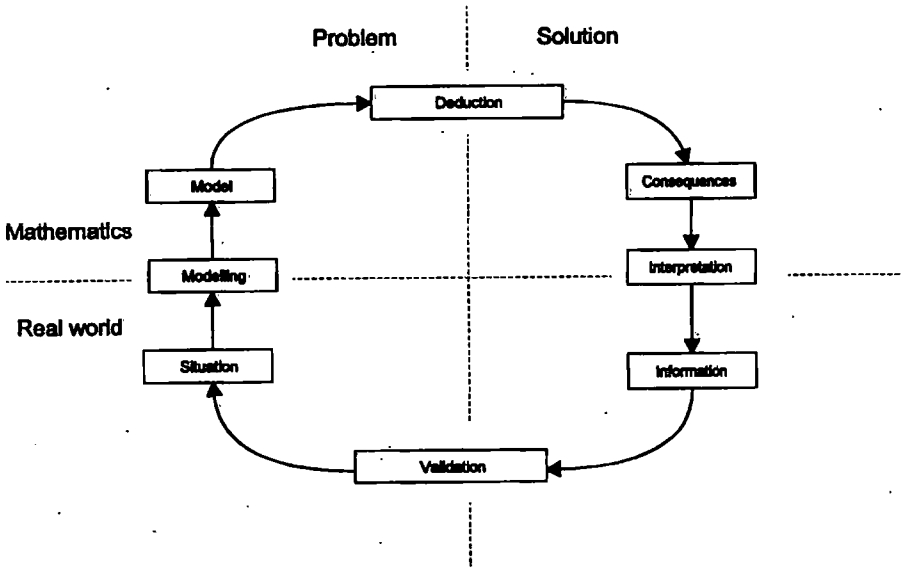


Figure 1.2

Schupp proposes four reasons for the lack of change in the 1980s:

1. teaching applied mathematics was considered ambitious;
2. few resources were available for teaching of real world problems;
3. the non-perfect nature of real life problems was considered a barrier;
4. the organisation of mathematics teaching was geared to solving small problems in context, and so did not lend itself to extended problems.

If these reasons are correct, they are consistent with a wish to teach concepts conservatively through applications. Attaching applications to techniques is the basis of scenario modelling.

Henn (Henn 97) gives a different view of the slow pace of mathematics reform in Germany. He makes the crucial point that the *Abitur* (the examination taken at age 18) means that teachers are very reluctant to change their ways of teaching or to tackle new syllabus material which is not relevant to the examination. The same applies in the UK, where innovative mathematics stays in the hands of enthusiasts. An example is the work of Watkins and Gadd (Watkins 94).

1.2.2 Scenario Modelling: current work

In this section I summarise the main characteristics of scenario modelling (in Keitel 93 and de Lange 93). I also examine discussion of issues in scenario modelling from established practitioners (Kaiser-Messmer 91 and Blum 95).

Scenario modelling formalises particular problem domains, and structures and relationships in them. Keitel (Keitel 93) gives a concise definition of mathematisation: "Extracting the appropriate concept from a concrete situation... It comprises exploration, structuring, finding the mathematical aspects, abstract concepts, discovering regularities and relations." He points out that there is a strong intuitive component in exploring. The work described in Chapters 6, 7 and 8 of this thesis aims to abstract concepts, automate the process and control the intuitive element. De Lange (de Lange 93) points out that this aim drives the way in which mathematics is taught. Teaching using mathematical structure is no longer appropriate: structure is dictated by didactics. He does not say that if teaching is done with a CAS (Computer Algebra System), a natural progression is to structure that teaching by CAS programming technique. De Lange searches for applications of mathematical techniques, but misinterprets business processes as mathematical applications. A CAS is not necessary to do manipulations in many of his examples: they can be done by any procedural programming language (C, Fortran etc.). Hence, some of his comments about implicit use of mathematics are misplaced. In practice, mathematisation reduces the size of a

problem domain and builds a realistic scenario around a particular mathematical technique. As a result, the technique drives the problem, and not vice versa. Despite this, using a scenario in conjunction with a technique does provide insight into why that technique is useful.

Kaiser-Messmer (Kaiser-Messmer 91) highlights the differences between scenario modelling and generic modelling. One significant difference is geographical. Scenario modelling developed and is widely used in German speaking areas whereas generic modelling tends to be used by English language users. There is clearly some bias on the part of the author. She readily acknowledges this by stating that she has little practical experience of generic modelling. Hence, her paper can only serve as a description of the scenario modelling method.

She states that the philosophical basis of scenario modelling is a "scientific-humanistic" principle. The idea is to enable students to "establish relations between mathematics and the real world". In order to teach mathematics in context, mathematical technique are contextualised. This aim is necessarily idealistic and she provides no empirical evidence to demonstrate that the aim is fulfilled. Research into the efficacy of scenario modelling remains a subject for further research. Kaiser-Messmer criticises some applications for being less relevant than others, but does not elaborate on these comments. She quotes the example of the Spode Group (Burghes 89), which produced an atypical course on Decision Mathematics. More representative modelling output from the Spode Group can be found in (Spode 82). Hence, her comments are misplaced.

Kaiser-Messmer distinguishes between two types of mathematisation. Applied mathematisation is used as part of a problem-solving process, and conceptual mathematisation is used as a way to introduce new concepts. Both are claimed to enhance "society and the social benefits" rather than "the individual and individual benefit". It is difficult to see how a particular modelling methodology can achieve this, and there is clearly a need for more rigorous justification. Scenario modelling may be prominent in the German speaking countries because there is a well-established tradition of "Sachrechnen"[1] for practical applied arithmetic. Similarly, modelling has to support utilitarian goals.

[1] Routine, formal exercises

Blum (Blum 95) echoes the idea expressed in (Schupp 89), that concepts should be taught conservatively through applications. Some discussion in (Blum 95) is not helpful because it is ill-defined (e.g. a problem-solving methodology), and relies on examples rather than analysis. He continues to ask old questions such as "What do applications and modelling mean?", but provides no solutions. The term "mathematise" is used without explanation of how this should be done. Perhaps this is a reflection of the "German" view that this step does not need to be taught because students acquire it naturally. I shall produce evidence in Chapter 9 to show that this is not true.

Blum thinks that certain barriers inhibit development of mathematical modelling, and the same views are also expressed in (Kaiser-Messmer 91):

- applications and modelling do not fit well into a rigid curriculum:
- it is more difficult for both learners and teachers;
- few resources are available.

These do not represent real barriers: they show a lack of innovation and awareness of courseware (e.g. Spode 82, Beilby 93).

Blum seems unaware of developments on computing in mathematics. In particular, when this paper was written (1993), DERIVE had been widely used in the United States, Germany and Austria for many years. He suggests that there are problems with using it, but does not substantiate his assertion. Research in the United States (discussed in Chapter 3) shows that devaluation of "traditional" skills is a problem, but not a significant one. Some of these skills are no longer as significant as they were. Blum considers that software was then too powerful for teaching and learning. This is no barrier to learning: it remains to find ways of using the software suitably. Many authors have done so (for example Brown 91, Kutzler 97, Berry 97, Keller 97 and Shay 97 for TI92 materials).

1.2.3 Scenario modelling examples

This section contains examples of scenario models. They show that some scenarios can be real (rather than merely plausible), but that it is easy to contrive situations that are realistic, but not real.

A 'natural' scenario

Henn (Henn 97) describes a favourite application of simple scenario modelling in the context of the German income tax system. This appears to be unique in using a monotonic increasing piecewise quadratic function of annual income x for tax payable T, rather than a piecewise linear function (as in the UK).

$$T(x)=\begin{cases} 0; & 0\le x<5617 \\ 0.19x-1067; & 5617\le x<8154 \\ 472+1900\left(\dfrac{x-8100}{10000}\right)+151.94\left(\dfrac{x-8100}{10000}\right)^2; & 8154\le x<120042 \\ 0.53x-22842; & x\ge 120042 \end{cases}$$

It is therefore an easy scenario, but does not attempt the harder task of taking a scenario, and finding a model which is applicable to it. Its benefits lie in supporting a conceptual understanding of graphs, continuity and principles of calculus, and investigative techniques. Unfortunately, Henn relies too heavily on intuitive appearance of graphs rather than analysis, and the computations are error prone. Henn models the following criterion. If x_1 and x_2 are the incomes of two people and $t(x)$ is the tax paid on income x, then the condition $t\left(\dfrac{x_1+x_2}{2}\right)\le\dfrac{t(x_1)+t(x_2)}{2}$ should hold in a fair tax system. Henn claims that this principle holds. Careful analysis shows that the criterion is violated at discontinuities. Henn also claims that a 1994 change to the tax schedule was "small". It is clear, even from a graph, that the tax paid for incomes less than 15000 DM is markedly different under the two systems. This demonstrates that a scenario can be improperly used.

'Contrived' scenarios

In this section I demonstrate that in an extreme form of scenario modelling, a 'model' is easy to contrive, but it may bear little relation to reality. This weakens the power of scenario modelling as a methodology. Contriving a model directly opposes the view in (Keitel 93), that concepts should be extracted from a real situation.

Problems that I set between 1989 and 1996 as Chief Examiner for the *Applicable Mathematics* AS-level paper of the University of Cambridge Local Examinations Syndicate (UCLES) illustrate how to construct models around contexts. This is more realistic in that the contexts precede the models, but they can be artificial. An example is (UCLES 95), where I produced a model of rectilinear motion with constant acceleration, supported by concocted data. This question is not unique in containing the phrase "... may be modelled by ...". This phrase was a response to syllabus directives (SCAA 93 and restated in UCLES 94), and was merely a means of creating a modelling problem out of an otherwise standard mechanics problem. It was not hard to provide a context for nearly every question on the paper in this way. A more interesting, and real, example did not make it into print. *The Guardian* published statistics (Guardian 94) on honours received by Chief Executives of QUANGOs who contributed to Conservative Party funds, suggesting that they were more likely to be rewarded than those who had made no contribution. Statistical tests (Appendix 1G) confirmed this hypothesis: they were highly significant, but revealed too much embarrassing information...

1.2.4 Evaluation of scenario modelling

The discussions in sections 1.2.1 to 1.2.3 illustrate the following advantages.

- Support of a rigid curriculum. It is easy to illustrate simple concepts and techniques with applications (real or contrived), and such illustrations do not affect the overall thrust of the curriculum significantly.
- Repeated practice of a technique can lead to relative fluency in that technique. The context adds interest. Boekaerts (Boekaerts 95) provides limited evidence that boys are more highly motivated if they can see the relevance of what they are doing, whereas girls are more reliable in applying rules.[2]
- The problems considered tend to be simple to formulate and are not wide-ranging. This aids accessibility by directing effort to the problem under consideration with as few distracting issues as possible.

The following are disadvantages of scenario modelling.

- The modeller can be constrained into associating a particular technique with particular scenarios. This makes it hard to develop models within other contexts.
- It can be hard to develop and refine a model.
- There is a danger of over-fitting by picking a scenario which is a good illustration of a given technique.
- Little account is taken of the thinking behind the model (assumptions, features etc.).
- It does not address the issue of how to formulate relations between features.

[2] But see comments in (Tall 93 - Chapter 3), which contradicts this finding.

1.3 Generic Modelling

I refer to the second of the three principal modelling methodologies as *Generic Modelling*. *Generic Modelling* is an attempt to make a methodology widely applicable across different contexts, retain flexibility, and define how the model is to be produced. The processes involved in analysing a real situation and modelling it constitute a *Generic Modelling Cycle*. The following sections discuss variations on this cycle, and point out advantages and shortcomings of generic modelling.

1.3.1 Generic Modelling Cycles.

Hodgson and Amend (Hodgson 95) point out that views of a generic modelling cycle vary between authors and applications, but have a common basis in the Hirstein Cubic modelling diagram (Hirstein 91, Figure 1.3). Its basis is scenario modelling, but it is a useful preliminary to the generic modelling cycle.

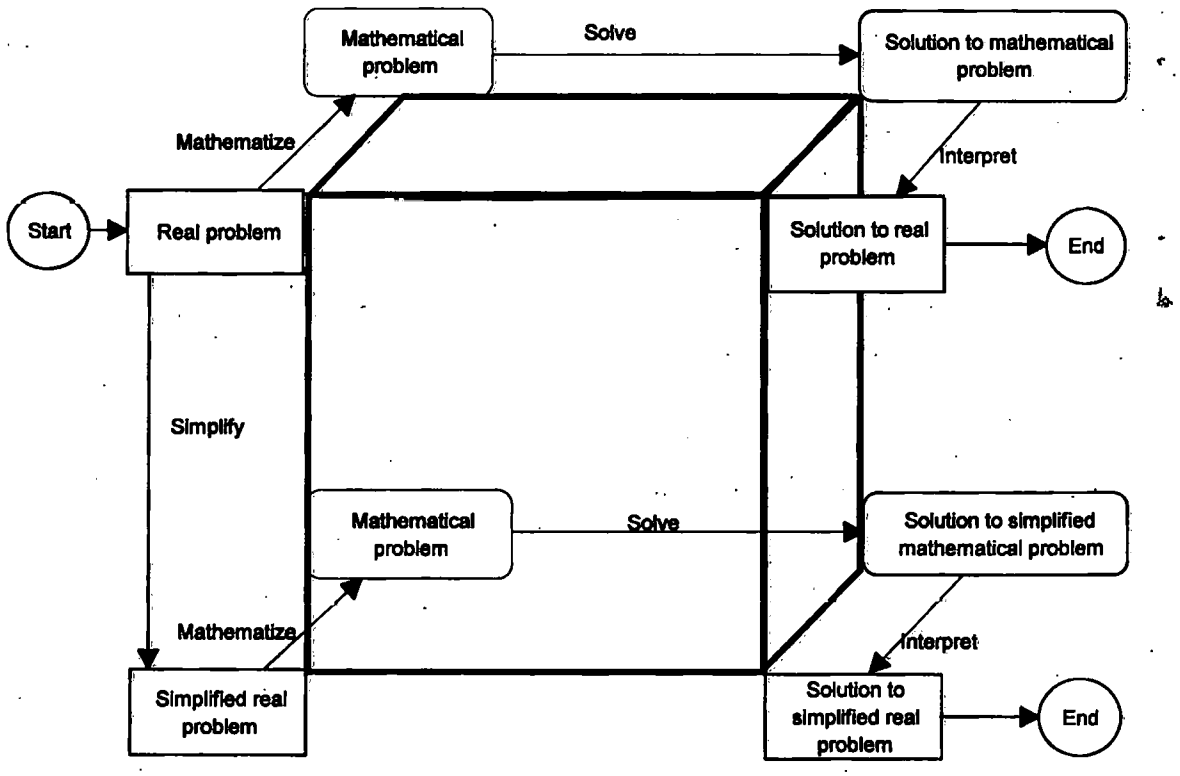


Figure 1.3

The diagram is a cuboid in which the near face represents the "real world", and the far face represents the mathematical world. As reprinted in (Hodgson 95), all of the edges

of the cube are joined by non-directional flows, implying that all paths around the edges are possible. This cannot make sense in all cases. For example, using the path *Simplified Real Problem → Solution to Simplified Real Problem* obviates the need for a mathematical model. In my version of this diagram I have only included paths which are meaningful. I have further enhanced the diagram such that transitions in the real world are represented by solid lines, transitions from the real world to the mathematical world and vice versa are represented by dotted lines, and transitions within the mathematical world are represented by dashed lines. Hirstein reprints his own diagram in (Hirstein 95), but actual flows are still not clear. His 1995 analysis only separates activities in the real world from those in the mathematical world, and shows a vague relationship between them. Nothing is said about how the model may be revised, validated or derived.

Two standard texts on modelling outline a short generic modelling cycle. Edwards and Hamson (Edwards 89) highlight the steps:

- identify the real problem;
- formulate a mathematical model;
- solve;
- interpret the solution;
- present the results.

The formulation stage comprises: draw diagrams where appropriate, collect data, list relevant factors, assign symbols to variables, state assumptions, and formulate relations and equations. Three problems arise from this. First, there is no formal connection between variables and features. Second, the sub-divisions of the *formulate* stage are not formalised in a procedural way. The third concerns the phrase used to qualify the *formulate relations* step: "using your mathematical skills e.g. proportionality, linear and non-linear relations, empirical relations, input-output principle." These rely heavily on experience and do not teach skills. The evidence of Chapter 9 shows that this treatment is insufficient to develop the key modelling skills of relationship formation. Giordano, Weir and Fox (Giordano 97) add a verification stage to this modelling cycle. They include steps such as: identify the problem, make assumptions, ask "does the model make sense?", ask "does it address the problem?", test and maintain the model. These combine to produce the modelling cycle in Figure 1.4, which emphasises validation and iterative steps at the expense of the formulation stage. The concept of "sensitivity" of a

model (the dependence of an output vector on an input vector) is mentioned, but they do not explain why this is important.

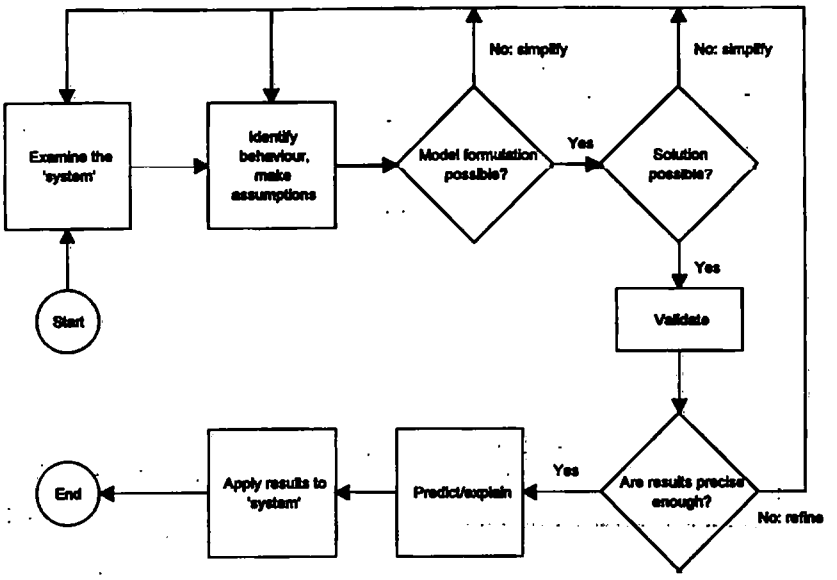


Figure 1.4

The 7-box modelling cycle of Penrose (Penrose 78) demonstrates the principal components of generic modelling, but provides no detail. It forms a basis for many others. The 7 stages are the same as the Open University's model (MST 204 89 - Figure 1.5), which has more detail on each stage. Penrose elaborates little on the stages in this process, and what there is occurs in the form of two example modelling cycles. One useful pointer for the "Set up Model" stage is the Input/Output principle, although this is not applicable to all models.

The 'Produce Model' stage in Penrose's scheme is refined in the Open University's scheme (MST 204 89 - Figure 1.5). It contains essential flow and component details, and is relatively detailed. The subdivisions in each major stage are intended to flow from top to bottom, although there is flexibility in this. The OU does not make these internal flows as transparent: it tends to present them as case studies. The actual guidance for students for the "Formulate Model" stage is not always adequate: it has to be illustrated by specific examples. Chapter 9 contains empirical evidence that students have problems with this stage. Generalisation from specific examples is expected, and this is not always easy for inexperienced modellers.

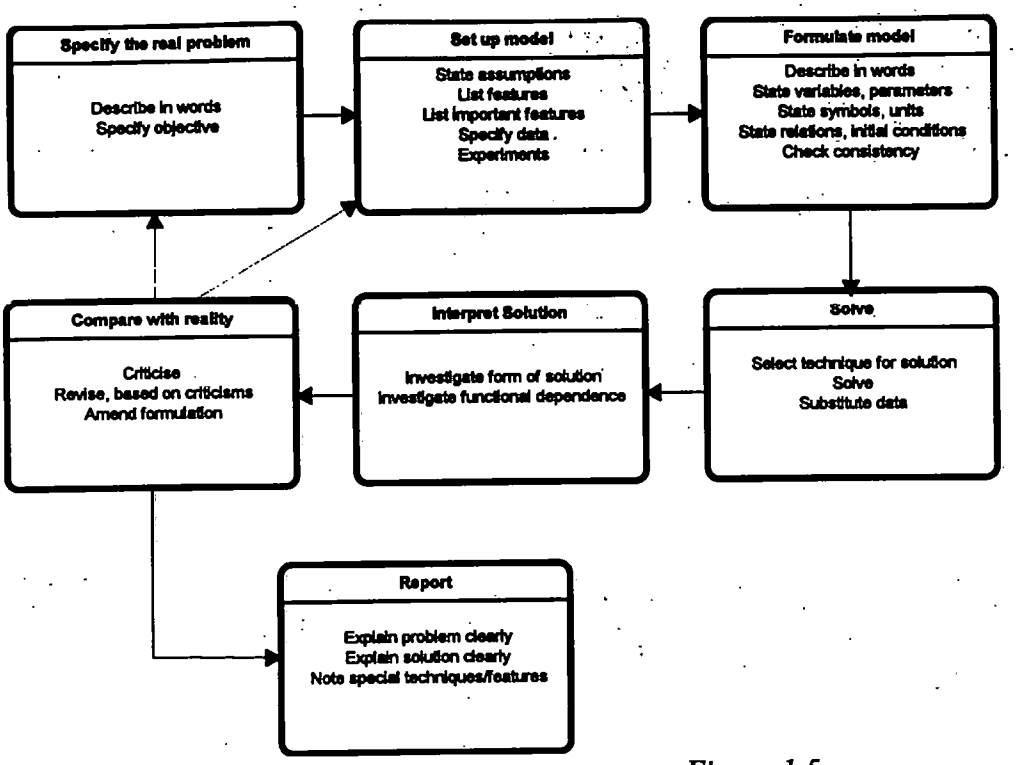


Figure 1.5

The successor to MST 204, MST 207, (Bromilow 97) makes no advance. It includes some new ideas (computer algebra for isolated computations, and multimedia presentations), but the 7-point modelling cycle remains unchanged. In particular, there is no further discussion of the relation formulation stage: the only guidance is based on a case study which uses the input-output principle.

1.3.2 Processes within Generic Modelling Cycles.

Ikeda (Ikeda 97) uses a Hirstein model to qualify the processes underlying variable and relationship generation. He identified six factors:

1 Mathematical knowledge and skill;

2 Knowledge about the real world;

3 Interest in solving the problem;

4 Knowledge about mathematical modelling;

5 Mathematical thinking that will promote the modelling process;

- 5.1 Are there any vague conditions which can be clarified?
- 5.2 To what extent does a variable effect the real solution, if at all?
- 5.3 Can the problem be solved easily?

6 Metacognitive skills about the mathematical modelling activity.

These are too general to be useful. For example, *mathematical knowledge and skills* is an ill-defined phrase, and it is not always possible to tell whether or not a problem can be solved easily before it has been solved. Sub-dividing the fifth component is more practical. In particular, Ikeda says nothing about how each stage in the modelling cycle is to be done. He sheds no light on the variable generation stage: 5.2 is used to "find out several variables that influence...", without saying how this can be done. Similarly the stage "finding relationships between variables" is left to the user, as is "reject marginal features". It would appear that the outcome of 5.2 or 5.3 cannot be determined until the modelling process has concluded.

Orman (Orman 95) illustrates experimental and prediction aspects of the modelling cycle. He uses several iterative stages, the novel one being a PREDICTION→EXPERIMENTS→REAL_WORLD loop, in which experimentation with validation determine applicable physical laws. The cycle emphasises knowledge of the problem domain by explicitly considering symbols, laws and parameters. Lack of such knowledge, or a lack of fluency in manipulating it mathematically, can be a significant barrier. Chapter 9 of this thesis provides empirical evidence for this. Each stage in this cycle must be regarded as a summary of several sub-stages. Without this assumption the cycle is too general to be workable. Experiments are a significant part of Ormans' cycle, but over-emphasis on experimental mathematics should not detract from a focus on mathematical skills. This approach would be defended on the grounds that many students have a weak mathematical background, and that compensation must be made for this.

Herring (Herring 97) considers some specific examples on traffic flow and movement. Generalising, it is possible to qualify the process of producing assumptions and generating variables. He concentrates on objects (a key point of this thesis): the car and the geometry of a junction. When he lists assumptions he is in fact listing features and associating assumptions with them. Herring is right in explicitly linking a symbol to each feature, explaining what the symbol means and stating units if relevant. In Herring's first example the objective function (i.e. the output of the modelling process) is only made explicit after the variables have been defined and some relations between them have been provided. It would have been better to clarify this in advance, as in (MST 204 89).

Hodgson (Hodgson 97) recognises that it is not always easy to equip students with the skills needed to do mathematical modelling. He considers that learning this process in the classroom sometimes fails because of over-packaging of specific examples. If a particular technique is connected to a particular scenario, that connection tends to remain fixed, even though it might not be wholly appropriate. Hodgson attempts to solve this problem by using real problems which are generated by the students themselves. He implies that motivation is a significant barrier to learning mathematical modelling, and that the solution is to tackle problems which interest them. These can be ill-defined and insufficiently taxing. Curve fitting forms a significant part of his modelling process (as it does in Orman 95). Simple data modelling it is a much easier process than formulating an algebraic model. Hodgson does not address the basic problem of how variables are generated and how they are related.

1.3.3 Evidence for problems in Generating Features and Relations

This section contains comments on a limited number of sources that indicate that a generic modelling cycle is not always adequate as a learning tool. Earlier research (mainly from Treilibs) provides some indication of the significance of generating features, variables and relations, and hints at problems in these processes. Potari's later study provides direct evidence that students cannot always generate features, variables and relations. I also discuss some indirect evidence. This section relates mainly, but not exclusively, to generic modelling,.

Early studies of Formulation Processes in Modelling

The research of Treilibs (Treilibs 80 and Treilibs 79) is useful in isolating detailed processes in mathematical modelling. He found a significant (at the 5% level) correlation between mathematical and modelling ability, and suggested that good modellers were creative, intuitive, had insight, and did not worry about detail or making mistakes. These conclusions seem too general to be useful, especially as Treilibs' reports some inherent bias: students' attitude to the problems affected their success at solving them, and students from a particular selective school produced particularly good results. Treilibs administered standard tests designed to assess selection and generation of variables and relations. There was a particular problem with the *select variables* test,

which was found to be ambiguous in practice, and the *select relations* test appears to test selection of a model and not relations. Precise answers were supplied for marking purposes, even though it is clear that these answers are not unique. Hence, this experiment was limited to an investigation of pre-defined factors, for only two problems. There is no evidence that the results are more generally applicable. Trielibs' conclusion, that conventional mathematics teaching develops application skills but not construction skills, is therefore limited. His flow diagram for formulation processes in modelling contains many imprecise phrases (such as "generate ideas on the empirical situation", "identify mathematical variables"), and appears to add variables as the main means of improving a model, without discussing the consequences of removing variables.

Shortly afterwards Salzano (Salzano 83) investigated teaching methods based on Treilibs' methodology. Her result (poor mathematical ability implies poor modelling ability) is severely biased by the low ability and lack of motivation of the students concerned. She does include the interesting idea of an "open (-connection) diagram", which consists of a features list linked in a (potentially) fully connected node-arc diagram. This identifies links between objects in the problem domain, which is an important idea in this thesis.

Direct evidence of problems in Feature and Relation Generation

Few direct evaluations of problems in feature and relation generation exist, and the principal source of explicit evidence is (Potari 93). This is discussed in detail in Chapter 9. Potari reported that students managed to explore relationships but could not formulate them beyond intuitive arguments. His sample was biased towards limited ability modellers. Earlier studies suffered from the same type of bias problem: (Salzano 83) from weak students, and (Treilibs 79, Treilibs 80) from a subset of strong students. Treilibs found that even strong students who managed to generate relationships tended to produce numeric rather than algebraic models, using relatively unsophisticated mathematical constructs. This phenomenon serves to expose problems with relation formulation.

The purpose of Hamson's *Interactive Simulation* software and *Systems Dynamics* modelling tool (Hamson 97) was to generate features and relations ("levels") in a model. He noted "Students did not always relate all their 'levels' appropriately...". This problem was ameliorated in Hamson's study by supervisor checks on model formulation and by explorations involving changing the numerical values of parameters. This diminishes any underlying algebraic model, which is limited to input-output scenarios involving ODEs. Beare (Beare 96) indicated limited success in generating relations: his students could formulate relations in words, but were unable to translate this into algebraic terms.

Indirect evidence of problems in Feature and Relation Generation

Indirect evidence from other sources identifies problems with model formulation, but it is not always possible to isolate the source of these problems. It is not possible to infer that problems with model formulation are solely due to problems with relation generation, but I suggest this as a possibility. This section contains a summary of instances in this category.

Several studies reviewed in this chapter state that student interaction is necessary to generate features in models (Ikeda 97 and Hodgson 97). This indicates that feature generation is a problem. Ikeda reports additional problems with non-algebraic approaches to finding properties of features (and hence in relation generation). Hodgson supports this finding by encouraging relation generation by practical data fitting. This could imply that an algebraic approach became intractable. A minor quantitative result comes from the *Kassel Project* (Kaiser 95). Only 20% of German students and 5% of English students tackled a 'mathematisation' problem in a comparative test. These percentages were much lower than the 'take-up' rates of other problems, despite an emphasis in England on 'real' problems. Kaiser did not define the difficulty clearly: students were "unable to mathematise". This is, however, a pointer to difficulties in defining rules and abstracting mathematical concepts. These inhibit development of actions and relations in a model.

1.3.4 Generic Modelling: Summary

The advantages of generic modelling are:

- It is designed to be applicable to any context.
- Using a large feature list gives the capability of generating diverse and varied models.
- Validation and refinement provide feedback on the effectiveness of the model.
- Starting points are provided which are accessible to most modellers, even if further stages in the modelling cycle are less accessible.

The first problem in the following list is the most important, and is tackled in this thesis.

- The stage that deals with construction of equations is particularly difficult to implement, and the methodology does not provide sufficient guidance. This applies even when constructing models based on an axiomatic system (e.g. Newtonian Mechanics), where the equation-generation stage is relatively well-defined and there are many prior examples.
- The method is hard to implement if the problem domain is very specific.
- The link between the features list and variables is not necessarily controlled. The result is that features and variables can be "unused" in the model or that features which were not in the features list appear in the model.
- There is no control over the use of assumptions in the model itself. Hence, assumptions can be listed and either not used or contradicted. Conversely, unstated assumptions can be used implicitly.
- Steps in the method are not quantified. Thus, there is no way of assessing completeness of a features list, or the relative importance of features.

1.4 Modelling through Case Studies

Scenario modelling applies contexts to techniques, whereas *Case Study* modelling associates techniques with contexts. Case Study Modelling stands apart from mainstream modelling methodologies, but is still widely practiced. An analysis of common case studies reveals much commonality in approach. Taylor (Taylor 86) makes the unfortunate comment that there is "... no obvious way to teach mathematical modelling...", and then attempts to do so by presenting differential equation case studies. The comment is significant in that Case Study Modelling provides a way forward in the absence of other methods.

1.4.1 The scope of Case Study Modelling

Examples of specific case studies where a mathematical model has been developed for a real scenario are typified by, for example, Huntley and James (Huntley 90). They present short and extended case studies, following a brief discussion on formulation of mathematical models. They do not address any general methodology. Hence their models are, in principle, no different from dedicated mathematical models which have been developed to solve "real" problems. There are many examples, either in dedicated modelling journals (for example Rodin 89, Alabi 89 and Greenhalgh 90), or in journals dedicated to a given technique (for example Haie 93) or in journals dedicated to a given tool (for example Mitic 94 and Loe 85). These accounts contain little or no information about how the model originated. The principal aim is to solve the problem, with little regard to the formulation process.

There are problems with a case study approach if the primary aim is to teach the process of modelling.

- A given model may be difficult to apply to other contexts: it may be too specific to be useful.
- Case studies are not always conducive for abstraction of ideas and techniques.
- It encourages the modeller to choose one technique from a limited library of techniques and to associate a particular technique with a particular scenario. None of these may be optimal for the particular problem under consideration.

1.4.2 Classification of modelling case studies

Giordano, Weir and Fox (Giordano 97) outline typical modelling contexts, which also appear elsewhere (e.g. Andrews 76, Berry 95, Huntley 90, Edwards 89 and MST204 89).

- Models based on an axiomatic system - (e.g. Newtonian Mechanics, heat transfer). This includes systems for which the axioms are not always explicit (e.g. queuing processes, where behaviour may be implied).
- Simulations for models which will not admit an analytic formulation or solution.
- Discrete dynamical systems (principally difference equations - discussed comprehensively in Sandefur 90), and stochastic processes.
- Continuous dynamical systems (principally differential equations).
- Optimisations: continuous and discrete, constrained and unconstrained.
- Empirical: curve fitting and dimensional analysis.
- Models based on an input-output (conservation) principle. Aris (Aris 78) cites this principle as *the* method for formulating relations. This is incorrect as it is not applicable in all cases.

The uniformity of modelling case studies is striking. Case study modelling associates a mathematical technique to a particular scenario, and it is difficult to break away from the association once it has been seen. There is also a marked similarity of treatments between texts for similar topics. Appendix 1GWF lists the topics covered in (Giordano 97), and also lists similar treatments in other texts (the topics and emphasis sometimes differ slightly). From this analysis, only a limited number of deviations from "standard" models can be isolated, and these are discussed below.

1.4.2 Deviations from 'Standard' models

Since source material for Case Study Modelling is static, the Case Study methodology lacks a formal way of producing a different or more advanced model. In this section I discuss some ways in which this has been done, although these techniques only occur as further case studies.

More advanced technique in a standard model

Giordano, Weir and Fox combine a standard continuous population model with population functions expressed as cubic splines. This complicates the solution process, which has to be done piecewise, using appropriate boundary conditions at interval ends. Huntley and James include a route planning model in which a oblique passage across a square is modelled by constructing an integral, representing an average distance travelled. This is more advanced treatment than most others in this section, and introduces different concepts into a sub-model. Both of these cases introduce new (and more advanced) mathematical ideas and techniques, which is interesting, but there are no fundamentally new concepts.

Standard model with a non-standard component

Edwards and Hamson consider a model which involves resisted motion with an unusual resistance function. The result is a differential equation which is either difficult or impossible to solve analytically, depending on the exact form of the resistance term. There are no fundamentally new modelling concepts: just difficulties in solving the resulting equations. Current research in Boundary Element (BE) Analysis represents an extensive example of how standard techniques may be amended to solve different classes of problem. Brebbia made the significant BE development steps in the late 1970s (Brebbia 78 and Brebbia 84), including the fundamental Fortran code (discussed in more detail in Brebbia 89). Subsequent models involve varying this basic Fortran code.

Model decomposition

Andrews and McLone (Andrews 76) present a number of advanced case studies, mostly involving standard techniques in optimisation, differential equations, applications of the input/output principle, and geometry. One model is particularly noteworthy. This is a business planning model for telecommunications, in which distinct sub-models representing income, current expenditure, manpower, capital expenditure, depreciation and finance are combined. It is unusual to see a model which can be decomposed in this way, and this is a more significant deviation from a standard methodology. Each sub-model is capable of separate analysis. Each case study in this text contains a set of skills required to formulate and solve the problem. It is significant that none are listed in the business planning model.

Case study for a 'real' model

Thomlinson and Norcliff (Thomlinson 89) describe a taught, but real modelling situation (in oil extraction) that contains no elements of artificiality. The model contains new techniques and ideas (for the students), and is totally realistic. It is therefore an excellent idea in principle. However, so much material must be supplied that many aspects of a generic modelling cycle are removed, and the modelling process reduces to a case study. The model itself, including variables, symbols and relations, and a methodology and techniques for obtaining a solution are supplied. Modelling is therefore tied to a particular route, from which it is very difficult to deviate. Whether or not any skills learned on this type of project are transferable to other scenarios is an open question.

1.4.3 Evaluation of Case Study Modelling

Case Study modelling has the advantage that it is capable of providing a reasonable template for modelling in a given context. The analysis in Appendix 1GWF shows that the number of modelling contexts is small, so providing a template for each is feasible. This 'advantage' is not necessarily beneficial: once a model becomes associated with a given context, it can be difficult to construct a completely different model for the same context.

The most significant problem with *Case Study* modelling is that it is static and contains no processes for constructing a model. Whether or not generalities can be inferred from specific models is questionable. Empirical evidence in Chapter 9 shows that even with the relatively well-structured *Generic* methodology, significant problems can occur when attempting to generate relations. *Case Study* modelling does not even consider relation formulation. There is also no guarantee that techniques can be extended to produce more complex models or models in other contexts.

1.5 Other methodologies

In this section I consider some further methodologies which have different elements to the ones considered so far, and also seek useful techniques which could supplement existing methodologies.

1.5.1 A link between cognitive processes and model formulation

The first alternative methodology is due to Beare (Beare 96). He describes the *Warwick Spreadsheet System* (WSS), which is a dedicated spreadsheet for mathematical modelling. It is embedded in a methodology which differs from other considered so far in the following respects:

1. It concentrates on cognitive aspects as a major component of the modelling cycle.
2. The modelling process is geared to producing a model on the spreadsheet.

The cognitive aspects of the methodology behind the WSS make it fundamentally different to the three mainstream methodologies considered so far.

Its basis is the *Warwick Spreadsheet* (WS), which is an Excel derivative, with dedicated menus and a means of avoiding explicit cell references by using templates. The WSS does not address the problem of how algebraic relations in the model are to be derived, and this has to be done 'by hand'. Although removing Excel syntax appears to be an advantage, being able to use a spreadsheet is a valuable skill in its own right, and using cell references provides valuable insight into algebraic processes.

The WSS gives rise to a 4-domain (real world - cognitive domain - mathematical model - computer model) modelling cycle. This acknowledges that modelling depends on techniques, facts, laws and equipment, and separates cognitive structures from implementation issues. It emphasises the roles of subject and domain knowledge, but leaves the processes associated with them unclear. This illustrates a fundamental problem with this cycle: it says 'what' but not 'how'. Processes such as 'Formulate mathematical model', 'Modify mathematical model' and 'Understand the situation' are harder to achieve than to specify. Beare gives limited guidance on many of these steps. One positive example is that writing down a relationship between variables in words helps in formulating an algebraic formula. There are indications that some problems occurred in formulating models which involve a non-axiomatic basis, which is evidence

that the WSS only provides a way of managing a model rather than formulating it. There is also no formal discussion of features, assumptions and relations. Another constraint on the modelling process is that a model has to be expressible in terms of spreadsheet constructs. This means that continuous processes (e.g. differential equations) must be discretised, which may not be a fundamental part of the modelling process. Beare does not indicate any advantages of the WSS over other software or methodologies, and its use has not been tested rigourously.

Warzel proposes a more wide-ranging approach (Warzel 89). He takes, as a starting point, Stachoviak's explanation of a model as a mapping (with inverse) from one object to another. This mapping is supported by an elaborate notation and terminology, but it appears to reduce to a diagrammatic representation of modelling which involves "the real world" and "the mathematical world" and a mapping between them. Warzel's approach is therefore closest to a generic modelling methodology, but it concentrates on theoretical aspects of the "real/mathematical world" division. As it stands it is not useful until relationships are explicit and determinable. Warzel goes on to discuss a theory of actions arising from this meta-model. This is also divorced from reality, but implies an investigation of the interactions of objects within the system. Warzel's analysis is similar to that in (Dreger 89): interactions and responsibilities of objects in the system are approached from the point of view of considering "function points", which are end-user business functions.

1.5.2 An approach based on linking elements in the problem domain

Hamson and Lynch (Hamson 97) describe a method of solving systems of non-linear ordinary differential equations numerically on the computer. The novel aspect of this work is to clarify functional dependency using the aid of "influence diagrams", which are drawn as directed graphs. The derivatives of state variables X, Y, ... are associated with functions F, G, ... of X, Y,... on the diagram, so that the model can be built from the diagram. Hamson and Lynch are established and experienced teachers of mathematical modelling, and their approach is extended considerably in this thesis. However, this process cannot determine functional forms for the relations between the state variables. Hamson and Lynch say that "modelling skill" is needed to be able to find them, without suggesting how this should be acquired. The methodology is also limited to systems of

differential equation, although it seems possible to extend it to other contexts. They report that the students found the projects difficult and that much discussion and tutor input was necessary in order to produce a reasonable influence diagram without serious errors or misconceptions. It is therefore hard to assess its effectiveness.

Hamson and Lynch's approach contains elements of scenario and case study modelling. It is based on specific mathematical entities (non-linear ODEs), within given contexts. It does attempt to formulate relations as part of a formal methodology, and is therefore more sophisticated than scenario modelling.

1.5.3 Problem-solving Heuristics

Polya's text (Polya 45) on mathematical problem-solving is significant in that it was an early text on general problem-solving issues. Its approach is to present a sequence of heuristic strategies, which can be used as a reference. Useful examples for modelling include "Could you restate the problem?", "Do you know a related problem?" and "What is the Unknown?". Each heuristic contains case study examples of how it may be useful, and Polya's approach is therefore similar to *Case Study* Modelling. Although such a reference may give some guidance in particular circumstances, it provides no detailed strategy for problem-solving. The only suggestion is an overall 4-stage process: *understand the problem*, *plan*, *carry out the plan* and *look back*. These are insufficiently specific to be useful for modelling. Even an extensive catalogue of heuristics which are directly relevant to modelling would suffer from the problems highlighted in Section 1.4.3: inflexibility and a static nature.

1.6 The use of Computers in Modelling

Having discussed the principal mathematical modelling methodologies and some minor ones, I now consider how using computers have supplemented them. I show, as a preparation for Chapter 7, that many of the ideas for using computers in modelling were not fundamental to the model formulation process. They are useful, and in some cases essential, but are not integrated into the model formulation process. These comments must be seen in the context of the hardware and software which was generally available at the time: a DOS environment in the 1980s, Windowed environments in the 1990s, multi-media from 1995 onwards, and computer algebra packages starting in the late 1980s. [3]

1.6.1 Computers in modelling: 1989-1997

Four papers from the late Eighties and early Nineties summarise thinking about the use of computers in modelling at that time.

Bowtell (Bowtell 89) advocates the use of computers to implement simulations which consist of animations based on the analytical or numerical solutions of differential equations. These have since become widespread but are limited because they are specific to the differential equation concerned.

Blum discusses how to use computers in mathematical modelling briefly in (Blum 91A). He includes Logo, statistical packages and simulation software, but it is not clear how he thinks these should be used in modelling. In principle, Blum considers that software is best used for illustration and numerical computation. He talks about using software for "acquisition of mathematical concepts" but does not say how. Blum mentions the disadvantage that devaluation of routine computational skills can result from using computers. He does not back this claim by empirical evidence, but there is limited evidence from one other source. Mayes (Mayes 95) reports devaluation of routine computational skills in using computer algebra laboratories in the United States.

Huntley (Huntley 91) discusses the types of programming languages which he thinks might be useful in modelling and applied mathematics. These are dedicated modelling

[3] I claim to have imported the first copy of DERIVE 2.0 into the UK from the United States in 1988

languages, Prolog, and specification languages such as Z. It is not clear from the examples given exactly how these would be used in modelling, or how to overcome the problem of learning these languages. He criticises the computer algebra systems which were available at the time for not being user-friendly and requiring the learning of a new language. It is odd that he does not make these criticisms of the three categories to which he devotes the most detail in his paper. There is also no discussion at all about procedural languages (particularly Basic) which were in common use at the time.

The software that Clements (Clements 91) thought would be useful in the mid- and late-1990s is now in widespread use. He advocated using a tool for numerical computation, a graphing tool, a differential equation solver and a tool for doing algebraic manipulation. All of these are performed by the principal computer algebra systems available today, and the first two are also fulfilled by spreadsheets.

The way in which the computer has actually been used in mathematical modelling is typified by Lawson and Tabor (Lawson 97), who describe computer based experiments in mechanics. This software now forms a mechanics component of MathWise. They do not discuss the type of application used to produce the software. The most common programming tools are languages such as Visual Basic, C++, Toolbook etc., which perform purely numeric computation. They are therefore not as versatile as symbolic computation packages. This type of simulation can reinforce theory, perform "impossible" experiments (such an motion in a vacuum or without gravity), and can slow down or speed up simulations. The main disadvantage of this approach is that the experiments are not real and tend to produce much more precise results than would be possible in a physical experiment. The range of experiments provided is also limited by the software. However, they are much more wide-ranging than could possibly be provided by a traditional text or actual experiment.

Overall, the software to date has provided a useful tool for visualisation and simulation, but has not had a major impact on modelling. Evidence for this comes from ICTMA-8.

1.6.2 Spreadsheet Modelling

The spreadsheet has gained prominence as a modelling tool in the late 1990s.[4] This late introduction into curricula indicates further reluctance to use software in mathematics and modelling. Clements (Clements 95A, Clements 95B, and Clements 95C) provides examples of spreadsheet analyses of differential equations. He emphasises the advantages of this technique in all three articles: "the spreadsheet can be used to illustrate, in an accessible and graphic way, some of the concepts which a student of mathematics must grasp when studying numerical analysis" (Clements 95B). He does not make the point that cell references in a spreadsheet act in the same way as algebraic symbols, thereby reinforcing algebraic concepts (see the comments on Dettori 95, in this section). Neither does he make the point that there is often a useful relationship between a spreadsheet layout and the model's geometry and that there is a disadvantage in having to program some methods (particularly in calculus) explicitly. Davies (Davies 97) illustrates the complexity of spreadsheet analyses. He describes how to a implement a FEM scheme for solving a PDE. The complexity of the programming adds significantly to the problem-solving task.

Rodgers (Rodgers 93) describes how fundamental modelling is done "on paper", not on the computer. He uses spreadsheets to compute finite difference approximations to partial differential equations, using models built on the established theory of eigenfunctions. The results that can be achieved are impressive. The spreadsheet formulae reinforce the finite difference schemes such that their precise role is clear, and it is possible to see how assumptions about step sizes affect the approximation. However, the spreadsheet is not fundamental to the formulation stage of the modelling process. In most cases, relevant equations are merely stated and the discretised form of them is programmed.

Shutler and Springham (Shutler 97) describe an apparently different problem. They model the shifting umbrella population of Singapore by constructing a Markov chain, the formulation of which requires a good understanding the problem, and a significant time to formulate it. An interesting consequence of using a spreadsheet in this context arises from an optimisation by repeated iteration. This optimisation uses search techniques instead of calculus, which would mean differentiating with respect to a

[4] but had been used in industry to find approximate solutions to PDEs since the early 1980s.

discrete variable: not so respectable! Shutler and Springham's application indicates ties to 'traditional' methods and curricula, and also some reluctance to replace traditional practice. More research is required to assess the effects of wholesale integration of computer methods into the curriculum.

Few authors link algebraic concepts with spreadsheet formulae. This link applies fundamentally to mathematics, and indirectly to modelling. Dettori (Dettori 95) attempts to do this, but fails to state the fundamental relationship between algebraic symbols and cell references. Indeed, he implies that this relationship is absent and that spreadsheet formulae are functions, not relations). He asserts that, as a result, the spreadsheet cannot be used to represent algebraic models, and a functional variable "X" cannot be isolated. Both assertions are wrong. The obvious isomorphism between terms in a formula such as $y=3x$ and the spreadsheet formula B1=3*A1 shows the relevant functional form. The variable "X" is also clear. The fact that the spreadsheet formula has multiple instances (B2=3*A2, B3= 3*A3 etc.) is merely a distractor.

1.6.3 Developments at ICTMA-8

ICTMA-8, in 1997, was a turning point for computing in modelling. There was a marked difference between this and previous ICTMA conferences, with a significant discussion of the use of computers and software, and computer algebra software in particular. Reports ranged from using Graphical calculators (mainly the TI92 - Brown 97), to specific modelling applications (Engebretson 97, Geiger 97), and general applications in mathematics (Jungerson 97, Pemberton 97). A range of software was covered. Virdefors (Virdefors 97) used Excel to model planetary motion and data models for solar eclipse prediction. McRae (McRae 97) used Cabri for geometric modelling, but argues that spreadsheet modelling is 'sterile'. Whether or not he has investigated possibilities is unclear. The paper by Goos (Goos 97) is contradictory: he prefers visualisation tools and graphical calculators to spreadsheets, but does not explore spreadsheet graphics. Huetinck (Huetinck 97) uses a variety of packages (Mathtype, DERIVE, Cabri, spreadsheets, multimedia) in an integrated programme of mathematics education that looks similar to the MathWise approach. Similarly, Bromilow (Bromilow 97) describes the OU MST 207 course, which uses multimedia and MathCad

(the latter only for 'calculator' type computation). Henn (Henn 97A) stresses the use of computer algebra packages such as Maple.[5]

This work is praiseworthy, but is not new. Neither are the arguments for and against its use. Authors who had said little about computation in modelling appeared to accept that it can be an invaluable tool at ICTMA-8. Henn describes an experimental use of Maple in *Abitur* work, saying that this experiment is unique in Germany.[6] Neill (Neill 97) relegates possible advantages of using Maple for mathematics learning research at the University of Ulster to a "postgraduate investigation". Hamson, who has done pioneering work in modelling at Glasgow University without advocating the use of the computer, belatedly describes a dedicated modelling package, *SB ModelMaker*, for modelling dynamical systems (Hamson 97A), and how he has used Excel to do simulations (Hamson 97B). The sudden injection of so many papers on computers in modelling and mathematics may indicate that the modelling community has been, in general, unaware of computer tools and experiments in their use elsewhere.

Using computers and computer algebra software in mathematics is not new, particularly in Austria. There were experiments in Austrian High Schools dating from the mid 1980s (Aspetsberger 84 and Aspetsberger 89, both using MuMath). Similarly Char and Calmet have commented on experiments in Canada and France respectively (Char 84 and Calmet 84). Arguments in favour of using a CAS are now 14 years old (Stoutemyer 84). *MathWise* courses were also established in the UK by 1993 (Bielby 93).

1.6.4 Computer Algebra applications in Modelling

Townend and Pountney (Townend 95) present an extensive source of examples of how DERIVE may be used in modelling. Their stated methodology is *generic* modelling, but this methodology is not apparent in many examples. Most are presented as case studies. Townend and Pountney's text concentrates on using computer algebra to do routine manipulations in algebra and calculus. Computer algebra is therefore not an integral component of the modelling. The authors acknowledge that the technicalities of using

[5] Proceedings for ICTMA8 were published after this thesis was submitted (ICTMA8 98). The papers by Geiger, Huetinck, Bromilow, Engebretson, Jungerson and Virdefors did not appear.

[6] but see comments on introduction of IT in German Schools in 1984 in Chapter 3: the Bund-Länder-Kommission für Bildungsplänung und Forschungfölderung

DERIVE to produce certain results can be difficult. Such additional difficulties can easily detract from the modelling process. Nevertheless, they present some interesting alternatives and variations for some modelling contexts. They formulate the *Firebreak* problem (see Chapter 6, section 8.2) as a linear program, and describe more sophisticated spring-dashpot systems for the *Speed Bump* problem (also Chapter 6, section 8.2). In both cases, they formulate the necessary equations with sufficient explanation of what they are doing, but with little indication of why they do it. Unusual features (e.g. a relationship between the height of a tree and its commercial value) indicate that they have a deeper than normal insight into some problem domains. This shows that domain heuristics play a significant part in model formulation.

Less extensive reports on the practice of using computer algebra in modelling (but more generally in mathematics teaching) come from conferences such as:

- T^3 (Teachers Teaching with Technology - worldwide but based in the US);
- ICTCM (International Conference on Teaching with Calculators in Mathematics - Texas Instruments, US);
- ICTMT (International Conference on Technology in Mathematics Teaching - Europe);
- ACTM (Asian Conference in Technology in Mathematics - Asia/Australasia).

Not all publish proceedings, but certain points are clear from the numerous presentations;

1. most applications deal with mathematical techniques, not modelling;
2. most applications are at a relatively low level (pre-calculus and simple calculus techniques);
3. few discussions use computer algebra directly, although their number is increasing as the TI-92 gains favour;
4. Few presentations do more than describe how to teach given topics with computer algebra.

From the 1998 T^3 and ICTCM conferences, the following are unusual in that they concentrate on modelling. Longhawk (Longhawk 98) models symbolic periodic motion with the TI-92. Lecture synopses indicate that Sheldon and Peterson's presentations [7] will probably be similar but use purely numeric models with graphical analysis. Kruczynski (Kruczynski 98) uses fitted data and piecewise functions to model resisted motion. Kawski (Kawski 98) makes the point that using technology makes a paradigm shift possible: computation allows for new methods and developments. Such comments are rarely explicit in papers from the above conferences. They are possibly implicit in other presentations. The numerous presentations (about 450 at the T^3-1998 and 300 at ICTCM-98) indicate an essentially concept-based and technique-based approach, similar to 'scenario' modelling.

1.6.5 Non-appropriate and hidden use of the computer

This section describes how inappropriate techniques, supported by the computer, may be used to solve modelling problems. The problem is that mere use of a technique without clear advantages (at best without disadvantages) does not justify use of that technique. Wilenski (Wilenski 97) discusses a problem from a textbook on calculus which was intended as a maximisation problem. Wilenski saw it as a problem in discrete probability and "solved" it using a simulation written in StarLogo. He argues that the solution is useful because it makes the problem "amenable to probabalistic methods" and is as general as an analytic solution obtained using calculus.[8] He does not justify his first comment. His second comment appears to depend upon the necessity to make modelling assumptions (no air resistance, fixed force applied to the ball etc.) for the calculus solution. He does not say that equivalent assumptions are necessary for the simulation: random numbers generated must originate from a given distribution. The same problem is apparent of much Logo research: authors do not consider other tools for the same job (Sacristan 97, Wilenski 97A, Edwards 92 and Kieren 92)

---

[7] Sheldon - Mathematical Modelling at the Pre-calculus/College algebra Level; Peterson - Hands-on Modelling of 1- and 2-D Motion with the TI-83, both at the 11th. ICTCM Conference, New Orleans, http://hepg.awl.com/ICTCM/ in November 1998

[8] I solved this problem in 5 minutes using a pen and (literally) the back of an envelope! Wilenski's program requires about 50 lines of code (including necessary comments), a computer and unquantified processing time.

At a more advanced level, using software, and computer algebra software in particular, merits no more than a mention (Tsai 98 and Siew 97), even though their tasks would have been impossible without. To some extent, this characterises modern 'applied mathematics', and indicates a reluctance to accept computer algebra methods.

1.7 Quantitative studies of the modelling process

Very little information is available to compare modelling methodologies. This section discusses research which has been done. Although the studies concerned (Kaiser 95, and Burghes 93) originally set out to compare modelling in Germany and England, where the main methodologies in use were *scenario* and *generic* modelling respectively, they were not intended to compare modelling methodologies. Modelling methodologies were not explicit. Burghes and Kaiser referred to modelling "in Germany" and "in England", so that no general inference can be made about particular methodologies.

1.7.1 Comparisons of modelling in Germany and in England

In 1993 Kaiser and Burghes (Kaiser 95) carried out a comparative study of learning contextual mathematics in England and Germany. This was largely a qualitative study and few data are reported. They also state several inherent biases:

1. different degrees of selection of students in the two countries;
2. early specialisation in England;
3. an inflexible curriculum in Germany.

Kaiser's view of English education is idealistic and outdated: "the English Schools knowledge tradition is described as humanistic, based on the principles of morality (ideal of the Christian gentleman), individualism, and specialism." This may have been true of independent schools in the UK many years ago but is not appropriate today. She points out specific differences between the two countries which introduce bias. She states that in England there is:

- the 'discovery' method rather than didactic teaching;
- more practical work;
- less class discussion and teacher direction;
- less emphasis on precision and correct mathematical speech.

These indicate a more rigorous treatment of mathematics teaching in Germany. The authors present data which indicate, on the basis of percentages of correct answers in a range of algebra, geometry and modelling questions, that German students perform better. Only 20% of German and 5% of English students attempted a mathematisation problem, indicating that neither country is particularly successful in teaching modelling. The data are insufficient to make deductions about the relative merits of mathematical modelling teaching from a statistical point of view, and it is impossible to test the significance of any one factor.

Kaiser's discussion has the same content, and some parts of the text are exactly the same as the discussion in (Burghes 92). It is therefore not original, and Burghes makes some points that Kaiser does not make. Burghes' study aimed to investigate the efficacy of modelling approaches in England and Germany. His attitude to quantitative studies is odd. He states that "quantitative-statistical" methods were not appropriate for the initial phase of the study, yet they do form the substantive part of the second phase of his study. Further sources of bias then emerge. An experimental teaching course was used in England, and this study took place at a time of marked change in attitudes and practice in England and Wales (the introduction of the National Curriculum). Also, the observation that teachers in England found it difficult to structure individualised work programmes casts doubt on the effectiveness of their teaching.

Quantitative results from this comparison are reported in (Burghes 93). The authors have taken care to select samples in Germany and England such that they are as near as possible equivalent in order to reduce bias. They developed tests for "mathematical potential", but there is no evidence that these tests can be used as a measure of acquired mathematical knowledge and skill, or as a test for modelling ability. The results indicate superficially that there is considerable advantage in being German. Of the 27 Chi-Squared tests, 14 were statistically significant: twelve in favour Germany and two in favour of England. However, a test of proportions, which uses the fact the Germans scored better for 18 out of 27 questions is not significant (Appendix 1B). The standard deviation data fail a Variance Ratio (F) test, indicating that the two samples need not originate from the same background population. These results weaken the superficial solidarity of the result in favour of Germany. Burghes explains the results by suggesting that many high-ability pupils in the English sample were creamed off to independent schools. It would be very difficult to quantify this. Secondly, he thinks that the

differences are attributable to different teaching styles and a different emphasis on topics in the two countries. He concludes by proposing future work in which mathematical attainment is to be determined as a function of factors such as schemes of work, class size, teaching style and type of school. The questions posed in (Burghes 92) were not answered in (Burghes 93).

1.7.2 Extensions: mathematics teaching worldwide and cultural influences

The results reported in (Burghes 98 - the *Kassel project,* which is a follow-up to his 1992 and 1993 papers) deviate from assessing how mathematical modelling should be taught. It is an extensive study "to carry out research into teaching and learning of mathematics in different countries, and ultimately to make recommendations about good practice in helping pupils achieve their mathematical potential", and to find "the key factors that give rise to successful progress in mathematics". It covers 19 countries, with the broad conclusion that mathematics teaching in Western Europe is inferior to mathematics teaching in Eastern Europe, which is inferior to mathematics teaching in Singapore. The statistical evidence to back this claim is absent. Factors such as work ethic, time spent on mathematics, and differing emphases on topics and techniques are mentioned as contributory factors. I attempted to confirm Burghes' assertions by requesting comparative data on recent A-level results from Singapore and the UK from UCLES (University of Cambridge Local Examinations Syndicate). UCLES declined to provide the necessary data, and Burghes' assertions remains unsubstantiated.[9] Burghes provides "recipes for success" in mathematics teaching, centred on depth and rigour of teaching, differentiation of work and whole class teaching. These claims need to be tested extensively, and must be placed in the context of national backgrounds. As such, they are untested. Burghes indicates that he will make "a series of universal recommendations for effective maths teaching - independent of culture and society". There are no indications from Burghes' previous papers to show how he intends to do this, and I await this result with much scepticism.

[9] The Syndicate probably thinks that A-level results in the UK and Singapore are not comparable because of differences in time spent on mathematics (including extra coaching), better examination preparation through O- and AO-level, and a wish for a high profile. Unofficially, it would be embarrassing for UCLES if it were shown that standards are higher for the Singapore examination season (Winter) than for the UK season (Summer), yet Singapore results are still much better.

The idea that cultural factors are important is reinforced by the study of Chandler (Chandler 97), who compared expectations of mathematical achievement in England and Wales and Bavaria. She considers that "...we demand of our school leavers considerably less ability to reason through a problem than the equivalent in Bavaria. Candidates here are given problems that are broken down into steps needed for the solution. In Bavaria, the questions are set so that candidates have to reason this out for themselves." Chandler backs her claim by quoting example questions from the relevant examinations, although it is not possible to tell whether or not these examples are typical.[10]

1.8 Concluding remarks

The modelling methodologies outlined in this chapter have, at times, been the subject of animated debate, but with no clear evidence for or against the superiority of any of them. They are still geographically distinct: 'scenario modelling' being concentrated in Germany. Statements of these methodologies include very little detail, and in particular, say little about how variables and relations between variables should be derived. This gap in knowledge is the subject of this thesis.

[10] Low expectation in the UK is consistent with my experience as a Chief Examiner. Questions often had to be eased to make them "accessible", and entry to some Universities is also now quite accessible.

Chapter 2

A Review of Computer Algebra Algorithm Development and its impact on Mathematical Modelling

2.0 Abstract

Shortly after their introduction in the 1960s, computer algebra systems were used to simplify the algebra in advanced computations: chiefly relativity and quantum mechanics. They were not used for simpler modelling problems, particularly in an educational context, and I give some reasons here. During the 1970s the use of computer algebra systems extended to other fields, but was still restricted to research. Papers written during the early 1980s showed that authors envisaged significant use in education, but nothing specific in modelling. Inaccurate computation, command-driven front-ends and a reluctance on the part of users to abandon traditional ways prevented educational applications from becoming widespread. Improvements in these features and competitive marketing by some vendors facilitated adoption by the education community, and prompted studies into the efficacy of teaching using computer algebra software. There is evidence from courses developed between 1985 and 1995 to show that relatively simple operations and procedures, in conjunction with graphics and numerics, were used to illustrate mathematical, but not modelling, teaching points. In parallel with this development, the use of computer algebra in other research applications became widespread within a community of experienced academic users. Consequently, applications in modelling did not develop far beyond the stage of doing computations which had been derived by hand. I illustrate the difficulties of modelling within the constraints of simple equation manipulation and calculus. This has left a gap in developing computer algebra tools for modelling.

2.1 The development of computer algebra algorithm

This section includes a brief account of the development of the principal algorithms relevant to symbolic computation with particular reference to those in integration and differential equation solving. It is intended to demonstrate several points:

- that the development of algebraic computation has driven the design of computer algebra packages;
- that there are difficulties in generating and manipulating mathematical expressions;
- an explanation of the origin of common constructs and programming paradigms.

2.2 The Origins of Computer Algebra in LISP

The origins of symbolic manipulation lie with computer languages which were designed to manipulate non-numerical objects, and were developed in the 1960s. The principal one is LISP, for which the primitive object is the list. As a result, the basic objects of computer algebra are strings, symbols and lists.

The LISP language forms the basis for many computer algebra languages, and the diversity of LISP dialects has given rise to distinct computer algebra languages (as discussed in Padget 85). There are two principal dialects of LISP. The first arose from Project Mac and is the precursor for MACSYMA (Pavelle 85). The second has its origin in the Standard LISP Report, and principal descendants are Cambridge LISP (Fitch 77) and REDUCE (Hearn 83). Some researchers who were active in LISP research later turned to symbolic computation and have made significant contributions to algorithm development ever since (Fitch, Wang, Pavelle, Norman and others - see, for example, Marti 83 and Fitch 98). Hence, the programming constructs of modern computer algebra languages are influenced by LISP constructs, and also by procedural programming languages. Not surprisingly, the purely algorithmic process of differentiation was one of the first to be computerised.

2.3 Principal stages in algorithm development

Table 2.1 gives a brief summary of the principal developments in CA algorithms.

Category	Date	Development	Significance	Reference
Calculus & ODEs	1835	Liouville integrationTheorem	Forms the basis of Risch's algorithm and subsequent work on solving differential equations algorithmically.	Liouville 35 Moses 71
	1953	Computerisation of differentiation algorithms in LISP[1]	Consistent version of differentiation rules implemented as rewrite rules in (Knuth 68).	Nolan 53 Kahrimanian 53
	1961	Symbolic integrator SAINT	First engine for symbolic, heuristic integration in finite terms. This is the basis of the DERIVE integrator.	Slagle 61
	1967	Combined heuristic and algorithmic integration algorithm	Basis of CAS integration engines, first implemented in MACSYMA	Moses 67
	1969	Risch algorithm for symbolic integration	A general-purpose decision procedure for symbolic integration in finite terms: the most significant algorithm in symbolic computation to date.	Risch 69 (Moses 71 gives a clear account)
	1971	Algorithms for symbolic indefinite integration	First computerisation of heuristic methods for indefinite integration	Wang 71
	1974	Risch-Norman algorithm	Replaces the highly recursive Risch algorithm with a complicated linear (but not complete) decision procedure[2]. Geddes gives a clear account of its Maple implementation	Norman 74 Geddes 89
	1986	Algorithmic solutions to first order ODEs	Extension of Risch Theorem to ODEs. (Davenport 91) gives theorems and examples	Davenport 86
	1986, 1987	Algorithmic solutions to 2nd. order ODEs	Basis of current algorithms for solving 2nd order ODEs	Kovacic 77 Kovacic 86
	1981, 1992	Symbolic solution of 3rd. order ODEs	Extensions of Kovacic algorithm and use of symmetry groups (now implemented in Axiom)	Singer 81 Singer 92 Bronstein 92

Table 2.1

[1] Davenport (Davenport 88) points out that the real problem with differentiation is how to *simplify* the result (e.g. $\frac{d}{dx}2x^3 = 2*3*x^2 + 0*x^3$ is *not* wanted).

[2] David Stoutemyer reports (July 1998) that the problem of complete decidability of the Risch-Norman algorithm is still unsolved. A finite number of terms in functional forms must be set up in advance, and there is no guarantee that sufficient terms have been included.

Factorisation	1967	Berlekamp algorithm	For factorising a polynomial modulo p (where p is prime) - has limitations when extended to factorisation over the integers.	Davenport 88
	1969	Zassenhaus algorithm	Extension of Berlekamp algorithm, which was inefficient in that the time required to factorise is an exponential function of the degree of the polynomial.	Davenport 88
	1978-83	Hensel's extensions of Berlekamp's algorithm	More efficient than Berlekamp's, and is the basis of the factorisation algorithms in Reduce, Maple and Macsyma. Contributory improvements by Wang (1978,83), Kaltofen (1983)	Davenport 88
Solving linear equation systems	1965	Reducing a set of equations to a Groebner basis	Proofs that solution of the Groebner basis problem is the same as the solution of the original system, and that the algorithm terminates.	Buchberger 65
	1985	Improved Groebner basis algorithm	Increased computational speed	Buchberger 85

Table 2.1(continued)

This chronology illustrates how slow algorithm development has been, and that the major effort in symbolic computation in the 1970s and 1980s has been directed to algorithm development, at the expense of user interface and routine applications. During this period, interfaces were all command-driven and output was principally text. Graphics output was only possible if the video adapter being used supported a graphics mode. Fateman (Fateman 96) indicates the difficulties associated with solving even simple non-linear equations, and these cases are widespread. Detailed analysis of the algorithms show that:

1. They are divorced from heuristic rules learned in elementary courses.
2. Obtaining intermediate results as a means of explanation is not useful: they can only relate to internal algorithms. For example, stages in a Risch algorithm computation would not provide an explanation of 'everyday' techniques of integration. Neither would a recursive heuristic integration algorithm (as in Slagle 61): its recursive nature would be too complex, even for simple integrations.
3. There is a fundamental algebraic problem in that the results obtained do not reflect the underlying algebraic structure from which they were derived. The consequence of this is that results can be wrongly applied in applications (particularly in modelling).

4. Users must take care to consider particular cases, and to provide sensible inputs: the algorithms are not yet sophisticated enough to do this. For example, expressing 2^{10} as an integer should not cause problems, but expressing 2^{10000} as an integer may return an *Out of Memory* message, even though the second input is a trivial extension of the first.
5. They cannot imitate the choices made by humans (e.g. in branch cuts, solutions etc.) in solving equations (Fateman 96).

2.4 Computer algebra usage: consequences of its algorithmic origin

The origin of computer algebra software has had a lasting effect on the capabilities of computer algebra systems and on the way in which they have been used. Pavelle (Pavelle 85A) notes the principal techniques for controlling elementary operations:

- arithmetic computations;
- algebraic functions (e.g. f(x)), based on a rewrite rule $X \rightarrow F(X)$;
- algebraic operations and simplifications (such as a+a+a), often based on repeated application of a rewrite rule such as $(A,n) \rightarrow (n+1)\ A$;
- simple mathematical operations that return a result, given a sequence of algebraic inputs. This class is extensive and includes operations such as Integrate(f, x), Expand(x);
- mathematical functions that return a list such as Solve(equation, variables).

The last two categories are discussed extensively in (Pavelle 85B), where the capabilities of MACSYMA are explored. Although the results of integrations and factorisations are impressive in terms of capability and speed, they completely hide the complexity of underlying algorithms and therefore appear deceptively simple. In order to progress from 'calculator mode', control structures are necessary. These shift mathematical operations into the domain of user-constructed algorithms, as listed below.

2.4.1 Recursion

Even simple mathematical operations often have to be implemented in recursive LISP-like structures, especially in non-sophisticated programming environments such as

DERIVE. For example, Mathematica provides the function `Exponent[p, x]` which returns the degree of x in a polynomial p. No function is available as a primitive in DERIVE, but the recursive nature of the relevant algorithm is visible in a utility file (DERIVE 97). A similar recursive example was given as a supplement to (Stoutemyer 91) in April 1992 (Stoutemyer 92A), and concerned the implementation of the straightforward set operations Union and Intersection.

2.4.2 Flow control by loops

The use of loops in an algebraic expression is nothing new intrinsically, but the computerisation of mathematical processes required explicit looping. This arose from algorithm development and drove the design of the user's programming interface. (Hearn 67) is a good example of the REDUCE code for computing the first 20 Legendre polynomials of x^2-2x. This code also includes many other programming structures: declaration and typing of variables, a formal IF statement and a block structure. Writing this type of code was a new skill for many mathematicians, and still is.

2.4.3 Modularisation

The process of modularisation is analagous to organising mathematical text into theorems, and using the results of these theorems elsewhere. The programming equivalent of the *theorem* is the *procedure*, and computer algebra encapsulates the modularisation process through variables, functions, procedures and packages. Although these elements are not new in programming, their implementation in computer algebra software has unique consequences.

- All created expressions accumulate in memory until they are removed by internal *garbage collection* procedures or by the user. Thus, previous definitions and expressions can remain unexpectedly active.

- The distinction between local and global variables is not always clear. All variables are global in DERIVE, even when they appear as local variables (in a formal parameter list, for example). Formal parameters in a Mathematica procedure argument list are not local variables in the Pascal or C sense, and this can be a problem for C/Pascal programmers who migrate to Mathematica. Maeder (Maeder 94) gives the following peculiar programming trick to emulate local variable behaviour:

```
f[x0_, y0_, z0_]:=
Module[{x=x0, y=y0, z=z0}, (* initialise local x,y,z *)
      ...
      ...
      x = ...    (* use local variables*)
      ]
```

2.4.4 Rewrite rules and pattern matching

Rewrite (production) rules are also fundamental to the implementation of modern computer algebra systems. They are used implicitly to make deductions in mathematical argument (e.g. to simplify $e^x e^y$ to e^{x+y}). This is done with a simple rewrite rule $e^x e^y \rightarrow e^{x+y}$, but rules are not always easy to program because of pattern-matching difficulties. For example, to stress the quadratic aspect of an expression such as $a\left(x-\frac{2}{t}\right)^4+b+c\left(x-\frac{2}{t}\right)^2$ applying the rewrite rule $\left(x-\frac{2}{t}\right)^2 \rightarrow z$ gives az^2+b+cz. The Mathematica code below does not work: the fourth power is not replaced by z^2.

```
exp = a (x-2/t)^4 + b + c (x-2/t)^2;
exp /.  (x-2/t)^2 -> z
```

The reason is that the pattern-matcher is not detecting a general case, and the programmer needs to be aware of how relevant forms are stored internally. The following rewrite rule accounts for this:

```
exp //. b_. + a_. Power[(x - 2/t), n_?Even] -> a z^(n/2) + b
```

The route to finding general expressions such as these is tortuous and does not make for easy programming. The difficulty of implementing this particular rewrite rule has to be contrasted with the 'obvious' pencil-and -paper substitution, which we do almost without thinking. Maeder (Maeder 94) rightly points out that, strictly, there are no formal procedures or functions in Mathematica: they are all rewrite rules, implemented by pattern matching. This programming paradigm originates from LISP, and teaching it needs to be explicit. Dershowitz (Dershowitz 85) gives a fuller discussion of rewrite rules. He differentiates between the use of rewrite rules for LISP-like simplification (as above), as opposed to a Prolog-like computation by completion. The completion procedure, as given by Knuth and Bendix (Knuth 70), was originally developed to generate rewrite systems that can be used to validate identities in equational theories.

2.4.5 List processing

List processing is significant because it appears to be a new technique for mathematicians. However, list operations have always been carried out routinely in 'pencil and paper' computations, albeit implicitly. The most common occurrence is in making an interactive choice from multiple solutions to an equation. Therefore the user's task is to anticipate list generation in automated processes.

2.4.6 Data types and pattern-matching

Data typing sometimes impedes mathematical computation in procedural languages because conversions are not always implicit. A simple example is the non-equivalence of the C declarations `int number1 = 1` and `float number2 = 1.0`. This does not correspond to reality, where they are essentially equivalent. Symbolic computation packages are not immune from this problem, and sometimes contain surprising errors. Mathematica, for example, recognises ½ as a rational object, but not 1, *a/b*, or *a*/2. These are all held internally in different forms. The situation is better with Maple, which can recognise particular algebraic structures, but not in all circumstances (see Maple 89 for factorisation examples).

The combination of user and package can cause conceptual problems in modelling. There is always a temptation to treat any symbol x as a continuous real variable, particularly by using it in calculus operations. The implication is that all algebra takes place in the context of the real number field, even when x is clearly discrete (e.g. Greenhalgh 90 - population dynamics). I justify a continuous approximation to a discrete variable in (Mitic 94), but such justification is rare.

2.5 The influence of computer algebra on modelling

In the previous section I described routine CAS techniques. These have a significant impact on modelling because they are the fundamental processes that carry out the necessary mathematical (and administrative - list operations and modularisation) operations of modelling. They have an even greater impact if modelling processes are automated: the user has to anticipate results when programming. In Chapter 1 I noted that computer algebra has had little effect on modelling, subject to the change noted at ICTM-8. In this section I give examples to show that computer algebra has not always been beneficial.

Some modelling examples quoted in (Harper 91) illustrate this point well. There is a similar example in (Maeder 87). Harper proposes a way of modelling the motion of a particle falling vertically under gravity, first freely and secondly in a resisting medium. The equation of motion is quite simple: $\ddot{x} = g; x(0) = 0; v(0) = 1$. He uses an iterative scheme to solve this equation, which is quite surprising considering its simplicity. The real motivation is to set up an iteration scheme which will cope with an ODE which cannot be integrated analytically. The iterative scheme below is used.

$$x_{n+1} \to x_n + v_n h$$
$$v_{n+1} \to v_n + gh$$
$$t_{n+1} \to t_n + h$$
$$x_0 = v_0 = 0$$

The REDUCE implementation contains modularisation, iteration in the form of an explicit loop and explicit declaration of variables. In Mathematica it would also include rewrite rules. The iteration is then amended by introducing a resistance term, so that the v-iteration is replaced by $v_{n+1} \to v_n + (g - kv_n)h$. Surprisingly, Harper states that the existence of analytical solutions is then more doubtful, even though the equation of motion is integrable. A more enterprising step comes in introducing a procedure that incorporates integration, using the scheme $x_{n+1} \to x_n + \int_0^t v_n \mathrm{d}t$; $v_{n+1} \to v_n + \int_0^t (g - kv_n)\mathrm{d}t$.

This introduces two new elements. The first is the use of procedures, and the second is inference by pattern spotting. The idea is to spot an emerging exponential series. Given the simple ODE in the first place, the value of this is somewhat dubious, especially as no proof of convergence (in any sense) is given or suggested. Using computer algebra software in this example is only useful for performing the integrations.

Otherwise, a spreadsheet, or a black-box `Nsolve[]` procedure, is a better tool. The complication of introducing iteration into this simple problem detracts from the modelling concepts. It appears that computer algebra programming constructs have been introduced simply because they are there.

The programming elements in the above analysis are typical of analyses found in many modelling processes. Recursion, rewrite rules and list processing tend to lie within the domain of Mathematica, where they are used extensively and necessarily, because of a need to manipulate lists. Recent examples are Chandler 97A, Mitic 97, Sakakibara 97 and Jacob 97.

2.6 The influence of Modelling on Computer Algebra Package design

In this section I examine some ways in which design of computer algebra packages has been influenced by modelling. The examples of Reduce and Axiom show how modelling requirements have driven CAS design, and why most CASs in use today are general purpose. This establishes the need for developing dedicated mathematical environments, as is done in the modelling application of this thesis.

2.6.1 Reduce

The origins of Reduce lie in theoretical physics. (Hearn 71) describes fruitful areas in which to apply symbolic computation techniques to problems in theoretical physics. When Hearn's article was written, the Risch algorithm was only two years old, and the pioneering work in finding solutions of algebraic and differential equations was yet to come. Hearn considered that the most important area for symbolic computation was in applications of existing theories to experimental testing. In practice this means developing perturbation techniques to greater and greater accuracy, and their experimental validation. A need for routine computations (polynomial manipulations, substitution of variables and calculus) provided the motivation for developing a general purpose CAS. More specific CASs are now less common: CAYLEY is still used for computations in group theory (Keady 96, Cannon 82). Modellers therefore have to adapt general purpose computer algebra packages for specific purposes.

Specific aspects of Reduce address its modelling requirements. They are: non-commutative algebra (see Thomas 88 for another example), pattern matching, rule definition and efficient integration algorithms. Fitch (Fitch 85) shows how the rule

```
FOR ALL A LET COS(A)**2 = (1+COS(2*A))/2;
```

can be applied in an iterative scheme for finding approximate solutions to a Duffing equation. Rayna (Rayna 68), gives the example of implementing a quarternion algebra, and makes the point that rule definitions are exactly as they would be with 'pencil and paper'.

Lastly, Hearn stresses two points which are pertinent to this thesis and to modelling in general. The first is that setting up a problem and interpreting the results are as important as the computations themselves. The second is that he looks forward to when non-trivial computations in Quantum Electro Dynamics can be done completely automatically. I attempt to automate a different type of mathematical process using a CAS in this thesis.

2.6.2 Axiom

The authors of Axiom, which was developed from the Scratchpad system (Jenks 84), addressed different specifications. Scratchpad was developed at IBM during the 1980s by Jenks, Davenport, Trager, Yun, Miller and others, for testing algorithms for symbolic computation. Scratchpad has now been subsumed into Axiom, development of which was taken over by NAG, Oxford (Jenks 92). The key point about Axiom is that the concept of algebraic structure is central to its design, so that it is a dedicated modelling tool for algebraic analysis as well as being a general purpose computer algebra tool. Axiom 'knows' about the common algebraic structures (ring, integral domain, field etc.), and can prevent the user from making elementary errors, such as trying to invert the * operation in a ring. Axiom outputs the algebraic structure to which the result of an expression evaluation belongs. This dependence on algebraic domains is helpful to algebraists, but not for others: an engineer would probably not be interested in knowing that an output polynomial is a member of a quotient ring of polynomials with integer coefficients ($\frac{1+2x+9x^2}{3}$) rather than a member of a ring of polynomials with rational coefficients ($\frac{1}{3}+\frac{2}{3}x+3x^2$).

2.6.3 MuMath and successors

MuMath, the precursor for DERIVE, is notable because is was designed (and first implemented in 1977) as a small system, fitting on one 360K floppy disk. The DERIVE for Windows kernel is still only 1400K. The result is an easy to learn, cheap and fast CAS, but one which does not lend itself to research. Research needs more than a minimal programming language, and programming constructs tend to be highly recursive due to an absence of memory variables. Recent conferences papers (3rd. Int. DERIVE Conference, Gettysburg, July 1998) show that its current use is mainly in education.[3] Its small size has made it possible to produce calculator ROM chips, so that DERIVE is available on the HP-28 (1993)[4], the TI-92 (1995) and the TI-89 (1998). Uses of hand held symbolic computation 'calculators' are being explored in educational contexts, but nowhere else. One of the first reviews of the capabilities of the TI-92 is (Waits 95), but it is largely descriptive and only gives a brief indication of potential uses. The possibilities for modelling remain unexplored.

2.7 CAS Front -Ends

Dewer (Dewer 92) took a different but related approach in implementing a computer algebra front end (for Reduce) to select appropriate numerical algorithms from the NAG Library for particular tasks. The ultimate task was numeric, but the required decisions were the same that would be required for a symbolic analysis. He addresses issues of continuity, singularities and smoothness (i.e. a measure of oscillation), and has formulated and automated a concept of 'reasonableness'. The CAS analyses which function should be called in particular circumstances. This approach is not new. Roach (Roach 92) discusses interactions of distinct algorithms used in symbolic definite integration in Mathematica, and this type of analysis is fundamental to any symbolic integration process.

The idea of a CAS front-end has been taken further in an AI context (Calmet 91). The aim of Calmet's work was to encapsulate mathematical knowledge by designing a shell capable of representing arbitrary mathematical domains linked by algebraic relations. It

[3] Al Rich (Soft Warehouse) told me of his intention to extend the DERIVE programming language in order to make it more suitable for research - July 1998.
[4] David Stoutemyer thought that this would happen in 1978: he had to wait a long time!

incorporates heuristics, exception handling and knowledge representation systems. There is some doubt about whether or not this system can be used in a general-purpose context because the knowledge encapulation used is context specific.

The front-ends of Dewer and Calmet show how heuristics may be incorporated into a CAS. More research is needed to advance this area of study.

2.8 Difficulties with Symbolic Computation in Modelling

In this section I catalogue some difficulties when symbolic computation is used in modelling. These difficulties impede progress in modelling. Examples of good and not so good practice are taken mainly from the Mathematica course by Brown, Porta and Uhl (Brown 91). This contains many effective uses of symbolic computation, and also anticipates potential problems and suggests solutions (often heuristic). Some difficulties with developing applications with symbolic computation have already been mentioned, and they are discussed briefly here for completeness.

2.8.1 Expression simplification

Several problems of simplification (e.g. arising from differentiation, or rewrite rules for expressions) have already been mentioned. In principle, simplification processes are driven by heuristic rules and strategies. The approach taken in CAS design depends on what the CAS developers consider to be the most effective, and there are three schools of thought. The first is that an explicit function can be called to activate a set of rules (as in Macsyma). The second is to load a package (as in Reduce). The third is to define *ad hoc* rules and apply them as required (as in DERIVE and Mathematica). All of these involve some knowledge of what is required or useful in given circumstances, and the third alternative can cause particular difficulties in constructing rule sets. These are discussed in (Stoutemyer 77). Finding ways to automate simplification strategies for expressions remains an unsolved problem.

A deviation from the above heuristic paradigm for simplification is Furse's Mathematics Understander (MU) program (Furse 91). This is capable of proving theorems in group theory, the inputs being a set of rules with a conjecture, and the outputs being steps in the proof of the conjecture, or a statement that the conjecture is false. This is a remarkable achievement, but its complexity and the development time required (about 10 years up to 1991), indicate the difficulty of the problem. MU relies on heuristics to direct (albeit automatically) the path of a proof, and an objective measure of progress at each step in the proof. The heuristics are programmed as desirable inferences, and this is a weakness of the system. MU also relies on rules sets supplied by the user, and these can be inconsistent.

2.8.2 CAS Knowledge base

Problems of non-propagation of algebraic rules in some CASs have already been raised. To the user, the origin of these problems is not apparent, and they appear the same as some simplification problems. The CAS does not always 'know' of some simplifications, but this is not surprising if simple algebraic constructs are not recognised. Sin[n Pi] ($n \in \mathbb{Z}$), for example, will not automatically simplify to 0 in Mathematica or DERIVE. Although the rewrite rule `Sin[n_Integer Pi] -> 0` can be defined in Mathematica, there is no easy way of declaring that n is an integer. Declaring n to be an integer is easy in DERIVE. Once it has been done, $\sin(n\pi)$ simplifies to 0 with no problem. Another common example in Mathematica is the fact that $a + 1 > 0$ does not follow immediately from $a > 0$. The consequences can be severe for modelling because decisions cannot be automated easily. It also forces regression to numerical work, which dilutes the power of the model.

2.8.3 Process justification

It is very tempting in modelling to make approximations without justifying them. A general-purpose CAS allows the user to make approximations to functions of arbitrary order, and the user has to be aware of potential pitfalls. Brown, Porta and Uhl (Brown 91 page 370-1) provide a good way to sidestep this problem by clarifying general principles on approximation accuracy. They provide an example in which an approximation to $\int_0^{\frac{1}{2}} e^{x^3}\,dx$ is required, and suggest constructing a polynomial p[x] such

that $\left|e^{x^3} - p(x)\right| < \varepsilon,\ \varepsilon > 0$. Taylor approximations provide successive approximations, p[x], to this exponential function. Plots show how good (or bad) these approximations are. Although this approach is empirical, it is much better than simply using a Taylor approximation with no justification of the number of terms used. However, the Gibb's phenomenon (Carslaw 50) is important in such approximations: more terms in an approximation do not guarantee improved accuracy.

2.8.4 Fitting and over-fitting

Brown, Porta and Uhl (Brown 91 page 320-1) also provide an effective demonstration of how a 'better' approximation can be worse, in the context of fitting to discrete data. Data fitting is important for modelling systems which do not have an axiomatic basis, and a common error is to employ too complex a fit, and not to consider whether the result is robust. The divergence of a higher order approximation is very plain to see: a quadratic approximation appears to fit the data well (judging by eye), but a quartic is clearly worse because it is less 'smooth'. A rigorous measure of this deviation is missing, and I suggest a way[5] in Appendix 2B. I also introduce a measure of robustness in this appendix by fitting only some of the data, with the aim of assessing whether or not the fit is in any way applicable to the non-fitted points also. A linear model is clearly the most robust intuitively. This non-rigorous measure is so striking that there is no point in producing supporting calculations. A robust fit, using only part of the data 'in-sample' and extrapolated 'out-of-sample' on the remaining data, gives some indication of the stability of the model.

---

[5] By summing two opposing deviation measures for a fitted curve (they oppose because they compute an error measure at, and as far away as possible, from fitted points):

1. If f(x) is a piecewise continuous linear function that joins data points in sequence. Calculate the sum of the deviations of the curve from f(x) at the mid-point of each adjacent data pair.
2. The sum of squares of deviations of the fitted curve at the data points.

2.8.5 Continuous approximation to a discrete variable

Brown, Porta and Uhl (Brown 91 page 252-3) discuss a predator-prey model, but do not justify using a continuous approximation to a discrete variable. They make assumptions about constant parameters in the model with no justification, but the nature of the variables is not questioned. Happily, symbolic parameters are retained for as long as possible, so the functional dependency on them can be investigated.

2.8.6 Implicit conditions in proof

Proofs often form a significant part of studies in applied analysis. Functions, in most CASs, are simply symbols, and can be manipulated in the same way that any other mathematical object can. Thus, we can differentiate a function f(x) and then use the result f '(x). The result will be meaningless if f violates continuity or differentiability conditions. Devitt (Devitt 89) gives an illustration of good practice. He proves the 1st Mean Value Theorem in Maple (Appendix 2D), starting with two points A and B on the curve defined by a function f. Devitt is careful to state the continuity and differentiability conditions required for this proof to be valid, even though they are not used explicitly. Devitt's use of a specific graph is misleading, as the theorem looks obvious for a continuous, differentiable function on an open real interval. The real task here is to find a case where the theorem breaks down, and violating a continuity condition provides one.

2.8.7 Lack of in-built mechanisms for equation manipulation

In some CASs there is no general mechanism for applying the same operation to both sides of an equation, and this makes routine manipulation difficult. Thus, given a Mathematica expression like `Sqrt[a] == Sqrt[b]`, the result `a == b` can only be obtained with the aid of many lines of program code, such as described in (Riddle 91). Riddle's method is to apply the same operation to both sides of the equation. In DERIVE, such operations can be performed, but they look odd: $\left(\sqrt{a}=\sqrt{b}\right)^2$ does simplify to $a = b$.

2.8.8 CAS Design and User Requirements

Fateman discusses a major CAS design issue in (Fateman 90). He compares Scratchpad II, which works within an algebraic structure with other CASs. Fateman says, in the context of MACSYMA: "I believe that we were unable to find a single, comprehensive, and effective, organisational structure to explain to users and programmers the system's mathematical and computational premises, and their consequences." Thus, he recognises that user requirements are at odds with design and that it is very difficult to emulate implicit processes in human thought. An example is when adding two square matrices. Any algorithm that does this must check the dimension of the matrices (and also compatibility of their elements) and this may not be known until run time. This causes late-binding problems which are also apparent in exclusively compiled O-O programming languages.

2.9 Inappropriate use of a CAS

It is easy to use a CAS badly. This section contains an example of using a CAS when an alternative technique is better. In (Iglesias 97), Iglesias and Power implement constant and linear element BEM techniques to solve a simple boundary value problem in (Brebbia 78).[6] The Mathematica implementation has several problems.

1. The code is too slow (and will fail!) when used for non-trivial applications.
2. It is faster to evaluate the integrals numerically, and symbolic techniques are not needed.
3. The Gaussian Quadrature technique used is superfluous: it is subsumed into the Mathematica function `NIntegrate`.
4. The BEM is typically used to calculate the value of the potential and potential gradient at interior points as well as on the boundary. This increases the number of nodes considered. The Iglesias-Power implementation does not do this, and including more nodes would decrease the speed further.
5. Iglesias and Power are mistaken in stating that symbolic computation programs can "carry out the calculations with infinite precision". This is clearly impossible.

[6] Comments are based on my 1996 Mathematica implementation of this technique.

There has to be some positive advantage in using a CAS in modelling, and it must outweigh disadvantages. Finding a suitable BEM application remains a matter for further research.

2.10 Educational applications of computer algebra

In this section I discuss the principal ideas which were advanced between 1975 and 1985 for applications in learning mathematics and modelling, and highlight some barriers that were recognised then. These ideas are still advanced today with little change. I also discuss some misconceptions and important considerations which passed almost unnoticed more than ten years ago.

Stoutemyer (Stoutemyer 84) provided several arguments as to why procedural programming languages (Fortran, Pascal etc.) are insufficient for learning mathematics, and argued that computer algebra software could fill the gap. His arguments partially explain why this type of programming has not entered mathematics teaching.

1. Inputs and outputs in Fortran, Pascal etc. are necessarily numerical.
2. Programming is an algorithmic process with few parallels in elementary mathematics.
3. Numeric computation is limited by machine precision, which causes round-off error.
4. Typing introduces conceptual incompatibilities (e.g. 1 (integer) is not the same as 1.0 (floating point)).
5. The 'language' of computing (binary and hex.) does not correspond to 'normal' language, decimal.

Given that Stoutemyer argues that procedural programming languages cannot do symbolic manipulation, it is not surprising that his solution is a CAS, which can. Stoutemyer saw the following items as potential targets for computer algebra in education in 1984.

1. For CAI (a role that MathWise can perform (see Chapter 3)).
2. To provide enrichment and motivation: rules can be discovered and conjectures formulated (but proof is much harder).
3. Preparation and checking of teaching materials.
4. In automated production of examples and testing, and in intelligent diagnosis.

Of these, only the first two have been discussed at length since. Stoutemyer predicted an effect, for better or worse, of computer algebra on mathematics and modelling, but does not elaborate on his comment.

Stoutemyer does not say:

- what curriculum changes may result;
- what the effects might be on algebraic skills;
- that programming skills (particularly for recursive routines) are hard to gain.

Hosack adds the following advantages of symbolic computation to Stoutemyer's list in (Hosack 84), but there are problems associated with all of them.

1. The user can concentrate on concepts rather than algorithms (although he does not say that this is often harder).
2. Greater realism is possible (but see Mitic 94B).
3. There is a specific use for numerical methods involving asymptotic series (this really lies within the domain of applied mathematics research).

Calmet (Calmet 84) provides empirical evidence from physics students for some of the problems discussed earlier in this chapter. His students considered that learning a new programming language is only worthwhile if the benefits of doing so are totally visible from the start, and that suitable teaching is required. The first point may reflect the limited capabilities of CASs and the difficulty of using them in 1984.

Aspetsberger and Funk (Aspetsberger 84) showed that some of these problems can be overcome. They succeeded in teaching Austrian High School students transformational geometry and number theory with MuMath. They report that they even succeeded in teaching functional and recursive programming. Such success is surprising unless the problems solved were extremely simple. Their success is likely not to be generally applicable: the sample used was small and had special treatment.

As a result of Maple trials with students at the University of Waterloo, Char (Char 84) echoes the 'immediate payoff' requirement that appears to be sought by some new users. Char also states that Maple was not powerful enough at the time. He argues for an integrated environment for mathematical processes and against a command-driven interface. Both of these issues are addressed in the software for this thesis. Char also mentions a common side effect, which is to have unrealistic expectations of a CAS by attempting extremely large (in terms of memory required) computations which require

the same order of input as a much smaller computation. A very trivial example is the calculation of 20! and 20000!.

Thomas (Thomas 88) advances the idea of easing the effects of a command-driven interface. He suggests additional software to supplement a CA engine (Reduce in this case) to pre-process sets of axioms which produce Reduce rules. This is a way of extending the programming language. This effective technique has been used in a number of contexts:

- To pre-process input files for the software developed in this thesis, thus avoiding syntax problems.
- To process exported expressions, and process them in a way that is not possible within the syntax of the CAS (Mitic 94A).
- To speed up numerical calculations (MathLink 91).

2.11 Summary

The following points summarise problems that arise from the way computer algebra systems developed, and how this development has affected modelling.

The origin of computer algebra systems in LISP has resulted in programming constructs that are recursive and require list manipulations. These appear to be divorced from mathematical operations, but are necessary for routine manipulation of expressions. Similarly, programming constructs such as rewrite rules, data typing and modularisation cannot be avoided.

Since the 1960s, computer algebra research has concentrated on algorithms, and significant user interface development has only occurred during the past six years. The result is that computer algebra systems have been difficult to use. Many users in the 1960s and 1970s were researchers who needed computer algebra tools to simplify complex and lengthy manipulations in algebra and calculus. They were motivated to overcome the difficulties of using a CAS, and found that a general purpose mathematical tool was more valuable than dedicated tools. This causes problems for mathematical modelling. Modellers can use a CAS to perform the routine

manipulations which arise when solving equations, but using a CAS to formulate the model is much harder.

Once attention shifted from algorithm development to applications (in the 1980s and 1990s), CASs gained favour in education as well as research. Many researchers used computer algebra in an inappropriate way: it was possible to do manipulations without regard to algebraic structure or pathological cases. There was also a tendency to use them without considering whether or not they were the most appropriate tool.

I end this chapter with a fallacy and a comment. Lichtenburger (Lichtenburger 84) states that symbolic computation processes can be 'stepped through' in order to gain some insight into the process. This section has demonstrated that the algorithms used in CASs are not appropriate for this purpose. None of the authors in this chapter mention one further problem: the construction of modelling problems 'suitable' for analysis with a CAS. This problem is essentially the same as constructing numerical problems where terms cancel. It is discussed in (Mitic 94B).

Chapter 3

Teaching and learning mathematics and modelling by Computer

3.0 Abstract

This chapter gives a summary of the general issues which can influence learning of mathematics through the medium of computer algebra. Research in this area is limited in scope and focuses on theoretical rather than practical issues. It therefore fails to address issues arising from the use of software in mathematics. Using software is not always beneficial because traditional methods may achieve the desired result faster, or without a considerable programming overhead, or may account for algebraic factors that the CAS cannot account for. The chapter concentrates on computer algebra laboratories, for which I give the most comprehensive review to date of quantitative studies of the efficacy of learning mathematics. Some of these studies contain insufficient statistical information to support any rigorous analysis. For those that contain more sophisticated statistical analysis, the experimental conditions and the statistical basis for the conclusions reached is questionable. Conclusions in many of these studies are therefore doubtful. When taken as a whole, these studies indicate that computer laboratories have an overall benign effect on learning. From considerations of these studies, I establish a case that the use of educational computer-based technology, and computer algebra in particular, is so fundamentally different in approach to a traditional presentation that any direct comparison is ill-founded. I propose that a package comprising the computer tool, the overall pedagogic approach, training and teacher enthusiasm, all acting together, is the important factor in determining success over traditional approaches. Therefore any such comparison is inherently unsatisfactory: the precise details of what is being compared is not well-defined. Devising a fair test for comparing a computer algebra approach to a traditional approach remains a subject for further research. In order to test the specific effects of the techniques devised in this thesis, I adopt an alternative approach which avoids a direct comparison of achievement.

3.1 Non-mathematical issues which influence learning

A number of learning models are applicable to mathematical learning and modelling. The following discussion shows that these models present some theoretical issues but tend not to pursue practicalities. Two such models are discussed in this section. In addition, learning is also influenced by the computer itself and more general environmental issues. This section contains a discussion of recent papers on these topics.

3.1.1 Learning Models

Norman and Pritchard (Norman 94) discuss the learning model originally due to Cocking and Chipman (Cocking 88), and using a Kruketskii approach to learning difficulties (Kruketskii 76). In this generic model the learning process is modified by three factors: *entry mastery* (which contains the theoretical constructs for this model), *opportunity to learn,* and *student motivation*, the last two categories being important but largely uncontrolled. The authors sub-divide the *entry mastery* category four ways: *mathematical concepts, language skills, reading* and *learning ability*. They claim that successful problem-solvers can generalise and are flexible thinkers. Discussion of how existing ideas might be applied to another context or of how a completely new model might be generated appears to be missing. Flexible thinking involves applying a successful solution process for a previous problem to a new problem. The authors state that a good problem-solver knows when and how to do this, and also when a previous solution is not appropriate to a new problem. This does not help the acquisition or development of any such knowledge. Norman and Pritchard also discuss two cognitive obstacles to learning: linguistic/representational obstacles and intuitive influence. Linguistic/representational problems involve misunderstanding notation [e.g. $\sin(\cos(x)) \rightarrow \sin(x)\cos(x)$]. This situation is hard to distinguish from the more serious problem of the non-legitimate application of a rule such as $\int fg = f\int g$. It is therefore difficult to distinguish notation difficulties from non-understanding, which does not help us to understand the latter. Norman and Pritchard point out a further difficulty with notation: processes that appear to need an action. There appears to be a compulsion to perform the action, even if this is impossible because a closed form has been presented.

Thus, finding F(a) where $F(x) = \int_{a}^{x} f(t)dt$ "cannot be done" because f(t) is an unknown function. The authors do not point out that this misunderstanding might be considered as a fundamental misunderstanding of algebra, and hence of mathematics. The second cognitive factor, intuitive influence, often applies to graphical problems, with the misconception that graphs must be continuous. There is sufficient evidence (discussed below) to suggest that a CAS can overcome such obstacles.

3.1.2 The effect of Computers on Learning Processes and Concepts

Some evidence for the efficacy of computers in a learning context is provided by Tall (Tall 92). In his learning model he attempts to isolate mathematical process rather than abstract ideas. For example, the symbol entity "A+B" refers to both the concept of summation and the addition process. Tall suggests that this functional approach leads to alternative strategies which have greater problem-solving potential. These are unevaluated conjectures, so there is no evidence that they are anything more than a good idea. Tall considers that the computer can aid learning by automating algorithms, and that there are two problems with this approach. First, the student needs to construct a meaning for symbols used, but he does not say how this may be done. Second, manipulation of these symbols should correspond to the way in which the student would do it. This is incorrect in some cases, the most notable of which concerns the Risch integration algorithms (Roach 92, Risch 69). A Risch approach would never be used to teach integration because even simple examples would be long and complicated to implement by hand. Similarly, a simpler pattern-recognition approach which involves recognition of algebraic forms (products, quotients, functions etc.), such as is used by DERIVE (Stoutemyer 91A) would also be complicated to implement by hand in a learning context.

In a later article (Tall 93), Tall and Razali put these ideas into context in a study of the difference in performance between low and high attaining students moving from school to university in Malaysia. They make a firm distinction between processes and concepts, and hence suggest improvements for the learning process. Hence, this study is more removed from mere abstractions, and can therefore be regarded as more significant. The authors uncover a difference in qualitative thinking between those who succeed and those who fail in mathematics, and suggest methods for remediation. The

study was done in 1988 with 350 students, but its generality is open to question because of the influence of national characteristics. The authors propose that weak students can do little more than apply processes, sometimes incorrectly. They need to remember more because they can reconstruct less, and repeated practice is unlikely to improve their performance. Stronger students can manipulate concepts with greater fluency and think in terms of mathematical objects (such as structures or equations). However, there is no direct evidence that objects had been recognised by strong students but not by weak students. In addition, the idea that repeated practice at the process level is unlikely to improve performance has not been tested and contradicts the accepted view that "practice makes perfect". More research is required here. The authors suggest that giving pre-course remedial teaching is needed for weak students. It should include revision, and extension of mathematical procedures. This is inconsistent with their stated finding that more revision does not necessarily improve understanding.

3.1.3 Reactions to problem-solving situations

Rosamond (Rosamond 94) studied emotions felt by novice and expert problem solvers, in which a fundamental observation stage appears to be uncontrolled in that it resulted in a non-homogeneous set of observations. For example, judgement of 'displacement activity' is subjective: apparent inactivity can clarify thoughts. The word "do-able" was often used, but in an ill-defined sense. In early phases there was some reluctance to use "advanced" methods because it was felt that this was cheating. This appears to have been an unexpected result and could have a significant effect on findings. It is surprising that more negative attitudes were not reported. The reactions observed were possibly more typical of US students and faculty, and there is no evidence of similar findings in other countries. This study relies on particular, unjustified, interpretations of observations. In particular, the effects of observer interaction are not quantified. The study does show that certain elements are needed for success in mathematical problem solving. These are understanding the problem, having a strategy with which to solve it, having sufficient confidence to tackle the problem, having a challenging problem, persistence and positive feedback. Rosamond did not record any more fundamental conclusions.

3.2 The influence of computers on Mathematics

Tully (Tully 96) studied specific ways in which computers can influence education in Germany. He considers that computers can change learning processes and that learning processes become more knowledge-oriented. This is not surprising if new techniques are available. Schools are not always able to adapt well to such new methods (again, not surprising). The German government had made a formal decision to teach elementary IT in German schools as early as 1984 (Bund-Länder-Kommission für Bildungsplänung und Forschungfölderung: Gesampkonzept für die Informationstechnische Bildung. Bonn 1987), and efforts were made in Germany to rectify deficiencies. It appears that basic knowledge ("input", "file", "drive") can be easily learned in school. However, functional knowledge (fluency in use) tends to be learned at home, through radio, TV, magazines, CD-ROM and peers. The authors do not state some implications of this finding: schools are unsuccessful in providing fluency in the skills concerned, and not every home can provide suitable equipment. It is therefore necessary to ensure that students have sufficient fluency in handling computers before they start to learn a hitherto non-computerised process. If this is not so the computer merely impedes progress. This observation is very significant because it means that the computer is not an automatic solution to learning problems, as appears to be the thought in other studies (e.g. Beilby 93). Tully's observations are relevant when validating the software described in Chapter 7 of this thesis. A modelling environment involving computers is far removed from 'traditional' modelling, and presents a potential barrier to learning in a computerised environment.

Dubinsky (Dubinsky 92) suggests that beliefs about learning lead to particular choices about teaching. For example, the idea that 'learning by doing' implies an exploratory approach to teaching rather than an approach based on fact finding. Sample problems indicate that he prefers teaching based on "doing" rather than "observing". He considers that if the applications component in teaching is too large, there can be a loss of generality: the user can fail to see a wider context for the theory and techniques involved. To compensate, there should be significant reflection on the construction process. This seems to be a conviction and remains unsupported. He also states that each topic must be analysed in order to find what constructions are necessary and relevant problem situations can be designed afterwards. This approach has two major

drawbacks. First, an over-emphasis on hierarchical structure can obscure an overall view of a topic. Second, this can lead to an artificial problem solving situation in which the method of solution for a problem determines the problem rather than the other way round. The overall effect is scenario modelling. Overall, Dubinsky's study only illustrates that a particular approach to learning affects teaching, and the context described is relatively standard.

A quantitative study of students' attitudes towards computers was made by Selwyn (Selwyn 97). He used factor analysis to extract and classify principal features which contribute towards computer literacy. His input factors are attitude towards computers, perceived usefulness, perceived control, behavioural attitudes, and perceptions and information regarding computers. The factor analysis demonstrated that these factors are statistically orthogonal, but it would have been more interesting to take a random collection of inputs, apply the factor analysis and then extract and explain the attributes of the principal components. Selwyn does not suggest how to improve attitudes.

Mayes (Mayes 98) provides a current quantitative study of student attitudes and belief in response to an unacceptably high drop-out rate (50%) in some College algebra courses. He uses a model in which emotions develop into attitudes. These attitudes develop into beliefs, and the process then works cyclically by generating further emotions. He compared attitudes and belief in a traditional algebra curriculum with a reform curriculum (Mayes 98A), which is a 'package' of written course materials, computer work, and a different mode of working for lecturers and students. Students completed 'before' and 'after' questionnaires on attitudes to mathematics in order to test seven distinct factors, including applications and experimentation. A thorough statistical treatment using a MANCOVA analysis revealed that there was no overall significant improvement in attitude due to the reform 'package'. Mayes suggests that the reform curriculum may be too demanding (for both students and lecturers). He has not yet tested this conjecture. He also comments that some attitude factors show improvement that is not statistically significant. This is a common conclusion in many quantitative studies discussed in this chapter, and is a very weak justification for using a computer algebra curriculum.

3.3 Embedding a programming environment in a learning process

In this section several examples of mathematical learning processes for which the computer is the essential contained element are discussed. Each typifies either the extent or complexity of the software or the extent or complexity of the embedded mathematical processes.

The study by Noss (Noss 97) is a recent example of how a learning process can be "fitted" with a computer-based learning environment. It is notable because of the way in which the computer amends the learning environment. The study is a simulation of a pattern spotting activity using a dialect of Logo, (Dynamic Algebra System). Noss has been using "MicroWorld" environments with Logo for many years (Hoyles 87 contains several examples), and this application merely replaces a simple pencil and paper activity by a computerised equivalent. In this case the purpose is to elucidate an algebraic formula. The associated physical activity (arranging matchsticks in patterns) is so simple that is not readily apparent what the computer is for, especially as a large programming environment and interface have been built for it. The authors state that they use computer resources to abstract algebraic principles and to connect an algebraic activity to virtual objects, and that they are not computerising a pencil and paper activity. Several important points remain unanswered by this comment. First, it does not appear to be an adequate justification for using this software. Second, the precise algebraic principles which are abstracted are unclear. Third, and most important, is why actual matchsticks are not adequate for this task: the computer resources described appear to be very costly in terms of effort required to construct them. Indeed, most mathematical learning at this level proceeds from the abstract to the concrete because it has long been considered that this is the best way to learn in the Piagetian 'concrete phase' (Piaget 78). Fourth, the software clearly does computerise a pencil and paper activity. A serious weakness of this study is that there is no attempt at exhaustive testing or validation. This "MicroWorld" is used with two subjects only, and the conclusion drawn is that it fulfilled its purpose. This conclusion cannot possibly be sustained on any statistical basis, and no criteria for success were established. I conclude from studies of this type that using particular software packages can be taken too far: Noss did not show that a computerised approach was superior to any other approach.

More wide-ranging, but still designed for a dedicated purpose is the Scenario Design Tool, SDT, (Li 96). This is designed for teaching mechanics. Interactive simulations in mechanics can be created and edited. SDT is a derivative of Smalltalk and is capable only of numeric computations, which is a disadvantage because functional variation cannot be studied. Essential calculus processes also have to be cast into numerical terms. It is however an object-orientated system, which makes sense in the context of mechanical objects. An 'F=MA' object encapsulates the acceleration and velocity characteristics of the physical object with which it is associated, but the algebraic properties of the F=MA object remain hidden in the Smalltalk code. This is not desirable in a system that emphasises the way in which objects in mechanics interact, which is fundamentally expressible algebraically. No evaluations of the use of SDT were reported, so there is no evidence that students who use it are better off than those who learn by non-computer methods.

3.4 Larger software systems for modelling

This section assesses some larger dedicated modelling software systems. *MathWise* in particular is increasingly being regarded as a standard teaching tool. They either complement or replace teaching by computer algebra, and are therefore alternative modelling tools. Such software systems have been designed as working software and are not intended to illustrate any theoretical viewpoint. This section discusses advantages and disadvantages of using dedicated modelling tools instead of computer algebra.

Greenman (Greenman 94) describes the simulation package Stella (Stella 94). This package is for modelling numeric simulations in dynamic systems, and is intended to motivate people who know little mathematics in modelling systems which are mathematically complex. An icon-driven front end is used to construct a network of objects and pipes, which link the objects. This type of interface is a considerable advantage when constructing a given scenario as it requires subject knowledge of system interactions but not of complicated syntax or mathematical relations. However, if the user has no, or very little, appreciation of the mathematical basis of the simulation, he or she may not fully appreciate interactions of parts of the system in a quantitative way. If this is the case, only general conclusions can be drawn about system behaviour, and non-systematic use of numerical data can lead to recognition only of pathological

cases and over-fitting. The user may also erroneously think that an "answer" obtained is "correct": it is at best approximate and may have a considerable error margin. No formal evaluations have been done, but this would not be possible anyway since this type of study would not be possible without the software. The idea of developing a model by linking objects is the basis of the software developed as part of this thesis.

MathWise (Beilby 93) is a computer-based environment for learning mathematics which is wide ranging in extent and type of course material. This multi-media approach takes the form of an animated text in which symbolic and arithmetic computations can be done. From Beilby's article it could be concluded that a major justification for MathWise is that it can replace traditional lectures and lecturers. Much quantification of the time required for lecture courses and remedial work is made, but he does not quantify the work required to produce MathWise units. These must be considerable because complete lecture courses must be cast into a programmed environment, which is time-consuming. Conclusions based on cost savings are therefore fallacious. He makes the extraordinary comment that MathWise course materials "contain the Mathematics that lecturers do not want to teach, and leave them to teach what they want to teach!". Since courseware modules cover basic lecture courses this appears to imply that lecturers do not wish to teach fundamental materials. Beilby's comment reflects badly on attitudes towards student learning and, it is to be hoped, is not consistent with the intended purpose of MathWise, which is to make course materials more accessible to a wider range of students. There is no formal evaluation of the success or otherwise of this courseware compared to traditional lecture courses, and the efficacy of this type of teaching approach remains an area for further research. Ironically, it may be testable by a trial in which MathWise is replaced by traditional (lecture) teaching for an experimental group, since its value appears to have been taken on trust.

Features of the MathWise units are typified in the astronomy unit, reviewed in (Harper 96). This is an introduction to astronomy which, in the opinion of the reviewer, makes good use of mathematical techniques, and also uses astronomy to illustrate practical mathematics. This comment is not entirely justified as mathematical bases tend to be hidden from the user, who sees animations and the results of computations. Harper's main criticisms of this unit are that some relevant course material is missing, consistency is lacking, and there are conceptual and programming bugs.

The last of these is readily apparent if one looks at only a few modules, and reflects hurried and badly tested software production. I provide further criticisms.

1. Information and explanations cannot be changed: there is no way of expressing the same information in a different way. (This is what a lecturer can do)
2. Self-study is often too slow, or there can be a tendency to rush and not absorb material. Either way, student control is not always a good thing.
3. There is a tendency for text to be a summary. Even if there are references to more detailed texts, these could be provided by an organised reference list.

Thus, the real strength of an 'authorware' approach to course materials lies in the interactive aspect which can be hard to provide in a traditional course. The effects of a wholesale move to authorware remains untested.

Mackie (in Mackie 97) reiterates earlier comments (in Mackie 92), to the effect that the key attributes of CAL are self-study, immediate feedback and improved motivation. Her 1997 study of Mathwise still does not establish the superiority of these attributes. She presents data which indicate that feedback is mainly positive, but there are no control data to make a comparison. Although her students did like some aspects of Mathwise (flexibility, visualisation capabilities and the chance to experiment), they did not like insufficient and inflexible explanations, nor a lack of contact with lecturers. This exposes a general weakness of CAL: it can only do what the programmer anticipates.

The simulation package ARENA (Fitzharris 96) uses templates for modelling scenarios involving discrete queues, with particular reference to manufacturing flow networks. As such it is necessarily numeric and is dedicated to this type of scenario. It is object-based in that objects, termed "entities" (parts, customers etc.) which flow through the system, have "attributes" in an object-oriented sense. However, the precise nature of the objects created and of the connections made are not readily apparent to the user. The mathematical aspect is thereby diminished, which makes this less useful as a mathematical tool. Modelling with objects is very natural in this context: there is a 1-to-1 relationship between physical objects in a real system and logical objects implemented in software.

3.5 Learning and assessment

In this section I review literature on assessing the use of computer algebra in mathematics teaching. This has an impact on modelling because the basic tools of modelling are the same mathematical techniques and knowledge which are increasingly being learned with the aid of CASs. This section starts with a discussion of factors which influence learning and assessment, then discusses informal assessment of the use of CASs, and ends with a review of quantitative studies of the use of CASs.

3.5.1 Factors influencing learning and assessment

References to studies on the quantitative effects of teaching and learning of computer algebra mostly originate from the United States, where teaching using computer laboratories has become widespread over the past 10 years. Two principal factors, teacher enthusiasm and student motivation need to be removed in order to draw objective conclusions. Many students may be motivated by the novelty value attached to using a computer to do mathematics. Watkins (Watkins 96) supports this view. These factors are only hinted at in most (and particularly early) studies, and this casts doubt on the results reported.

A study of student reaction to and the results of a tool to program a spreadsheet was carried out by Beare (Beare 96). Informal student feedback from only 12 students showed that they found it helpful to have examples of the use of this tool and that they would have preferred more time to learn how to use the tool for themselves. The sample size is too small to be statistically significant and the study lacks quantitative information. The modelling technique was also flawed: data fitting was in-sample only.

Meyer and Parsons (Meyer 96) give a summary and analysis of factors which influence learning processes in mathematics. They proposed 6 factors which they thought were important for mathematical learning. Students were interviewed to see if these factors were actually present. The six factors were:

1. Relating: applying a context or a purely abstract approach.
2. Problem-solving: use of algorithms, pattern recognition and non-sequential approaches.
3. Integrating: concentration on specific examples; using data to develop principles.
4. Practising: copying examples and working through similar examples.
5. Translating: symbols into words and abstracting relations in terms of symbols.
6. Explaining: talking about problem-solving; performing a pencil and paper activity.

Meyer and Parsons found that all of these factors apart from "Integrating" were present in the students' problem-solving activities. It is not sufficient to conclude that these factors were present without some quantification of their relative importance. It appears that no "other" category was suggested. Even if it had been, it is often difficult to find a suitable response. Ideally, a survey should not lead by suggesting factors, as appears to have been done here. Pattern recognition appeared as a distinct problem-solving process, possibly because mathematics is often taught in terms of solving many similar problems, all corresponding to the same pattern. It would be interesting to see if a "pattern matching" attribute could be found in another subject if this method of teaching is applied. The process of "integrating" includes two processes: using discrete data to develop principles and operating within the context of those principles. Research from other authors indicates that use of a CAS can enhance the "integrating" attribute (e.g. Palmiter 91). Meyer and Parsons consider that the process of "integrating" may involve concepts or strategies that only become explicitly clear later in the experience of learning mathematics, and this is why the "integrating" attribute was not found in this analysis. This conclusion seems to be more consistent with their explanation of the "practising" attribute, in the sense that work on more advanced topics is said to reinforce earlier work. Further research is required to support this view.

3.6.2 Informal assessment of the use of computers in learning

Some informal attempts at assessment of computer-based learning have been made in the past few years. These largely provide anecdotal evidence that users react favourably to using a CAS, but tend to give only cursory details and explanations.

The study of Neill and Curran (Neill 94) is of this type, and is particularly unfortunate in that they employed a large sample (486) but made insufficient use of it. A positive but unquantified response was recorded from the students, and this response was not attributed to any contributory factor. They made no attempt to isolate any such factor.

Maclean and Scotney (McLean 94) advance matters slightly in describing computer-based training using Minitab. There is no formal statistical significance testing, possibly because there was no control group. After 70 students had returned evaluation forms, two thirds found computer tutorials had been helpful, but 44% said that they would not use computer-based training again if they had the chance. This is insufficient to draw valid conclusions from. The authors state that it was "hoped" that computer-based training would overcome this problem. This study really illustrates the way in which the benefits of computer-based learning are taken largely on trust, with only informal evaluation of the usefulness of computer-based learning.

Draper (Draper 96) summarises studies of performance and effectiveness of CAL. His study illustrates several general problems in measuring the effectiveness of CAL. It is significant because it gives detailed descriptions of critical evaluations of the methods used and much empirical evidence. However, it is still not quantitative, despite pre- and post- knowledge assessments. Students filled in "confidence logs" before and after a CAL session. These are self-assessments to test their confidence in grasping principles and ability to perform tasks. It was found that these confidence logs were necessary but not sufficient to show a positive effect. This remark is important because it indicates that the authors were very aware of the fragility of evidence based on such returns. The researchers report several problems in their measurements, including bias due to non cooperative participants, low expectation of lectures coupled with higher expectations for CAL, and an inability to isolate the effects of CAL from other influences. Draper does not suggest any way of circumventing these difficulties.

Gunn (Gunn 96) considers that Draper's approach to CAL evaluation is not appropriate, and claims that the above problems are unimportant. She questions the application of rigorous scientific experimental methodology to CAL. As an alternative to pre- and post-testing she advocates the use of SECAL (Situated Evaluation of CAL), claiming that using SECAL meets all the criteria for evaluation study design that Draper found hard. This appears to be based more on faith than reasoned argument, and she advances no arguments as to why Draper should be wrong. SECAL involves obtaining continuous feedback after using CAL for narrowly-focused tasks. This method appears to make no advance at all, since key components of motivation and cooperation are not recognised as problems.

Thus the problem of isolating influences other than the particular computer-based learning tool in question remains. Designing a fair trial for the computer-based software, based on elimination of those influences, has not yet been done. Alternatively, it may be necessary to look for other rationales for computerising mathematical activities.

3.6.3 Quantitative assessments of the use of computer algebra in learning

Several studies have been carried out in order formally to test statistical hypotheses that students taught using a CAS perform better than students taught by traditional methods. Many originate from calculus laboratories which are now commonplace in the United States, and are therefore subject to bias from teaching methodology, teaching culture and syllabi employed in the United States. It would be interesting to see the results of some of the trials described below, applied in Europe and Asia. I suggest that the work ethic in Asia is so strong that a CAS would not appear beneficial in a comparative trial. Results in Europe are harder to predict. A weak work ethic is unlikely to result in any measurable improvement.

In some studies only the sample sizes and the means of scores obtained are recorded, in which case no formal significance test can be applied. It is difficult to see why standard deviations are not always recorded, since they are easily calculable from raw data. Usually sample sizes for experimental groups are small (typically less than 30), and population parameters (usually the mean and variance) have to be estimated from experimental data. Either a *t*-test or a non-parametric test is then appropriate to assess significance. In the case of large samples (>30) a *z*-test is preferable to a *t*-test because no assumption about a background population (of test scores) is necessary, provided that population parameters are known. Usually they are not, but approximating them from data is usually sufficient. If a *t*-test is used two assumptions must be made:

1. independent random samples are employed;
2. the background population must be Normal.

In most cases formal studies do not attend to these assumptions. Therefore not too much reliance should be placed on these results from a statistical point of view.

An earlier quantification of the effects of teaching mathematics by computer is due to Wenger (Wenger 88). Wenger considers that books tend to present apparently unrelated topics. Learning is expected to happen through repeated practice using these examples, but inferences are left for the students to discover. This does apply to some texts, such as (Watkins 93, Leinbach 91 and Glynn 90), but they contain 'leading' exercises. There is evidence to suggest that this 'discovery' method reinforces conceptual understanding (e.g. Palmiter 91 and Klinger 94). Wenger's generalisation is not true for the text by Brown, Porta and Uhl (Brown 91) which has an abundance of examples, but often states

results and methods, even before illustrating them with Mathematica. The texts by Arney (Arney 91) and Berry (Berry 93) are similar. They contain a clear exposition of theory. Wenger attempted to measure achievement as a function of computer usage. Heavy users of the computer performed worse in three out of six gradings in pre-tests, so it is clear that the amount of computer usage has no beneficial effect on attainment. From post-tests there is informal evidence that the heavy users of computers obtained better grades than the light users of computers, but Wenger does not quantify this. Chi-Squared analyses formalise these results. In each case the measured value of Chi-Squared is significant at the 1% level, showing that heavier usage of the computer has a beneficial effect on gradings. (see Appendix 3W). This is not surprising: the students worked harder! Wenger also collected data to assess the effects of the number of diagnostic tests taken on the number of passes obtained. A Chi-Squared test shows that there is no evidence, at the 5% level, that diagnostic tests have a beneficial effect on passes obtained in subsequent tests (see Appendix 3W). In the surprising opinion of Wenger's co-workers, these results are not solid enough for serious diagnostic purposes in mathematics, despite the large samples involved (about 250). This illustrates a general finding in this type of quantitative analysis: weak significance in favour of using a CAS. In this case the reason for the doubt expressed about the validity of the result is unclear, but may relate to bias due to motivation.

The paper by Hillel (Hillel 92), contains an example of a claim that use of a CAS improves performance, supposedly backed up by simple percentage pass rates in a final exam. Despite Hillel's claim that he had not set out to evaluate the educational role of his CAS (Maple), he produced extensive informal student evaluations and measured pass rates in an experimental group (size 18) and a control group (size unstated). The average marks for the experimental and control groups were 65% and 53% respectively, and this is said to be an overall improvement in performance. However, Appendix 3H1 shows that this conclusion cannot be justified statistically on the basis of a sampling proportions test.

Heid's study (Heid 88) produced contradictory results. Experimental and control groups differed in treatments. Their courses were of different lengths, emphasised different skills, studied different subject matter in varying depth, had different pre- and post-tests, had different degrees of coaching, but took the same final examination. The versions of MuMath (and Mathematica) used also produced incorrect results in some cases (Mitic 91). Some reasons for non-equivalent experimental and control groups are not supportable (e.g. time-filling for the experimental group), but the implication is that it is inappropriate for the two groups to be treated equally. Appendix 3H contains tests, which Heid does not do, for differences of proportions based on her "quiz" data. Accumulating results produces a significant result with single-sample *t*-tests (the one for her 'Final exam' data is in Appendix 3H), but this is over-fitting. The summary and sample tests for differences in proportions in Appendix 3H show that only a minority (10 out of 32 in the 'Quiz' and 0 out of 18 in the 'Final exam.') of experimental groups significantly out-performed their control groups. Furthermore, 11 experimental groups performed worse than their control groups. Heid's conclusion, that students in experimental groups were better able to answer conceptually oriented questions cannot therefore be substantiated on statistical grounds, or on the grounds of the experimental design. The summary results in Appendix 3H also show that her claims certainly do not extrapolate to subsequent tests (the 'final exam').

Mayes (Mayes 97) provides a review of research since 1988 into the effect of using computer algebra in an educational context, and summarises 15 dissertation findings. He states that the problems with these studies are that some are not refereed, they are inconsistent in approach, they concentrate only on calculus and algebra, and the research methodology employed is not always clear. The first of these is odd as these dissertations should be refereed by thesis examination. He also does not say what level of dissertation (B.S., M.Sc. etc.) they are. His other criticisms are pertinent. Grouping these studies shows, informally, that manipulation skills are either the same or worse if computer algebra is used. Attitudes to mathematics, conceptual understanding, and problem solving/mathematical modelling skills are either the same or improved using computer algebra. The use of computer algebra for demonstrations and homework was not found to be effective. Appendix 3M shows that, using Wilcoxon Sign tests (which are distribution-free), only improvement in conceptual understanding is significant (at the 5% level). Manipulation skills even worsen as a result of using a CAS. Some notable points from these dissertations were:

- Keller (1994) obtained a weak positive result in the study of students' development of symbol sense, but it was not statistically significant. An experimental group appeared to have gained a more widely applicable skill.
- Crocker (1992) noted improved problem-solving skills using Mathematica, particularly for higher ability students. This might be expected anyway, and it is difficult to attribute success to the CAS used.
- Park (1993) found that both conceptual understanding and attitude towards mathematics improved. However, he noted blind execution of commands without understanding concepts and procedures, and the time consuming aspect of this mode of learning.
- Nowakowski's qualitative study (1992) concluded that, in teachers' opinions, a CAS was not useful for teaching formal proof, but allowed for exploration and discovery.
- Rochowicz (1993) found that even among "progressive" educators, only 39% used a CAS frequently in teaching.

These last two studies illustrate that educators' own preferences can be important in driving experimental CAS projects, and hence that this factor ideally should be accounted for in any quantitative study.

Mayes' own studies (Mayes 94 and Mayes 95) are notable because they are quantitative, they recognise problems highlighted in this discussion, and also because of the statistical analysis employed. Groups were allocated to a control or experimental status at random, which eliminates one possible source of bias. Mayes recognises that it is necessary to establish equivalence of the two groups before any experimentation, and an ANOVA analysis showed that there was no significant difference between the two groups in any category of pre-test. Experimental group instructors had special training, which biases the treatments. This could explain the significant (0.1% level) ANOVA result obtained on a combined final measure. This high level of significance is out of line with other studies. Another doubtful experimental factor was the actual test questions used, particularly for the "inductive reasoning" treatment. If the sample given in Mayes' paper is typical, inductive reasoning exercises concentrate on pattern spotting, for which a CAS is not necessary. Mayes points out that the significant result for the final measure of problem-solving must be attributed to the computer context as a whole. He considers

that studying the effect of computers on teaching and learning is a fruitful area for more research.

A similar statistical methodology was employed by Melin-Conejeros (Melin 92). He showed that there were no significant differences between treatment groups on overall achievement, skills achievement or conceptual achievement. There were, again, problems with different treatments. In particular, experimental and control groups took the same pencil-and-paper exam, which may not have been appropriate for the computer group. The sample sizes are also too small for statistical conclusions to be drawn. Melin-Conejeros' conclusion that a CAS should be integrated into teaching does not follow from his statistical results.

The study by Hurley (Hurley 91) is typical of the type of experiment conducted at the beginning of the 1990s, and also of statistical techniques employed in this type of study. He presented the results in Table 3.1 with no further comment.

	Number of students	Mean	Standard Deviation
Computer lab	25	68. 18	17. 34
Traditional	217	59. 98	21. 36

Table 3.1

Making suitable assumptions about the sampling and the background population, a Variance Ratio (F) test shows that, at the 5% significance level, the two samples can be considered to come from populations with the same variance. This establishes that the experimental and control groups are equivalent before experimentation. Applying a *t*-test, $t = 1.85$ is significant at the 5% level but not at the 1% level. This level of significance is typical.

Rickhuss' study (Rickhuss 93) is similar. He found no significant difference in mean scores of pre-test CA and control groups in equation solving or factorisation. However, a significant improvement (at the 5% level) in scores for factorisation resulted. The conclusion that there was no evidence that the CA group did worse than the control group is inappropriate. A positive result in favour of the CA group should be required

in view of the investment in effort and expense in setting up a computer learning environment.

Palmiter's study (Palmiter 91) is notable because of the large sample sizes (about 40 for experimental and control groups), random assignment of students to an experimental or a control group, and uniform teaching standards. These factors reduce bias, but it is difficult to see how uniform teaching quality can be ensured or quantified. There were also differences in course content, no pre-testing, and the *same* tests were used for both groups. It is hard to tell whether or not this was appropriate, but it does provide a uniform experimental environment. Palmiter uses the relatively uncommon Hotelling T^2 statistic to reach the conclusion that the experimental group's performance is better than the control group's. As presented, the results do not appear to make sense. The Hotelling T^2 statistic is used for multi-variate analysis (the two variates being "Conceptual" and "Computational" in this case), and one value of T^2 should be calculated to cover these two variates. One value of T^2 is given for each variate, so there is doubt about the precise nature of the calculations. 2-sample z-test analyses for each variate (Table 3.2) are highly significant ($p<0.0001$) and substantiate Palmiter's conclusion.

Examination	MACSYMA			Control			
	m	s	n	m	s	n	z
Concepts	89.8	15.9	39	72.0	21.4	39	4.11
Computation	90.0	13.3	38	69.6	24.2	39	4.51

Table 3.2

Palmiter gives examples of examination problems, and these provide a partial explanation for the significant result in the "Computational" exam: they are trivial if computer algebra software is used, but can be very difficult if not. There is some evidence that any benefit from using the CAS is persistent: the experimental group also fared at least as well as the control group in subsequent calculus tests. Interestingly, the experimental group felt that one of the most important things they had learned was "techniques of integration", which were not formally taught. Palmiter notes, correctly, that the significant results obtained must be regarded with some scepticism because they cannot be attributed wholly to use of the computer algebra system..

Hunter (Hunter 95) provides a slightly different twist with a novel post-testing strategy. Experimental and control groups had pre-and post-tests in algebra and graphics topics, but some experimental groups had two post-tests, one with and one without the CAS that they had been taught with. In both cases, these groups performed better with the CAS than without, but in some cases they performed worse than the corresponding control groups. The researchers indicate that groups were not of equivalent ability and that the novelty value of treatments played an important part. Thus, little can be gained from studying these results other than to indicate ways of improving experimental conditions.

Experiments in teaching mathematics using DERIVE have been made in Austria since 1984. Heugl (Heugl 94) gives an outline of the Austrian DERIVE project (distinct from the TI-92 project, discussed later in this chapter), and Klinger's paper (Klinger 94) gives some quantitative results. Computers were not permitted in post-tests, and no pre-test was given. Both of these factors cast doubt on the result, which was to conclude that DERIVE improved the performance of the experimental group. It is a pity that no standard deviation information is given, so it is not possible to carry out a significance test. However, a test of proportions (Appendix 3K) indicates a statistically significant result for both experimental groups.

A further significant result was obtained in 1995 and 1996 by Keller and Russell (Keller 97). They found that in courses with a conceptual bias and courses with an applied bias, students who had used a TI-92 calculator performed significantly better (at the 0.1% level). The authors attempt to explain their results by suggesting that use of the TI-92 gives students more confidence and that during the course they are able to concentrate much more on conceptual understanding. A simpler explanation is that, for short answer questions at least, some of the questions asked required much effort to solve by hand, but very little using the symbolic capability of the calculator.

Repo (Repo 94) studied reflective abstraction in the context of derivative concepts. Some of her statistical analyses are arithmetically incorrect, but my recalculations show that there is statistical evidence at the 5% significance level that the DERIVE groups scored better in conceptual understanding. There is no such evidence for computational skills. The result on reversibility of the differentiation operator failed a Variance Ratio test, and the result for the corresponding *t*-test quoted is therefore invalid. Furthermore,

an incorrect Chi-squared test in which cells in a contingency table should have been grouped (but were not), invalidates the 'significant' result quoted. Overall, it is hard to draw conclusions from this study because of inaccuracy and dubious experimental conditions (particularly using leading questions in exercises).

The study by Stiff, McCollum and Johnson (Stiff 92) illustrates an attempt to eliminate one source of bias, but unfortunately introduces others. The sample sizes (typically 70) were large, and experimental and control groups were taught by the same instructor. However, the effect of frequent instructor intervention in laboratory situations could have introduced bias. Measuring techniques appear to be subjective because essay questions, instead of actual computations, were set. This study is a clear indication that a full "package" of tools, teaching methods and testing conditions contribute to the overall result.

Smith (Smith 94) obtained a contradictory result. Several problems arise with this study. Pre-tests consisted of a miscellaneous collection of previous test results, for which there is no evidence of uniformity. Assignments were done jointly, and it appears (but is not clear) that tests were taken individually. An ANCOVA analysis indicated no statistically significant difference between achievement in the experimental and control groups. An ANOVA test showed a statistically significant difference between the groups in student attitude to the mode of instruction. This is consistent with the finding of Schoenfeld (Schoenfeld 88), which was that the use of multiple representations of the same information helps students to understand concepts. There are also indications of delay caused by unfamiliarity with the software, which caused some work to be less effective than it might otherwise have been.

3.6.4 The Austrian *New Technologies in Mathematics Education* (TI-92) Project

Austria has embraced the culture where a CAS environment is becoming the norm, and there is limited evidence that, having assumed the efficacy of the computer, the computer drives learning modes. Nocker (Nocker 98) gives an account of extensive experiments for finding profitable ways to use the TI-92 symbolic calculator. The *New Technologies in Mathematics Education* project started in 1997, and follows similar trials with DERIVE (Austria bought a national licence for DERIVE in schools in 1991). Quantitative results are likely to be published in 1999, but are unlikely to provide much statistical information (see comments on Klinger 94). Nocker reports positive, but unquantified results, including increased volume of computation, increased focus on applications, increased focus on explanation and interpretation, and use of programs created by students, teachers and third parties (as discussed in Heugl 98). This last factor partly determines a curriculum, since what is learned depends on the capabilities of the hardware and what can be programmed. Nocker notes predictable disadvantages: an increased and more demanding workload for teachers and students, and organisational problems. The emphasis is also on contextual mathematics rather than modelling, so the explicit modelling problems of Chapter 1 are avoided. Sharke, who is a contributor to the Austrian project, concludes: "...teachers who use this calculator in their classes will in general get better results", and "Students who have been taught with this calculator and who use this calculator on their own will get better grades and will understand mathematics much better" (Sharke 98). Unfortunately, he does not justify these conclusions, but merely indicates positive feedback.

Despite a lack of quantitative evidence so far, the Austrian TI-92 project remains the only extensive trial of the TI-92 which is capable of providing some quantitative results. There were several reports of smaller TI-92 projects in the USA at the 1998 Gettysburg DERIVE and TI-92 Conference (e.g. Gonzalez 98, Pröpper 98). All reported favourably, but quantitative results were lacking. Such comments must be seen in the light of the way in which using a computer or symbolic calculator changes operational conditions. I discuss this factor later in this chapter.

A recent interesting development in the opposite direction is the decision by the Bavarian Ministry for Education to ban electronic calculators up to mainstream 'Year 8' in Bavarian upper schools (GSO 98). In England and Wales a similar decision was made at roughly the same time to ban calculators below 'Year 4'. These decisions do not seem to be backed by evidence.

3.7 The influence of computer algebra on the measurement of teaching and learning

In this section I examine how computer algebra can influence mathematics teaching. More fundamentally for this thesis, I consider how using computer algebra can affect the result of measurements of achievement.

3.7.1 Early views on how computer algebra can influence mathematics teaching

Evidence from studies in the previous section shows that teaching and learning before and after introduction of a CAS are necessarily unalike. Computer algebra affects teaching and learning and the evaluative measurement process. This devalues the result of a comparison between the two cases. It is interesting to refer back to earlier views on how CA could affect the mathematics curriculum. Stoutemyer (Stoutemyer 84) and Char (Char 84) express similar views, listed below. Stoutemyer is slightly more radical, and Char's comments are backed up by early experience of using Maple with students.

1. There is a reduced 'need' to teach methods: the computer does it.
2. There is a reduced 'need' to teach background topics (e.g. trigonometry for calculus).
3. There is no need to devise problems to fit (but see some counter-examples in Mitic 92).
4. The curriculum can be extended in depth and breadth (e.g. in Brown 91).
5. Different approaches can be used: exploration, generalisation, conjecture (rarely proof).
6. New subject areas can be studied (e.g. applications reported in IMS 95 and IMS 97).
7. Understanding can be stressed.

These points were later implemented in schemes such as (Brown 91), (Ramsden 95), and in texts such as (Glynn 90) and (Leinbach 91). I also illustrate many of them in (Mitic 98), but also give warnings in (Mitic 94B). In these circumstances there is no point in teaching and assessing the same course in the same way.

3.7.2 Implications for statistical trials

A summary of quantitative statistical analyses reviewed in this section reveals that methodology, the actual computations done and the soundness of conclusions cannot be taken for granted. In a typical unsophisticated analysis, the mean score m_C for a control group is compared with mean score m_E for an experimental group, with no further analysis. The claim is then that if $m_E > m_C$, the CAS is beneficial. This is not so, as statistical significance cannot be assessed. In cases where the data collected makes a formal significance test possible, statistically significant results often occur in conjunction with non-significant results. The results are then inconclusive. Overall, trials provide weak evidence that a CAS is neutral or mildly beneficial. Some results are highly statistically significant and there is some evidence that software can worsen certain aspects of learning.

A number of quantifiable factors which can influence the outcome of trials can be proposed. These amount to no more than recognising that certain assumptions are necessary before any given statistical test may be applied. It is easy to ignore these assumptions since they do not directly affect numerical computations. Significant quantifiable factors are:

- sample sizes - the larger the better;
- pre-test results to establish that control and experimental groups can be considered to come from the same background population;
- the distribution of the background population, which is particularly important when applying a *t*-test, for which a Normal background population is needed.

Research papers also indicate that other factors can influence the outcome of trials. These are hard, and perhaps impossible to quantify. Among these are:

- teacher enthusiasm - possibly enthusiasts tend to conduct trials, and their enthusiasm affects the attitude of student and teacher participants;
- the perceived attitude of participants to computers and mathematical software;
- the perception of what can be achieved, in terms of the amount of material covered, its depth and complexity, and the time in which this can be done, all referred to a 'paper-and-pen' benchmark;
- the ease of use of the tool;
- pedagogic and didactic factors, possibly manifest in geographical and cultural terms, which imply that a CAS is more suitable for particular types of learning. Calmet (Calmet 84) also makes the point that computer algebra packages in use in the early eighties were not particularly relevant for the French mathematics curriculum.

Particular note must be made of factors which are directly concerned with the conduct of trials, such as:

- inappropriate use of a CAS (e.g. to do tasks which could be done better by other packages or where programming has replaced a direct mathematical activity);
- non-equivalence of experimental and control groups (e.g. different tests, study times, instructor training).

These factors contrive to make a fair trial very difficult. CAS techniques and 'paper-and-pen' techniques are opposed in terms of what can be achieved immediately by a relatively inexperienced user, and what can be demonstrated by an experienced user. For example, evaluating the expressions $\int\frac{1}{1+x^2}dx$, $\int\frac{1}{1-x^2}dx$, $\int\frac{1}{\sqrt{1+x^2}}dx$ and $\int\frac{1}{\sqrt{1-x^2}}dx$ is equivalent in effort using a CAS, but is different in each case when done by hand. This causes an inequality in the time spent teaching, what is taught, what can be meaningfully examined and how much time is spent on these processes. Hence, a control and an experimental group are so much opposed that any comparison between them is necessarily flawed. The nature of a CAS (or a computer package in general) is that its appropriate use can affect the outcome of the experiment and that, consequently, statistical comparisons are not as meaningful as they appear at first.

3.7.3 Other rationales for using computer algebra packages

I establish in this Chapter that there are severe problems in using comparative trials to measure the effectiveness of teaching and learning with and without a CAS. There are then two alternatives. The first is to recognise that using a CAS is a different mode of study that does not need justification. The second is to find alternative justifications. These are essentially the points in Section 3.7.1: the original arguments in favour of using computer algebra systems have not changed since the mid 1980s. The following recent references illustrate this.

Quigley (Quigley 97) argues that using a CAS shifts emphasis from routine manipulative skills to making strategic and tactical decisions in modelling: using the computer complements and supplements 'human' modelling skills. Similarly, Heubach (Heubach 98) uses a CAS to bypass traditional calculus and differential equation courses for students who do not have a sound understanding of relevant techniques. Geometric visualisations of functions and vectors replace these topics. Visualisation is also a key component in Kawski's approach to vector calculus (Kawski 97). He uses Mathematica to develop practical skills in using vector calculus (Stokes theorem, *div*, *curl*, etc.) before detailed theory is developed. He does this by viewing plots of vector fields. His course is designed to make the purpose of vector calculus constructs clear to non-specialists. A comparison with 'traditional' methods is not even possible because the computer algebra methods achieve results that cannot be achieved using 'traditional' methods. Ohtsuka (Ohtsuka 98) emphasises the role of algorithms with formal CAS programming. This bold move amounts to replacing mathematical skills by programming skills. All these cases extend the scope of mathematics and modelling (and require course restucturing) rather than act as direct replacements for non-computerised activities.

3.8 Summary

This chapter contains a comprehensive review of studies of using computers, and computer algebra in particular, in teaching and learning. From these I have established that the statistical analyses of the results of comparative field trials are often unreliable. Either the statistics collected are insufficient to substantiate stated conclusions, or differences between control and experimental groups influence the outcome. In either case, the effect of using a CA package cannot be measured in isolation: there are always other factors which contribute to the measured result. The most significant of these factors is the subject matter taught.

This conclusion has a profound effect on validating the software developed as part of this thesis. I discuss an alternative method of validation which does not involve field trials in Chapter 9.

Chapter 4

A Summary of Research problems arising from Literature Reviews

4.0 Abstract

The research in this thesis draws upon four main strands: mathematical modelling, computer algebra, computers in teaching and learning, and object-oriented methodology and programming. The main problem addressed here arises from mathematical modelling, and the other strands have a bearing upon this. This thesis addresses two major problems arising from the literature review of mathematical modelling: there is no formal method for linking features in a model, and feature elucidation is not addressed systematically. I provide an object-oriented methodology for casting the problem domain into object-oriented terms, and a new modelling methodology with associated computer algebra software to associate and elucidate features. A further significant problem is that little research has been done on constructing models which have no axiomatic basis (either in terms of relationship production or validation). The modelling methodology developed in this thesis outlines a solution for this. The major problem which arises from the literature review of computers in teaching and learning is that no effective way has yet been found to separate distinct elements (teaching methodology, software, instructor enthusiasm, student motivation and subject matter) in a learning context. This has a significant bearing on validation of the software developed in this thesis. Other research problems not tackled here are to find effective ways of using software, whilst not diminishing algebraic fluency, and to find other effective teaching strategies for modelling. Several problems arise from the review of computer algebra in mathematical modelling. The main one is to find a way to integrate a computer algebra engine into software, and this is done here. This problem is closely associated with that of having to program a computer algebra system and this thesis tackles this problem implicitly by providing templates and front-end software.

4.1 Problems arising from the review of literature on mathematical modelling

Using literature on mathematical modelling methodologies and techniques, and the ICTMA conferences in particular (the latest published proceedings are ICTMA3 89, ICTMA4 91, ICTMA5 93, ICTMA6 95 and ICTMA7 97), I have identified three main gaps in knowledge. I summarise these in the following paragraphs.

1. *How to relate features in a model.* Little research has been done on this specific aspect of modelling, and it is central to this thesis. The Generic modelling cycle assumes success of the step *Formulate model: State relations* in the Penrose 7-point modelling cycle (Penrose 78). Papers (e.g. MST204 89, Herring 97, Berry 95) indicate that this modelling cycle forms the basis of current practice, particularly in the UK. A detailed error analysis of modelling projects in Chapter 9 provides evidence that a significant proportion of students have difficulty with this stage: 23% of the total errors identified represent inability to formulate any model, or a material error in the formulation. This figure represents a failure at a crucial stage of model formulation, yet the problem has attracted little attention.

2. *How to identify features in a model.* The problem of relating variables is closely related to the problem of formally identifying features in a model. Authors are consistent in categorising modelling scenarios and in cataloguing features in each category (e.g. Berry 95, Edwards 89 or Giordano 97). 'Standard' models and features result, which makes it difficult to elucidate material features in different contexts. Standarised models cannot even guarantee that appropriate features are incorporated in a model, or that they are related correctly. There are few formal discussions on this topic. It seems hard to stray from specific examples, and evidence from Beare and Orman (Beare 96 and Orman 95) shows that knowledge of the problem domain is an important part of modelling. The problem of presenting relevant information for a given problem domain therefore still remains.

3. *How to find effective ways of using computers and software in modelling.* Software support is mostly for teaching methods of applied mathematics. Software that supports a modelling methodology has yet to be developed beyond this thesis.

I have also identified three further gaps in knowledge, but do not address them in this thesis.

1. No method has yet been identified for quantifying or optimising the number of elements included in a model, nor of assessing the complexity of interactions between them. Modelling methodologies tend to stress the modelling process instead of the content of the model that results. At the validation stage there is therefore no way of assessing whether or not a model of differing complexity would be better.

2. Little research has been done on constructing models which have no axiomatic basis, either in terms of relationship production or validation. Ways to formulate adequate domain heuristics that can identify important features and relations between them need to be found.

3. In the debate between 'scenario' and 'generic' modelling, little advance towards unification or agreement has emerged. The discussions of Blum (in Blum 91A) and Edwards (in Edwards 89) respectively typify the two views. Optimising existing methodologies continues to be a subject for debate and has not been resolved.

4.2 Problems arising from the review of literature on computers in teaching and learning

My review of the literature on using computers (and computer algebra in particular) in teaching mathematics concentrates on the many trials that have been carried out during the past ten years to assess the benefits of using computer algebra systems. One principal problem arises from this review, and I summarise it in the following paragraph. I address this problem in Chapter 9 by finding an alternative way of validating the ideas and techniques in this thesis.

There is weak evidence from comparative trials of attainment in mathematics with and without a CAS that using a CAS in mathematics education is beneficial, except in the case of fluency in routine algebra. A significant factor is that the results of using a CAS depend on a combination of many factors that are distinct from the CAS itself. Only Mayes (Mayes 95) and Palmiter (Palmiter 91) recognise this. Using a CAS changes the subject matter taught and the way in which it is taught, and this forces non-equivalence of experimental and control groups. Comparisons between the two (statistical or otherwise) are therefore severely weakened. Other contributory factors to this effect include attitudes to using a computer, teacher and student enthusiasm for computer algebra, ease of use of the software and non-equivalence of the treatment of experimental and control groups. No suitable way has yet been devised for isolating the effect of a CAS alone. This problem is addressed in this thesis by finding an alternative method of assessment.

I have also identified two other significant open problems which I do not address in this thesis.

1. Finding effective ways to use mathematical software, without diminishing algebraic fluency remains to be done. This is a particular problem for modelling, which needs to combine elementary algebraic activities with a modelling strategy.

2. There is no research to assess the value of replacing mathematics, wholly or in part, with programming. Programming a CAS is often kept to a minimum, with the applied assumption that mathematical activities are more important. Research in this area is likely to involve a much longer term project.

4.3 Problems arising from the review of computer algebra in mathematical modelling

The following two problems arise from using computer algebra software in a context other than simple mathematical manipulation. Both are addressed in software for this thesis.

1. Few examples exist where computer algebra engines have been integrated into a dedicated mathematical environment. Some MathWise modules include a Maple kernel, but computation in most modules is purely numeric. The open problem is to find and develop ways of integrating computer algebra engines into software such that the resulting environment is sufficiently flexible.

2. General operations provided by a computer algebra package cause particular problems in doing routine manipulations in algebra and calculus. Ways need to be found to ease algebraic manipulation and pattern matching.

Three more minor problems are also apparent, but I do not tackle them in this thesis.

1. There is virtually no research on comparing the effects of using different software packages in the same context. This is important because there is a tendency to use a CAS for almost any mathematical task without regard to other software. The result may be more effective if either a different CAS or a different category of software is used.

2. Algebraic rules are not always propagated during the course of routine manipulations. This makes using a CAS difficult because its capabilities do not correspond to established mathematical behaviour. Amending existing software would involve considerable redesign of the algebra engine.

3. Considerable effort has already gone into finding ways to simplify expressions, and to cast output expressions into suitable and appropriate forms. This is an ongoing task for CAS developers.

Chapter 5

The Case for Object-Oriented Modelling

5.0 Abstract

In this chapter the basis for an object-oriented solution to the problem of identifying and relating features in a model is established. An initial description of object-oriented concepts shows what new concepts are required and which of them are important for mathematical modelling. A detailed analysis of why an object-oriented solution is feasible and desirable follows and is based on two main considerations. First, treating the elements in the problem domain as objects forces an analysis of their features, and how they behave and interact. Second, encapsulation of this knowledge allows manipulation of elements in the problem domain such that the problem can be formulated in a precise and logical way. I stress the important aspect of object-oriented analysis in a modelling context: the object model. Evidence shows that object-oriented solutions have been successful in mathematical contexts in the past. This thesis describes the major features of the object-oriented analysis and design methodologies which have been in use during the past 10 years. They may work well for large business systems. However, they do not focus successfully on constructing objects for mathematical modelling. They contain a significant overhead in terms of software maintenance and management. Their most useful aspects for mathematical modelling are noun-verb analysis and the use of heuristics of the problem domain as a starting point for object design. These ideas are advanced in the next chapter.

5.1 Object-Oriented concepts

Many attempts have been made to define the fundamental terms *object* and *class*, but all of them are somewhat subjective. A useful guide is given in (Shlaer 88 and Coad 90). Objects are: *abstractions of a set of real-word things such that all things in the set (the instances) have the same characteristics and conform to this same rules*. Objects have characteristics (*attributes*) and their behaviour is described by procedures (*methods*). The sort of objects that Shlaer and Mellor (and others) considered were not mathematical, and one problem that I address in this thesis is how to cast a mathematical context into O-O terms. The thrust of the O-O paradigm is to treat a class as an abstract unit (*encapsulation*), as in (Parnas 79, Abelson 85 and Seidewitz 86). Classes can then be organised into an object hierarchy using *inheritance*. In this way, classes can be based on previously-defined classes without the need for repetitive programming. Appendix 5A provides a brief guide to these and other O-O concepts.

Switching from a procedural paradigm to an object-oriented paradigm is often difficult. Beck (Beck 89) considers that the process of object construction relies on a heuristic knowledge of the problem domain. Many O-O design methodologies suggested collaborative work to construct objects. Collaboration is also a feature of current mathematical modelling practice.

5.2 The case for an Object-Oriented Solution

In this section I consider how an O-O approach to mathematical modelling can be beneficial. There are two main strands. Evidence from problem-solving indicates that problem statements and solution strategies are limited in number. These characteristics are easily cast into O-O terms.

5.2.1 The need for a new methodology

Examining papers from the last five ICTMA conferences, and other sources (e.g. Trielibs 79), reveals little insight into tackling the problem of how to identify relevant features in a model and how to formulate relations between them. Three points provide the motivation for introducing an O-O approach. Each is discussed in greater detail later in this chapter.

- It is easy to cast modelling contexts, and particularly Newtonian mechanics, in object terms.
- O-O techniques concentrate on how objects interact, which is fundamental to modelling.
- O-O analysis provides explicit ways to describe the behaviour of a physical object.

An object model for a mathematical context can be supplemented by an explicit O-O modelling methodology, which is integral to the process of relationship formulation. This is demonstrated in Chapter 6.

5.2.2 Evidence from Problem-Solving

Within the context of Newtonian mechanics, a number of relevant established texts (Berry 95, Capildeo 68, Easthope 64, Edwards 89, Dyke 92, Stephenson 61, Milne 48, Taylor 86, Turner 73, UCLES 95, Quadling 57) show consistency in the ideas behind problem statements. These references span forty years, but the concepts in them are essentially the same and are independent of the level of study.

- The problem domains consists of elements such as particle, solid cylinder, spring, force, gravity etc.
- The goal of each problem is essentially the same: to derive and solve an equation of motion.
- The types of problem considered in these references differ only in the sophistication of the mathematical techniques employed.
- The techniques for solving problems in mechanics are essentially the same, and involve the following steps:
    - drawing a diagram;
    - inserting relevant forces on the diagram;
    - isolating massive objects;
    - inserting accelerations for each massive object;
    - obtaining an equation of motion by applying Newton's 2nd Law of Motion for each massive object;
    - eliminating terms from the resulting equations;
    - solving.

The following examples illustrate this uniformity for a projectile problem, which is typical of simple problems in mechanics. They demonstrate that the variable factor over the past 40 years is the approach to finding a solution. Problem statements and solution methods are paraphrased.

Problem 1: (Quadling 57)

A boy throws a stone horizontally from the top of a cliff with a speed of 40 ft/sec. Discuss its subsequent motion.

Outline solution (supplied):

1. statement that horizontal motion is constant;
2. statement that vertical motion is described by an equation of the form "$s = ut+ft^2/2$";
3. set up coordinates;
4. diagram drawn;
5. statement of equations of motion for $x(t)$, $y(t)$;
6. solve: eliminate t to obtain $y(x)$.

Problem 2: (Turner 73)

A stone is thrown out to sea from the top of a cliff. If its initial velocity is ***u*** *and thereafter it has an acceleration* ***g*** *vertically downwards, find its position after time t.*

Outline solution (supplied):

1. diagram drawn;
2. set up coordinates in terms of orthogonal unit vectors;
3. statement of equation of motion for r(t), expressed vectorially;
4. solve: integrate twice;
5. a numerical example.

Problem 3: (Dyke 92)

A car drives off a horizontal pier with a speed of 45 kmh^{-1} and crashes into the sea 2s after leaving the pier. Find (a) the height of the pier and (b) the horizontal distance the car travels before entering the sea.

Outline solution (not supplied but inferred from the previous example):

1. diagram drawn;
2. set up coordinates x and y;
3. use "$v^2 = u^2 + 2as$" and/or "$s = ut + ft^2/2$" substituting initial conditions;
4. solve: for t and other quantities required.

Problem 4: (Bostock 96)

This is an up-to-date version which comes near to my 'ideal' solution method.

The top of a tower is 10m above horizontal ground. A boy fires a stone horizontally with a velocity of 12 ms^{-1}. Find how far from the foot of the tower the stone hits the ground.

Outline solution (supplied with preliminary theory):

1. diagram drawn;
2. set up coordinates x and y;
3. write down equations for $\dot{x}$ and $\dot{y}$;
4. integrate to obtain $x(t)$ and $y(t)$;
5. solve: for t using $y = 10$.

I prefer a slightly more fundamental approach than Bostock's, by subdividing Step 3:

3.1 write Newton's 2nd Law for horizontal and vertical motion;

3.2 integrate to obtain $\dot{x}$ and $\dot{y}$;

These treatments look similar, but even small differences are notable because they express a general approach to modelling. There are varying degrees of generality, different orders of operations in the solution process, different degrees to which assumptions are discussed (including none at all) and different dependencies on Newton's 2nd Law. My 'fundamental approach' (Steps 3.1 and 3.2, above) is the only one which contains an explicit dependence on an axiom, which is a rare approach in a projectile problem. It is, however, fundamental to automating problem-solving of this sort because it expresses generality.

Problems 1 to 4 of this section indicate that the following should be considered:

- what physical objects are in the problem domain;
- how analysis proceeds for each such object;
- how objects in the problem domain interact with each other;
- how to include sufficient generality to be able to solve a reasonable variety of problems with the same objects;
- a unified approach for solving similar problems.

5.2.3 O-O concepts in Heuristic Modelling Strategies

Many activities in problem-solving apply to almost all situations in Newtonian mechanics. Analysis proceeds for each massive object, by applying Newton's 2nd Law. In some cases (e.g. jointed rods) parts of the system are logically separated. This implies the existence of fundamental physical objects. The model must mirror the way in which they interact. The following points provide the motivation for a formal object-oriented treatment for Newtonian particle mechanics.

1. Newton's Second Law is applicable to massive objects only. This provides an aim for a problem solving methodology: to derive an equation of motion. Although there are clear exceptions to this statement (momentum-impulse problems, for example), such exceptions merely require a different object treatment.
2. Certain elements or combinations of elements in the problem domain behave consistently in the same way. For example, massive objects in contact always a give rise to contact forces through Newton's 3rd Law. This is a generic treatment for interacting objects which can be implemented in terms of message passing and class methods.
3. Objects are routinely created and destroyed in non-O-O analysis. A good example is a stretched spring. When it is connected to a massive object and stretched, a tension is created, and the spring can cease to exist as far as further analysis is concerned.
4. Sometimes unusual cases arise in modelling Newtonian particle mechanics. For example, a spring may not obey a linear force law. In this case the analysis of the problem is exactly the same as for a perfect spring but with a different force law. This can be approached by defining a descendent class with a relevant overloaded method.
5. Certain well-defined operations are often performed on elements in the problem domain. These include geometrical transformations, resolutions along co-ordinate axes and interactions with the other elements in the problem domain. These can be interpreted as object methods in O-O terms.
6. Certain properties of elements in the problem domain are well-defined and are always associated with those elements. These are often physical properties of the elements, such as mass, length or velocity. These can be interpreted as object attributes.

7. Examining the Newtonian particle mechanics problem domain indicates that it has few distinct elements. They are often introduced implicitly or, in more recent examples, using the phrase "is modelled by" to indicate a modelling assumption. The same is true for other problem domains. For this reason, it becomes feasible to consider a non-extensive object model, possibly with no need for inheritance.
8. Drawing a diagram is often regarded as an essential part of the modelling process and is often taught as a formal stage in problem-solving. The diagram can indicate what the objects in the problem domain are. Diagrams are used for reasons such as the following, all of which can be expressed in terms of class methods:
    - To clarify which elements are in the problem domain.
    - To determine the geometry of a system, especially if trigonometry is involved.
    - To represent different simultaneous views (e.g. forces and accelerations).
    - To initiate formal analysis (for example, resolution of forces).
    - Reinforcing concepts (for example, the temperature gradient in a heat conduction problem can be represented by a sloping line for which the higher temperature is nearer the top of the page than the lower temperature).

5.3 The potential benefits of an O-O treatment

I now consider the ways in which an object-oriented strategy can be beneficial. Each important O-O concept merits consideration, and most fundamental ones are the class hierarchy, and the attributes and methods of the classes in it.

A vital aspect of mathematical modelling is to know what the characteristics of elements of the problem domain are. A spring, for example, has stiffness, unstretched length and co-ordinates of its ends. A further vital aspect is to know how elements of the problem domain interact with each other. A spring can interact with a particle to produce a force (its tension), which then interacts with the particle. Formal properties of elements in the problem domain are easily expressible in O-O terms. Furthermore, having to express a problem in precise O-O terms focuses attention on what the formal properties of elements in the problem domain are.

It is therefore necessary to supply:

1. all necessary attributes (so there is no missing information about characteristics);
2. all necessary methods (to define interactions with other objects);
3. a description of the geometry of the system (supplied using attributes);
4. details of "invisible" objects, such as fixed points and gravity.

Object attributes can be made to correspond to the physical properties of an element in the problem domain, or to properties closely related to them. Prime candidates are masses, lengths and velocities. Attributes can also be used to record the state of a system through Boolean variables. An example is the interaction of a spring and a particle in which the spring is, in effect, inactive after it interacts with the particle. At the point of interaction an attribute of the spring object can be set to "inactive". This has the advantage that a full history of a system can be recorded, and no information is lost.

Descriptions of interactions between objects correspond to Object methods. They provide a formal description of not only of how these interactions take place, but also of conditions under which they may take place. King (King 89) expresses this in terms of 'behavioural abstractions'. His key point is that the O-O approach stresses manipulation of data through message-passing, not representation of data. This is important because it provides a way of forming relationships between features in a model. It also provides a means for determining whether or not a potential interaction is allowable or not, thus preventing gross modelling errors. Object methods also provide the means to access the object's private attributes. This is an added complication as far as modelling is concerned.

Object creation (e.g. using a Constructor method) is useful as a formal way of declaring that an object is in the problem domain. For example, declaring `new[Particle, m, {a,b,c}]` formally places a new object, a particle of mass m, at co-ordinates $\{a,b,c\}$. This is equivalent to drawing the object on a diagram and labelling it. It also ensures that objects cannot be used unless they are properly defined. This serves to supply features in a model. Calling a destructor is a formal way of indicating that an object need take no further part in the modelling process. It is not strictly necessary but can be useful to show that particular processes have taken place. For example, once two forces have been combined, they may be replaced by the resultant force, and then take no further part in the computation.

New mathematical models are often based on old ones. The concept of reusability is therefore not new to mathematics. One way to do this is formalised in O-O terms by a class library. Once a class been defined, objects can be constructed with very little extra effort. Capturing attributes and methods is potentially long and generally needs experience, but is a one-off activity. Once a class library exists it can be used by other users. The second function is to define descendent objects, which is useful if a small amendment to an existing object is needed.

The concept of encapsulation (data hiding) is not particularly beneficial for mathematical modelling except to provide a class hierarchy for future use, such that the user need not be aware of implementation details. Problems arising from a need to do further coding may be eased by providing a code-generating tool to define methods and/or attributes.

Polymorphism corresponds to an intuitive idea in modelling: the behaviour of different types of objects can be described in the same way. For example, Newton's 2nd Law of Motion may be applied to an extended massive object as well as a particle. The user can apply Newton's 2nd Law to either, and is not concerned with any programming problem caused by extended objects and particles being different types of object. An O-O treatment can fulfil this need through polymorphism, which need only be a concern for the designer of a class hierarchy.

5.4 Can an O-O approach be applied to Mathematical Modelling?

In this section I ask the question: "Is it possible to determine, in advance, whether or not an object-oriented environment can be constructed for any given domain". With hindsight, object-oriented techniques have been used successfully in a variety of systems, including databases, control systems, management systems and also in computer algebra systems. It is often taken on trust that it is possible to cast any domain into O-O terms, but it is preferable to have some prior indication that it is feasible to do so. There is no direct evidence that it is possible to construct an object-oriented environment for mathematical modelling. In this section I demonstrate that this is feasible by considering a one major task and three minor tasks in constructing such an environment.

The major task is to identify objects. Many established methodologies already exist for this, and their initial task is to identify candidate classes. They do this by considering processes performed by the system (Shlaer 92), physical elements in the system (Wirfs-Brock 90) and logical elements in the system (Rumbaugh 91). These methodologies tend to be applied to large-scale software systems. In Section 5.8 of this chapter I show that they are not always appropriate for identifying objects in a mathematical modelling context. Chapter 6 presents a more appropriate methodology.

To find what a *class* is in the context of mathematical modelling, it helps to consider a modelling diagram. It is unclear precisely what constitutes a distinct object in a diagram, so we can consider that the diagram has been drawn by placing symbols, from a discrete and finite set, on the page. Each placement of a symbol on the page constitutes the creation of an object. From this, potential classes can be identified. This concept exactly mirrors the way in which symbols are placed on screen by a graphics package.

Booch (Booch 94) provides some guidance as to what an object is. Intuitively an object is:

- *A tangible and/or a visible thing*
- *Something that may be apprehended intellectually* [1]
- *Something toward which thought or action is directed.*

These ideas can be condensed into the definition of Smith and Tockey (Smith 88): "An object represents an individual, an identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain". Cox (Cox 86) also thought of an object as anything with a crisply defined boundary, which coincides to some extent with symbols placed on screen by a graphics package. All of these definitions are condensed into Booch's own: "An object has state, behaviour and identity; the structure and behaviour of similar objects are defined in their common class." Khoshafian (Khoshafian 86) defines *identity* as "that property of an object which distinguishes it from all other objects". As part of an object definition this is unsatisfactory because it is self-referential. It does, however, provide the idea that distinct objects should be clearly distinguishable from each other.

---

[1] '*comprehended*' ?

Wirfs-Brock provides a more useful idea (Wirfs-Brock 90). She considers the *responsibilities* of an object. This includes the "knowledge" that the object maintains and the actions that it can perform. The idea is to express a sense of the purpose for the object in the system, which is key to relationship formation among features.

A number of other indicators show that an O-O approach is likely to succeed, because the suggested items to look for are present, in some form, in mathematical modelling domains. Shlaer and Mellor (Shlaer 88) suggest tangible things, roles, events and interactions. Only "tangible things" seems directly relevant, and there are many of them. In the domain of mechanics there are 'standard' objects such as particles, cylinders, springs, and rough surfaces. In the domain of heat transfer there are walls, insulation, and also more abstract entities such as temperatures and thermal conductivities. Using the objects in a modelling environment shows what events and interactions might be relevant. The concept of 'roles' is relevant if a co-ordinate system is treated as something that performs a role rather than acts as a tangible thing. Ross (Ross 87) adds concepts (principles, ideas) to the list. This is important because it signals the inclusion of mathematical laws, axioms and heuristics. Finally, Coad and Yourdon (Coad 90) suggest structure ("is-a" and "part-of" relationships) and external systems. Structure relations indicate how a class hierarchy might be built. External systems might include gravity.

Despite many attempts to define an object rigorously, none is really satisfactory and it is not hard to provide a mathematical modelling example that does not fit the definition well. *Gravitational Field* does not have a crisply defined boundary, is tangible but not visible, can be comprehended intellectually, and action is not really directed at it. Smith's definition (Smith 88) seems to get closest, but only because it is very general.

Five further considerations indicate that an object model for mathematical modelling is feasible. After stating them I illustrate them by considering a simple modelling problem (Problem 5). This problem shows that, although queries to elucidate details of classes may be clear, interpreting the information from those queries is not problem free. The five considerations are:

1. The nouns in a problem statement indicate classes or class attributes (suggested in, for example, Abbott 83, Booch 94 and Rumbaugh 91).
2. The verbs in a problem statement indicate instance methods.

3. If elements are introduced into the system one by one, information can be gained about changes to the system and how new elements effect the elements that are already there. This information then forms the basis of definitions for object methods.
4. Whether or not an object model works is subjective, but it should include an element of progress towards a mathematical goal (e.g. an equation of motion). An initial object model can encapsulate these goals by defining suitable classes. This serves to order the modelling process.
5. A solution strategy can be encapsulated as a class method.

The following problem (Problem 5), from (Bostock 96), shows how points 1 to 5, above, may be applied.

Problem 5

An aircraft is looping the loop on a path which is a vertical circle of radius 400m. Find the minimum speed at the top of the loop for which the pilot would remain in contact with the seat without wearing a seat belt.

Point 1: The nouns *aircraft, pilot* and *seat* are potential classes: they could be drawn on a diagram (Figure 5.1). Other nouns include *loop, circle, speed* and *seat belt* They would not normally be drawn on a diagram, and this indicates that they are not candidate classes. *Speed* is a property of the aircraft, so this an attribute of the *aircraft* object. Domain knowledge suggests that *seat* and *aircraft* are identical classes, since they are rigidly joined.

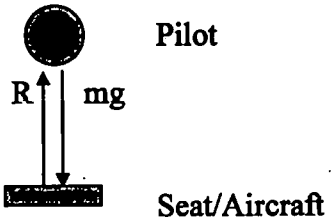


Figure 5.1

Point 2: The verbs in this problem statement are not helpful. The context indicates that gravity is present, and hence that a *gravity* class is needed. The verb phrase *remain in contact* points to an interaction of the *seat* with the *pilot*. This could

reduce to a method of the *pilot* class. The same verb phrase also provides a solution strategy by setting a contact force to zero.

Point 3: If the objects *seat* (*aircraft*), *pilot* and *gravity* are drawn in that order, the it is possible to isolate the following consequences. The *seat/pilot* interaction gives rise to a contact force by Newton's 3rd Law. The *pilot/gravity* interaction gives rise to a *weight*, which acts on the *pilot*.

Point 4: The context, and the goal *Find the minimum speed...* indicate that an equation of motion is required. This comes from an interaction between a massive object (*seat* or *pilot*) and the forces on it.

Point 5: Casting the equation of motion as a class encapsulates the goal of finding an equation of motion.

This example shows that even if it is possible to cast a problem domain into O-O terms, the details of doing so may not be straightforward. In particular, an analysis of nouns and verbs in a problem statement does not necessarily produce a unique, or even a reasonable object model. Despite problems of ambiguity and semantics, noun-verb analysis can be automated, as it was at Fujitsu (Saeki 89). Domain knowledge is nearly always necessary to define reasonable candidate classes. Overall, the subject of pre-assessing a problem domain for suitability for O-O treatment is a subject for further research.

5.5 Pointers from Toolkit approaches and existing applications

In this section I review existing applications which provide evidence that an O-O approach is likely to succeed.

In (Dubisch 90), Dubisch describes a Mathematica implementation of an energy 'toolkit' approach to modelling applied mathematics. He aims to divert the user's attention away from programming syntax towards problem-solving strategies. Dubisch talks about problem 'attributes', which are not attributes in an object-oriented sense, but need the same considerations to find them. They are:

1. the problem domain contains a finite number of bodies possessing kinetic and potential energies;
2. a co-ordinate value is part of the description of each body;
3. The problem domain can be described by a finite number of instances (meaning "a collection of information about the system at a given time").

This solution process has stages that are appropriate for an O-O analysis:

1. allocate symbols to objects;
2. define energy terms and express them symbolically;
3. solve an energy balance equation for a named parameter.

This analysis is insufficient in an object-oriented treatment because it contains a confusion of terms and pays insufficient attention to constructing objects. However, identifying objects with their attributes and methods, and characterising the solution process, makes it possible to cast the problem into O-O terms.

A formal object-oriented environment for solving problems in Finite Element Analysis was implemented in 1992 by Viklund and Fritzson (Viklund 92). The software integrated a Mathematica algebra engine with numerical routines in C++ and a higher level language (ObjectMath) for programming. This system outwardly demonstrates that it is possible to implement an object-oriented system for certain classes of mathematical modelling. There are indications in their paper that the algebra engine is not central to the design, since many methods are implemented in C++. Its main use appears to be for rapid prototyping and it appears that there is an intention to dispense with it completely in future implementations.

Another dedicated modelling tool is *Tangram*, which is an O-O based configurable toolset implemented in Prolog (Page 89). It is used for input-output systems such as queuing problems involving Markov Chains. The authors state that modelling typically requires knowledge of the problem domain. The aim of this software is to capture problem domain heuristics and solution strategies in the form of classes by abstracting specific models into general cases. Its basis is a set of classes defined as Prolog modules. Message passing is interpreted as "proving a goal in a module". This architecture has limited flexibility because rules relate to the system as a whole rather than individual objects. Prolog could not form a basis for the software of this thesis

because it has no associated modelling methodology, has limited numerical capability and no capability at all for doing symbolic computation.

The computer algebra system *Views*, (Abdali 86), implemented in Smalltalk-80 by Abdali, Cherry and Soiffer, demonstrates that the O-O paradigm can be applied to a CAS. They make the valid point that 'traditional' CASs are symbol-oriented, meaning that the data involved are symbolic expressions which can be manipulated without regard to any underlying algebraic or mathematical structure. This is a clear disadvantage if there is a need to work within a particular algebraic structure, and the object model in this system is based on a hierarchy of algebraic structures. *Views* does not include a dedicated modelling environment but is more of a tool for symbolic computation in a robust algebraic environment.

Tan and Steeb (Tan 98) describe a more recent O-O based CAS, *SymbolicC++*. It is also a general purpose tool, but is capable of supporting a dedicated modelling environment by defining suitable classes in C++, using the symbolic manipulation classes already provided, and programming a dedicated front-end. As such, it would be very suitable for the purposes of this thesis. However, its symbolic manipulation capabilities are very limited. There are classes for very large numbers, matrices, linked lists, basic algebraic operations, polynomials and algebraic structures. These are used successfully to solve some problems in physics, but major elements of symbolic computation are not implemented. These include integration procedures (despite a discussion on integration techniques), a general purpose equation solver, factorisation procedures, a differential equation solver, and a programming language. To include these would be a monumental task, but the aim is to provide basic building blocks rather than a complete CAS. Enhancing *SymbolicC++* by building dedicated modelling procedures is equivalent to accessing a commercial computer algebra kernel directly. In principle, this would be easier than programming fundamental symbolic computation capabilities.

5.6 Reduced programming overhead and risk of error

In this section I consider how an O-O treatment can be more beneficial than a 'conventional' treatment using a CAS. The discussion concentrates on isolating important elements in the problem domain and analysing their properties.

To do this, I use a modelling problem from (Zimmerman 95). It illustrates how Mathematica CA techniques may be applied to problems in physics, and exhibits a common phenomenon: it is easier to do by hand than by a 'conventional' computer treatment. The O-O approach is useful for removing unnecessary programming syntax, which is not part of the mathematical problem, and in structuring the solution process. A reasonable comparison must be between the two computer treatments because the computer is necessary to automate more complicated problems and problem-solving strategies. The goal of the problem is to derive and solve an equation of motion from the Lagrangian of a dynamic system. The Lagrangian arises from a physical object and non-ignorable forces, and is created as an implicit object in this type of analysis. It therefore makes sense formally to implement these problems in O-O terms.

Problem statement (Figure 5.2)

A bead B slides on a rotating wire, fixed at A and rotating about a vertical axis with uniform angular speed w. The distance AB = $r(t)$ is variable. The goal is to obtain the equation of motion by considering the Lagrangian of the system.

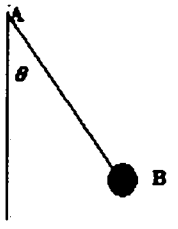


Figure 5.2

If spherical polar coordinates are used, $r(t)$ is the only independent variable.The kinetic energy T, potential energy V and Lagrangian L of the bead are:

$$T = \frac{m}{2}\left(\dot{r}^2 + (rw\sin\theta)^2\right)$$

$$V = -mgr\cos\theta$$

$$L = T - V$$

From these it is easy to derive the equation of motion.

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{r}}\right)-\frac{\partial L}{\partial r}=0\Rightarrow \ddot{r}=rw^2\sin^2\theta+g\cos\theta$$

Defining a simple procedure to produce the Lagrangian automates this computation:

```
Lagrangian[qi_,L_,Qi_:0,t_:t]:=
   D[D[L, D[qi[t],t]], t] - D[L,qi[t]] == Qi
```

However, several problems are apparent in the 'conventional' computer treatment.

- The most efficient way of implementing the differentiations is to express r as a Mathematica pure function (i.e. without explicit reference to t) when converting from Cartesian to polar coordinates. This is tricky to express in a workable form without experience.
- Solving the equation of motion involves several rewrite rules to simplify trigonometric terms, and it is not always clear in which order these should be applied.

The potential advantages of using an O-O treatment are listed below.

- Since an O-O treatment forces the user to concentrate on defining which elements are in the problem domain and what the properties of those elements are, it is a natural basis for analysing problems with distinct physical objects. In this case, the user needs to concentrate on the massive objects in the system, since they are the ones required for a Lagrangian treatment. The forces in this system are either ignorable or not, and this property can also be encapsulated. In general, a non-O-O treatment is in danger of paying insufficient attention to the properties of elements in the problem domain.
- Difficult programming constructs can be encapsulated in class methods. In the Lagrangian example, the differentiations are all done automatically, and there is a mechanism for ensuring that changes of variables are propagated in all such differentiations. This encapsulation makes the code easy to use elsewhere by using change of variable methods for other classes.
- Common transformations (e.g. Cartesian $\rightarrow$ polar) can be encapsulated in class methods. The mechanics of coding coordinate change procedures can be complicated, and it is easier to call a pre-defined method.
- It is less error prone to adopt a well-defined methodology in which objects interact in such a way as to produce and solve an equation of motion. The specific steps in

the Lagrangian example might be to construct the appropriate objects (particle, forces and gravity), call the Lagrangian method of the particle to produce a Lagrangian object, and then call methods of that Lagrangian object to simplify and solve the equation of motion.

5.7 Warnings

During the late 1980s object-oriented analysis and design techniques gained greater prominence, along with increased use of O-O programming languages, but few authors raised specific problems associated with this use. I now consider some of these objections to the use of object-oriented techniques based on the critiques in (Hoydalsvik 93) and (Guthery 89). The former must be put into context, which is that of a large-scale software project involving an administration information system and engineering design. These involve significant OOA and OOD phases which are not applicable to the context of small mathematical modelling systems.

The objections raised by these authors are summarised in the following list, and my replies to them are given alongside in italics. Some of these replies (Note 1 and Note 2) fall into general categories which are discussed at the end of the list. Items 1-12 are raised by Hoydalsvik and Items 13-16 are raised by Guthery.

1. Failure to identify changes in the human aspects of an organisation. *These factors have very little relevance to mathematical modelling since they belong mainly to external systems.*
2. It is hard to determine which parts of the system should and should not be automated. *This is a potential problem but a generally applicable rule adopted in this thesis is that processes should be manual but the consequences of those processes should be automated.*
3. Objects and a class hierarchy are hard to maintain in a continuously evolving system. *This is unlikely to be a problem in a small system which is generally static and determinable in advance.*
4. An object model isn't necessarily good if it is based on the way humans think. *This is unsupported by evidence. The object model need not be visible to the user, so its details do not matter.*
5. The object model might not fit well in all cases. (*Note 1*)

6. Modelling requirements can conflict with object-oriented requirements. (*Note 2*)
7. "Business rules" (heuristics) are hard to capture. (*Note 2*) *This does not apply in a mathematical modelling scenario if it is based on an axiomatic system. It is a potential problem for non-axiomatic systems.*
8. Heuristics in real-time processes can cause implementation problems. *Most mathematical modelling contexts are not based on real-time processes. In principle it is possible to build a real-time interaction into an object model by implementing message-passing which blocks certain events until other events have occurred.* (*Note 1*)
9. Strict encapsulation of processes involving interaction of two different objects is awkward. *This comment is based on a purist interpretation of O-O principles. The situations relevant to this thesis do not require a strict application of object-oriented rules. They depend on broad O-O principles and take a pragmatic approach.*
10. Validation can be hard. (*Note 1*)
11. It is sometimes necessary to "bend reality" to fit the object model. (*Note 1*) *This is true in any mathematical model. This process is almost synonymous with the phrase "model simplification", and is therefore an essential part of modelling.*
12. The object model obtained as a result of OOA depends on whether or not the primary consideration is the dynamic model (concentrating on message passing) or the static model (concentrating on entities). (*Note 1*) *All that is required is a workable object model. Of course, there will probably be scope for improvement.*
13. There is no need to express a problem in object-oriented terms if it can be solved satisfactorily by other methods. *In this analysis I am trying to solve particular modelling problems for which alternative solution strategies have been inadequate (detailed in Chapters 1,3 and 9). Automation is also a primary goal, and O-O techniques do can achieve this in a convenient way.*
14. There is a considerable software overload in designing and maintaining an object hierarchy. (*Note 2*). *A different thought process must be used to construct a class hierarchy and this is likely to be difficult initially. However, the problem is simplified because the main objective is to use objects and not to define new ones.*
15. It is not easy to combine or amend object hierarchies. (*Note 2*)
16. Management of a system which contains too many objects can be extremely difficult. *This is unlikely to be a problem in a mathematical modelling context*

because relatively few objects are likely to exist. There may be, typically, 10 or 20 distinct objects compared with thousands in a large software product.

(*Note 1*)

This is a generic problem, and does not necessarily relate to an object-oriented system.

(*Note 2*)

This is a potential problem in producing an object-oriented system. Care is required in the implementation of message passing in particular, and this will represent an overhead for an O-O methodology.

5.8 Object-Oriented Design Methologies

In this section I examine the major O-O design methodologies, and assess their applicability to object design for mathematical modelling.

5.8.1 Wirfs-Brock

The Wirfs-Brock methodology (Wirfs-Brock 89) stresses the concept of *responsibility*: objects are responsible for actions. This idea is potentially useful in mathematical modelling for deciding what should result from the interaction of two objects. The Wirfs-Brock design process contains steps which are designed to organise classes into natural groupings or in client/server terms . The aim of this process is to define inheritances, collaborations and responsibilities. These factors seem more appropriate for refining systems in which classes have already been determined. Wirfs-Brock addresses how to 'discover' objects in (Wirfs-Brock 90). This tackles some key questions for mathematical modelling, including guidelines for defining classes. They include:

1. Identify physical objects.
2. Identify noun phrases.
3. Identify concepts (one word corresponding to one concept).
4. Use the requirements specification.
5. Examine interfaces to the outside of the system.

Of these, the first two are very important for mathematical modelling because physical entities figure prominently, and noun phrases originate from problem statements. The third is important because it helps to identify non-physical entities such as forces. The

others are more relevant for business modelling. Problems with misleading phraseology, plurals and passive phrases complicate these processes, and ways to resolve them are not clear.

Identifying verb phrases in the 'requirements specification' can serve to define responsibilities and provide instance methods. Other suggested methods are to find where actions occur, and to seek "is-a", "is-like" and "is-part-of" relations. This can cause problems because mathematical modelling problems do not always flow in a piecewise fashion, and because it is not always clear to which classes responsibilities should be assigned, if there is a choice. Responsibilities are fulfilled by finding collaborations, which represent client-server relationships. It can be hard to define client-server relationships in mathematical modelling. It would be better to ask the question "what happens if you combine an instance of Class *A* with an instance of Class *B*?", which can be used to define a modelling sequence.

Thus, the Wirfs-Brock methodology incorporates some ways of searching for classes: consideration of physical objects, noun phrases and using concepts. The idea of 'responsibilities' is important when trying to determine methods for classes but it is harder to find them in isolation, and iterative processes are essential.

5.8.2 Class-Responsibility-Collaboration (CRC) Cards

CRC cards are usually used within the context of other methodologies (particularly Wirfs-Brock) but can be used in isolation. Bellin (Bellin 97) describes a methodology for using them and CRC card development is described in (Cunningham 86) and (Beck 89). They are designed to document collaborative design decisions, and take the form of small cards with details of the class name, superclass, attributes, responsibilities and corresponding collaborators.

The purpose of CRC cards is to organise information in a *known-to-unknown* manner as opposed to a *top-down* or a *bottom-up* way. The designers considered that there was considerable value in physically moving the cards around and therefore resisted computerising them. This is in marked contrast to other methodologies. Bellin recommends brainstroming and team effort to derive classes. Group work is currently popular in mathematical modelling. The need for actual cards is unclear, although they may be more appropriate for smaller projects, including mathematical modelling contexts.

5.8.3 Shlaer-Mellor

The principles of Shlaer and Mellor's data-driven design methodology are given in (Shlaer 88) and (Shlaer 89). (Shlaer 92) later emphasised the data-driven aspects. Their methodology aims to capture domain-specific knowledge analytically, in a way that can be programmed accurately. The three stages in Figure 5.3 are the most important because they deal with elucidation of classes and of the relationships between them. These are key activities for mathematical modelling, but the overall thrust of the methodology is to provide contrasting views of the system. This is more appropriate for large business systems.

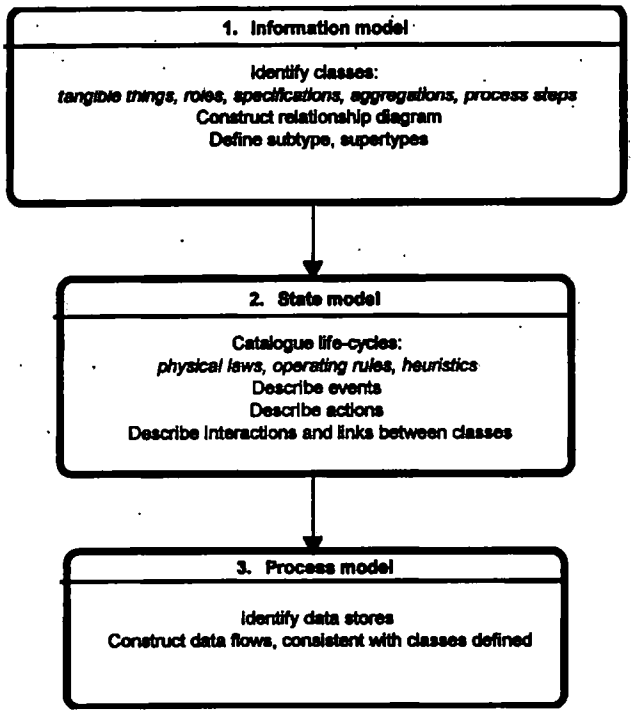


Figure 5.3

Information models stage

Shlaer and Mellor give some indicators for identifying classes, but only one useful trick for finding attributes: draw and label a diagram. I develop this idea in Chapter 6 so that it becomes an integral part of the mathematical modelling process. Shlaer and Mellor favour Chen's notation (Chen 77) for an object diagram, showing relationships between classes. Deriving classes from information flows could cause problems in mathematical modelling where information flows may not be clear.

State models stage

A state diagram examines the dynamic behaviour of the system, and contains useful pointers for mathematical modelling. There are specific directives to be aware of physical laws, established practice and heuristics. The methodology does not clarify how interactions can be manipulated so as to drive a mathematical model.

Process models stage

The purpose of this stage is to construct a Data Flow Diagram (DFD) for each state of the system, which details data stores and actions for that state. Data stores are rarely appropriate for mathematical modelling, but clarifying actions may be.

Shlaer and Mellor recognise that the problem of identifying objects needs more work, although later methodologies appear to have made little headway with this problem.

5.8.4 Coad-Yourdon

The origins of the Coad-Yourdon technique lie in Yourdon's structured design methodology (Yourdon 79) and its extension to structured analysis (Yourdon 89). This method was extended to a formal O-O technique by the addition of an object-analysis stage (Coad 90), and has become known as the *OOA/OOD* method. Although the methodology stresses system organisation through a class hierarchy (consisting of "Class-&-Objects"), there may not be much to organise in a small mathematical context. Coad stresses the importance of problem domain knowledge and gives useful pointers for finding attributes (e.g. states of the system, 'How...' questions, needed information). Similarly, responsibilities, behaviour and links can be used to find methods. Of these, behaviour and links are the most useful for mathematical modelling because they define how mathematical entities interact. The example of Problem 5 (Section 5.4) shows that refining these considerations to produce workable classes is not always straightforward.

5.8.5 Rumbaugh: OMT

The design technique discussed in (Rumbaugh 91) is known as the Object Modelling Technique (OMT). The notation for it is adapted from a preliminary version of OMT in (Loomis 87). It consists of three principal stages which are split into sub-stages as below. The first stage is the most important as far as mathematical modelling is concerned because it deals with the initial design of objects. The other two stages are more applicable for large design projects, and are not discussed further.

1. **Analysis**
   - a) **Define objects**
   - b) **Determine dynamic flows**
   - c) **Determine functional relations**
2. **Systems Design**
   - a) **Define sub-systems**
   - b) **Define concurrency**
   - c) **Define communications and data stores**
3. **Object Design**
   - a) **Define algorithms to implement system functions**
   - b) **Optimise object model**
   - c) **Implement associations of attributes**

The analysis phase depends on constructing three distinct models. The *Object* model is the most relevant for mathematical modelling. Rumbaugh gives few hints on how to derive classes, which are loosely defined as entities which "have identity and are distinguishable, concepts, abstractions or things with crisp boundaries and meaning". When seeking attributes he searches for nouns in a specification or problem statement. The idea of linking objects is a key consideration in this thesis and OMT approaches it by considering verbs and verb phrases in a problem description. There are no hints, apart from seeking generalisations, for deriving abstract classes. The *Dynamic* model determines events in the system, but what these should be is not clear cut in mathematical modelling. Rumbaugh uses the *Functional* model to identify inputs and outputs, although these are usually interpreted as user interactions, data stores or file operations, which have little relevance for mathematical modelling. The end result of building the functional model is to construct a DFD, which is also not particularly relevant for a mathematical model. Hayes and Coleman (Hayes 91) recommend a modified form of OMT which concentrates on keeping the overall model self-consistent. This is useful, but not necessary for mathematical modelling.

5.8.6 Booch

Booch (Booch 94) outlines a micro-process and also a macro-process for class design. The former is more relevant because it deals with initial design, and there are three important stages (1-3, below) for mathematical modelling. Stages 2(a) and 2(b) come from an original 5-stage process in (Booch 86).

1. **Identify classes**
2. **Identify semantics of those classes**
   a) **Identify operations suffered by and requested of each class**
   b) **Establish the visibility of each class in relation to others**
3. **Identify the relationships among classes**

In the *Identify classes* stage, a Data Dictionary ("list of things") is central to Booch's discussion. To derive classes he considers tangible things (nouns), events, interactions and concepts, as well as the system's dynamic behaviour. It is unclear how dynamic behaviour would apply to mathematical modelling. Rubin and Goldberg (Rubin 92) clarify this to a limited extent. They suggest that it is necessary to understand what events take place in the system and to assign these events to parts of the system. Booch also suggests function-point analysis (Dreger 89) (a function-point is an end-user business process) and *Use-Case* analysis (Jacobson 92), but these techniques depend on finding clear processes and data flows, which are rare in mathematical modelling. The purpose of the *Identify semantics* stage is to determine the behaviour and the attributes of the abstractions found in the previous stage. Booch suggests using CRC Cards. In the *Identify Relations* stage, Booch suggests looking for common behaviour between the pairs of classes. He does not say how suitable pairs may be found.

5.8.7 UML

The most recent addition to the set of object modelling methodologies is due to, amongst others, Booch and Rumbaugh, and is known as the Unified Modelling Language, UML (Erikkson 98). It claims to be a synthesis of good practice from previous methodologies, and originates from the Object-Oriented Structured Engineering (OOSE) approach of Jacobson (Jacobson 87). The principal stages of UML are outlined in Figure 5.4. Stages 1, 2 and 3 of the diagram are central to mathematical modelling problems.

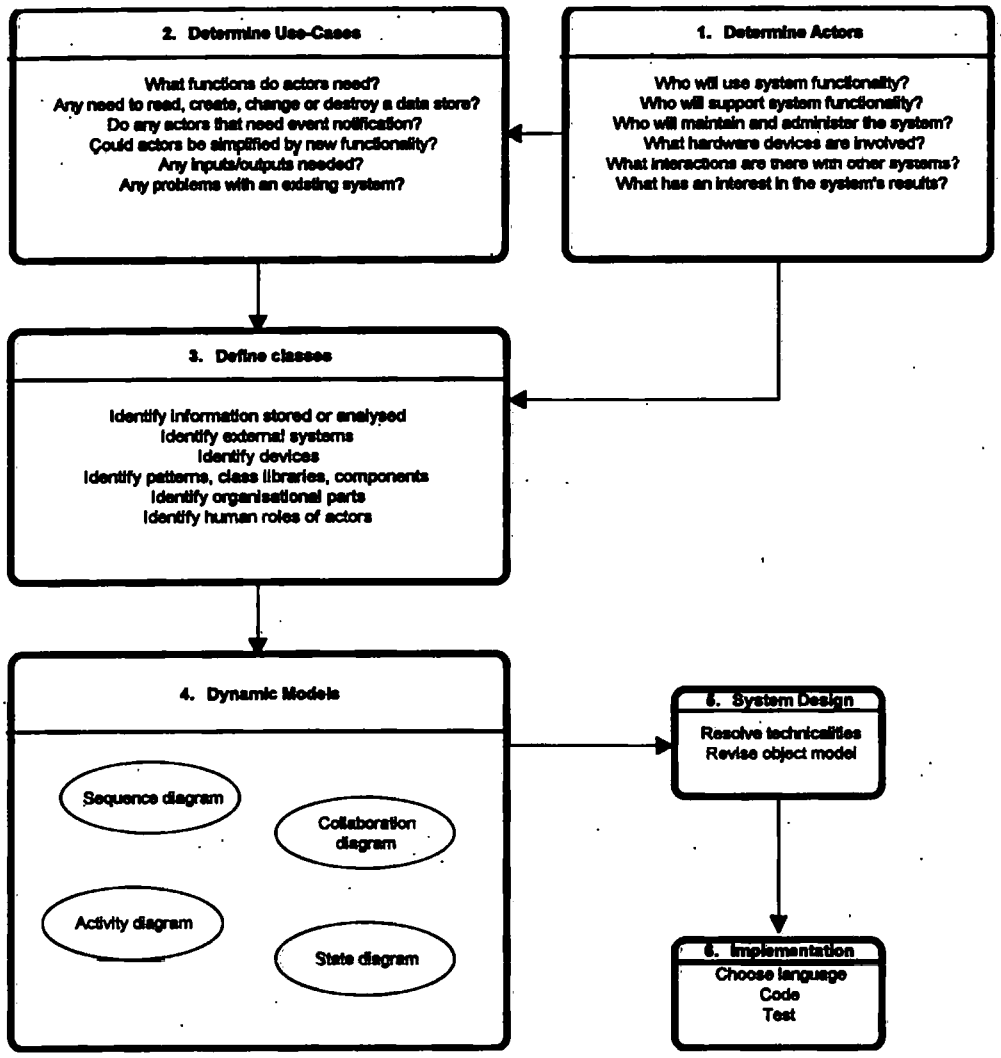


Figure 5.4

Actors are central to UML. Jacobson (Jacobson 92) defines them as "who or what is in the system". Only the last detail in Box 1 of the Figure 5.4 seems relevant for mathematical modelling. It could refer to, for example, an equation of motion. Quatrani (Quatrani 98) is more explicit. He makes the point that actors are *external* to the system and interact with it. They tend to be inputs or outputs, but these are rare in mathematical

modelling. *Use-Cases* refer to functionality as perceived by an actor: they model interaction between actors and the system and tend to be verbs (Quatrani 98). The questions in Box 2 of Figure 5.4 depend on a sequential set of the events, but it is not obvious that such a set of events exists in a mathematical modelling scenario. Candidate *Classes* are derived from actors and their methods are derived from Use-Cases, but UML (in Erikkson 98, Quatrani 98 and Jacobson 92) provides little advice on how to find them. Lee and Tepfenhart (Lee 97) rely largely on the OOD/OOA ideas of Coad and Yourdon to derive classes, and recommend ideas borrowed from Shlaer and Mellor, and Wirfs-Brock to find attributes and specify class behaviour. The *Dynamic Models* of Stage 4 in Figure 5.4 are diagrams which provide distinct views of the system. Producing them for mathematical modelling problems is too large an overhead to be worthwhile. Only the *Collaboration diagram* could be useful for modelling. This describes interactions and links between objects, and these links can drive the modelling process.

UML is another methodology that is intended for designing and maintaining large systems with "obvious" objects and "obvious" processes.[2] Although it is based on analysis of Actors and Use-Cases, it is possible (Lee 97) not to stress their roles. This reduces UML to an aggregation of previous methods. I make a brief attempt to use the UML CASE tool, Rational Rose 4.0 (Rational 98) in Chapter 6, and discuss the problems of doing so in a mathematical modelling context.

[2] Evidence from a current project shows that a CASE tool which uses the UML methodology (Select 96) fails to produce a design that can be coded quickly and easily. It was difficult to integrate the classes generated with a workable database model.

5.9 The object oriented paradigm and mathematical modelling

This section summarises the principal requirements for object-oriented concepts to be applied to mathematical modelling contexts. I argue that no established O-O design methodology contains all the necessary elements.

5.9.1 Classes

Finding classes is the most important activity for mathematical modelling since these are needed to drive the modelling process (in a way described in Chapter 6). Many established methodologies (Wirfs-Brock 90 and Coad 90) stress the importance of seeking nouns in a requirement specification. The mathematical equivalent of a requirements specification is a *problem statement*. Although a problem statement can be worded precisely, the nouns it contains do not always translate into useful objects. For example, they often contain spurious context information. In a Business Systems context noun analysis can be more worthwhile because it is easier to identify potential classes from data sources, data sinks, users and system components. (Wirfs-Brock 90) mentions elements such as these in the context of an ATM. Useful nouns have to be differentiated from non-useful nouns in mathematical modelling problem statements, and this is likely to be a problem for mathematicians who are inexperienced in O-O techniques. There is also a particular problem in that gravity is often implicit (as it is in Problems 1 to 5 of this chapter), and is therefore not formally in the problem statement.

UML poses a particular problem for mathematical modelling. It is hard to find *actors* (external agents), which are needed to determine classes. Gravity can be considered as an external agent but is better treated as internal since the interaction of gravity with massive objects is important in creating weights.

5.9.2 Attributes

It is sometimes hard to distinguish between nouns which are appropriate for identifying candidate classes and nouns which are more appropriate for identifying candidate attributes. For mathematical modelling it is easier to avoid this decision by finalising classes first and then finding their attributes. This is because the classes are often well-defined mathematical entities (particles, forces etc.), and their attributes then correspond to physical characteristics. A "classes first - attributes next" approach makes "overall design" (as in Shlaer 88, Booch 94, Rumbaugh 91) hard. Some methodologies provide little guidance on finding attributes (e.g. UML in Erikkson 98), and are therefore of little use to modellers.

5.9.3 Methods

A key feature of mathematical modelling is how elements in the problem domain interact. This is not addressed in a systematic way by existing methodologies other than through responsibilities (Wirfs-Brock 90) and links (Coad 90). Responsibilities are a potentially useful idea for mathematical modelling, but represent a level of abstraction that isn't strictly necessary. Problem 5 of this Chapter shows that verb analysis can produce spurious verbs which confuse the issue.

5.9.4 Class libraries

Constructing and maintaining a class library and hierarchy is important to structure and maintain large software projects. Existing methodologies are designed to facilitate this process. An elaborate class hierarchy is not necessary for mathematical modelling, although inheritance can be useful (as Chapter 7 shows), and a formal mechanism for constructing a class library is unnecessary.

5.10 Summary

The object-oriented modelling methodologies described in this section have a number of general characteristics.

- They are designed for large software projects which need to record not only class details, but also views of the project. Large projects need to provide a mechanism for design and maintenance because personnel changes, a need to distribute work between members of a team, and changes in requirements are generally involved. This does not apply for mathematical modelling.
- They are written for business applications in which objects are sometimes easier to isolate, particularly abstract ones. Interactions between classes are often easier to define because they correspond to business processes.
- Domain knowledge is necessary, particularly for class design.
- Descriptive diagrams are a major feature of many methodologies, but they can be overwhelming and obscure the class-building process.

The methodologies described in this Chapter partially fulfil the needs of mathematical modelling. Four major requirements are missing:

1. linking the object methodology with key aspects of the practice of mathematical modelling - drawing diagrams, formulating equations and solving them;
2. using objects to drive the way in which modelling proceeds;
3. abstracting ways in which common elements of mathematical modelling problem domains interact, in order to create methods;
4. distinguishing spurious and non-spurious nouns and verbs in a problem statement.

These problems will be addressed in Chapter 6.

Chapter 6

A new Technique and Methodology for Mathematical Modelling

6.0 Abstract

This chapter describes an O-O approach to modelling, as a solution to the problem of elucidating features in the model and constructing relations between them. Empirical evidence in Chapter 5 shows that this problem is central to any modelling cycle, and it therefore has high priority here. It sets out requirements for an O-O methodology which is suitable for modelling. Ideas from existing methodologies, and Wirfs-Brock in particular, contribute to this. First, Wirfs-Brock 'responsibilities' help to identify how objects relate to each other, and hence how to link variables and parameters. Second, a noun-verb analysis corresponds well to an analysis of a model which is to be derived from a textual description. The principal idea advanced is to enhance the role of the modelling diagram, which would normally be drawn when formulating models. This is used to discover modelling features. It is backed up by considering domain axioms and heuristics. These result in the *Diagram-Axiom* modelling methodology, and contributory elements of it are classified. A general modelling heuristic, the *Principle of Adjacency*, results from this analysis. Using it supplements the generic modelling cycle by linking objects in a *Construct→Link→Invoke_Methods* modelling cycle. This entails systematically linking objects to create new objects, until an equation of state for the system results. A short modelling exercise using a contemporary methodology, UML, then highlights deficiencies of established computerised methodologies: they concentrate on processes rather than translating problem domain elements into classes. Illustrations of the *Diagram-Axiom* methodology then show its wide applicability. The cases considered range from those where drawing a diagram is normal practice to those where a diagram is contrived. They also cover problem domains which do and do not have an axiomatic basis. From these, I give an honest assessment of the successes and failures of the *Diagram-Axiom* methodology. The case of Newtonian particle mechanics is particularly important because it is the subject of computerisation in the next chapter.

6.1 Methodology Requirements

An object-oriented design methodology used for mathematical modelling must satisfy the requirements below.

1. It must be consistent with and incorporate "normal" activities of mathematical modelling.
2. It must encapsulate details of space and time coordinates.
3. There must be a way to encapsulate laws, axioms and heuristics within the problem domain.
4. Manipulation of the objects (usually by invoking their methods) should allow for the development of the mathematical model. Important steps in the development of the mathematical model need to be identified and mirrored when manipulating the object model.
5. The methodology should be reasonably simple so that it can be used by users who are not experienced in object-oriented methods. It should be mathematically intuitive and any mathematical ideas that it contains must make sense.

The methodology proposed in this thesis comprises two strands: *Diagram* and *Axiom*. The next two sections explain each of these.

6.2 Diagram Strand

A modelling diagram is a natural part of a mathematical model, and serves here to construct classes. This diagram draws attention directly to elements in the problem domain. In this way, it isolates instances of candidate classes. An assumption is that at some early stage in the development of the mathematical model, a relevant diagram is drawn. It may contain information about, for example:

- the geometry of the problem domain;
- elements in the problem domain;
- symbols for elements in the problem domain;
- kinetic quantities.

Using such a diagram satisfies the requirements stated above as follows.

- Drawing a diagram is "normal" mathematical modelling practice. It is therefore a potentially useful starting point for generating classes (Requirement 1).
- As the diagram is constructed, object methods can be invoked which drive the modelling process. Therefore, components of a mathematical model can be constructed as the diagram is drawn (Requirement 4).
- Object-oriented techniques are possibly alien to mathematicians but this difficulty is eased by the process of drawing a diagram, which is a familiar activity. Otherwise, it is unlikely that this problem can be eased further (Requirement 5).
- Certain common elements are "known" components of mathematical modelling domains. These may not be apparent in drawing a diagram. Examples include an equation of state, space and time coordinates x, t. (Requirements 2 and 3)

This approach raises certain difficulties.

1. If a diagram cannot be drawn, the technique cannot proceed in the form outlined. Either the user must contrive a diagram or there must be an alternative to using a diagram.
2. It is not normal to analyse the order in which components of a mathematical modelling diagram are drawn. However, it is quite clear that some order exists, even if it is arbitrary, because an entire diagram cannot be drawn in one stroke. Given that some order exists, it is possible to analyse the precise role of each element drawn, as it is drawn.
3. The precise definition of an "element" in a diagram is unclear at this stage. In order to clarify this term, suppose that a finite set of graphic shapes is available in a computer graphics package. This ensures that a diagram comprises only these shapes (with the possibility of resizing them). Each graphic placed on the screen is then an object.

6.3 Axiom Strand

Considering axioms, laws and heuristics complements the *Diagram* strand and attempts to fulfil functionality that cannot be achieved by use of a diagram. It also performs an important role in its own right in identifying candidate classes. This is because it draws attention to the fundamental behaviour of the system and provides a focus for relationship formation. The *Axiom* strand gathers information about class attributes, which classes can and should collaborate and the result of that collaboration. Heuristics provide information about how the system behaves, but using heuristics is more difficult because they are less well defined.

The *Axiom* Strand satisfies the requirements for a mathematical modelling object design methodology in the following ways:

- Many mathematical scenarios involve laws or axioms. For example, problems in mechanics have Newton's Laws as their basis, and use the Second Law particularly (so much that writing "F = MA" is instantly recognisable, even though the symbols are undefined). (Requirements 2 and 3).
- Problem specifications nearly always contain heuristics about space and time co-ordinates. (Requirement 2).
- Using axioms and laws is a natural part of mathematical modelling, and obtaining an equation of state is likely to be either a total or a partial goal of the modelling process (Requirements 1,4 and 5).

The difficulties with this approach are:

1. It may not be convenient to make space and time co-ordinates independent of other objects in the problem domain. Hence, it may not be strictly necessary to abstract time and space co-ordinates.
2. Capturing accurate domain heuristics is a subjective process.
3. The *Axiom* strand cannot be used in isolation because it is not sensitive enough to define all required class instances.
4. It may not be clear which form of a particular law should be encapsulated (e.g. Newton's 2nd Law in the form *Force* = *Mass* × *Acceleration* and *Force* = *Rate of Change of Momentum*).

6.4 A new methodology for mathematical modelling

The schemes below list components in the *Diagram* and *Axiom* strands. They are categorised but not presented as an algorithm. With experience, they can be used in any order, and imposing an order on them could result in an algorithm which takes a long time to follow, with a null result returned from many of its stages. The second part of this chapter gives examples.

1. **Diagram Strand**

   a) Identify Classes: for each element drawn:

      i) **[DC1]** Decide if it is an instance of an existing class

      ii) **[DC2]** Link with nouns in a problem statement

      iii) **[DC3]** Look for nouns in a problem statement which do not correspond to elements on the diagram

   b) Identify Attributes: for class identified:

      i) **[DA1]** Define characteristics

      ii) **[DA2]** Define properties

      iii) **[DA3]** Describe using the 'has-a' relation

      iv) **[DA4]** Express as a possessive: ObjectName's attr ▯ ObjectName.attr

      v) **[DA5]** Consider labels on the diagram

   c) Identify Methods: for each class identified:

      i) **[DM1]** Link with verbs in a problem statement

      ii) **[DM2]** Consider consequences of linking an instance of this class with an instance of another class

      iii) **[DM3]** Consider responsibilities and associated collaborations

      iv) **[DM4]** Express as a subject-verb-object sentence: <Object Name> <Verb Phrase> <Target Object> ▯ ObjectName.VerbPhrase(TargetObject)

      v) **[DM5]** Consider candidates from arrows indicating links on the diagram

d) Construct Class Hierarchy: for each pair of classes:

i) **[DH1]** Consider 'likeness' of characteristics and behaviour

ii) **[DH2]** Look for additional attributes in one class compared to another

iii) **[DH3]** Look for additional methods in one class compared to another

iv) **[DH4]** Look for override methods in one class compared to another: same name - different action

v) **[DH5]** Identify candidate abstract classes

2. Axiom Strand

a) Derivations using Laws

i) **[AL1]** Write law in words

ii) **[AL2]** Write law in symbols

iii) **[AL3]** expressing the law as a class

iv) **[AL4]** Assign candidate attributes: each word/symbol in the law

v) **[AL5]** Assign each attribute to a class

vi) **[AL6]** Identify methods from verb phrases relevant to the law

vii) **[AL7]** Identify attributes from noun phrases

b) Derivations using Heuristics of the problem domain.

i) **[AH1]** Write statements of problem-solving and implementation techniques in subject-verb-object format

ii) **[AH2]** Identify classes from subjects in statements

iii) **[AH3]** Identify attributes from objects in statements for which verb = *has*

iv) **[AH4]** Identify methods from verbs in statements

v) **[AH5]** Use domain knowledge

6.5 Mathematical Modelling by Linking Objects

Assume that all necessary object instances have been identified and coded, and that a class hierarchy has been built. This section gives a brief indication of how modelling then proceeds and provides a context for the examples that follow. Principles are noted here for reference and there is a fuller discussion in the next chapter.

Modelling proceeds by formally linking objects. The result is either a new object or a non-object-based expression, such as the value of an object attribute. This process should be seen in terms of the question 'What do you get when you combine an object A with an object B?'. For example, if a force is combined with another force, the result is a third force, all the characteristics of which are calculable from the inputs. The goal is to create an equation of state of the system, such as an equation of motion for Newtonian dynamics. This is not necessarily a complete decision procedure, and this part of the process requires further research, as does the equivalent problem in any modelling methodology.

As a simple illustration of these principles, consider a situation in which a 1-D force F(t) acts on a particle of mass m. Without loss of generality, the x-axis can be taken as the direction of motion. The first stage is to create a particle with two attribute: Mass, m, and coordinate x[t]. This uses components **DC1, DC2** and **DA3** of the Diagram-Axiom methodology.

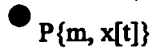


Figure 6.1a

The second stage is to create the force, which has a Components attribute {F(t)}. This uses components **DC1, DC2, DA3, DA1** and **DA5** .

Figure 6.1b

Linking the two objects (component **DM2**) produces an equation of motion, which can be expressed as an *EquationOfMotion* class. Calling a method, *GetEquationOfMotion*, of the *EquationOfMotion* class (component **AL6**) then returns the equation of motion. The modelling process is therefore summarised by the following sequence.

```
Create Particle(P, m, x[t])
Create Force(F, F(t))
LinkObjects(P, F)              [produces an EquationOfMotion object EoM) ]
EoM.GetEquationOfMotion        [returns m x''[t] = F(t) ]
```

In Section 6.8 I discuss variations on this example, and give examples from other contexts.

6.5.1 The Principle of Adjacency

An important modelling principle developed in this thesis is the **Principle of Adjacency**. This requires that any two objects which can be linked must be physically adjacent to each other on a diagram. The rationale behind this principle is that objects can only interact if there is a physical or logical bond between them, and that this bond can be illustrated in diagrammatic form. This principle aids the modelling process in two ways.

1. It provides guidance as to which objects can or should be linked.
2. It lessens the chance of linking two objects in an inappropriate way (and may actually prevent such a link in a software implementation).

In the examples that follow, application of the Principle of Adjacency is indicated by **[PA]**.

6.5.2 Alternative approaches for linking objects

Two differing approaches to linking objects were apparent in developing an O-O modelling methodology. The second of these was easier to program, so this was used.

The first approach involves a call to a class method which has, as one of its arguments, an instance of another class. Suppose that an instance A_{inst} of class A (e.g. a *Particle*) and an instance B_{inst} of class B (e.g. a *Plane*) are already in the problem domain. The link can proceed by calling a *LinkObjects* method of class A, which has B_{inst} as its argument: $A_{inst}.LinkObjects(B_{inst})$. The result depends on A and B. In the case

A = *Particle* and B = *Plane*, the result is a new object: a contact force. This is consistent with O-O principles but had two problems.

- It proved to be awkward to program the necessary LinkObjects methods. A function LinkObjects was required for each object A in the problem domain which could link with each relevant other object B. Multiple versions of this function were therefore required as methods for each object.
- This approach is not consistent with a mathematical modelling approach that depends on linking objects by considering only instances of those objects. In effect, this approach proved to be more suitable for programmers but less suitable (conceptually) from a mathematical point of view.

The second alternative was to define a polymorphic[1] function, *LinkObjects*, that is not formally a part of any class in the system. Its arguments are class instances. Thus, to link the instances A_{inst} and B_{inst} of the classes A and B, an appropriate form of *LinkObjects* is used: *LinkObjects(Ainst, Binst)*. Mathematica, C++ and other programming languages can distinguish which form of *LinkObjects* is appropriate by examining the arguments supplied. This is not good O-O style because it is divorced from the principle that message passing is done by method calls. It does have the following advantages.

- It was easier to program in Mathematica, because it was possible to identify a small set of 'sensible' links in the problem domain. It was then easy to define a version of LinkObjects for each 'sensible' link.
- It emphasises a formal mathematical link between objects, which is the basis of this modelling methodology.
- It prevents 'non-sensible' links because no LinkObjects template for 'non-sensible' links exists. The system is therefore less error prone.

The advantages of the latter 'template' approach outweighed its disadvantages, so this was the approach adopted in the detailed analysis in Chapter 7.

[1] one that has a different form, depending on differing circumstances, but has the same name in all cases

6.6 Abstract classes

Abstract classes are not essential for many of the mathematical problem domains considered here, but two useful pointers to defining candidate abstract classes have emerged.

1. Consideration of commonality - like objects could be cast as descendants of a common ancestor. In particle mechanics I found few clear examples. Horizontal and inclined planes are similar because they have a common attribute: a coefficient of friction, and behave mathematically in the same way. The inclined plane has an additional attribute: an angle of inclination to the horizontal. Thus, an inclined plane can be considered as a horizontal plane with an extra attribute.
2. Identification of objects which are 'properties of the problem domain'. Time and space co-ordinates often fit this category since they are shared and often required by all objects in the problem domain. This phenomenon occurs extensively in the detailed implementation in Chapter 7. Entities such as particles, springs, and forces, which have very different mathematical properties, all have co-ordinate attributes. They can therefore be descendants of a common *Coordinate* class, which has the co-ordinate attribute they all need.

6.7 Alternative O-O Modelling Approaches

In Chapter 5 I reviewed several O-O modelling methodologies, and pointed out their shortcomings for mathematical modelling. In order to demonstrate the practical difficulties of using an established methodology, and thereby stress the superiority of the Diagram-Axiom methodology of this chapter, I attempted to use UML to solve a simple modelling problem. UML is the most recent of the established design methodologies and is (supposedly) a synthesis of the most useful aspects of its predecessors (Booch and OMT in particular). It is also supplemented by several CASE tools. I did a top-level design using one of these, Rational Rose (Rational 98), which implements most aspects of the UML methodology. The particle mechanics problem was a very simple one:

A particle of mass m is released from rest and falls under gravity in conditions of no resistance. Find the equation of motion.

The comments that follow are not just a function of the simplicity of this problem: design methodologies are really there to structure, document and simplify much more complex domains. What is of interest here is whether or not the methodology provides sufficient insight into class construction and object linking. A Use Case view, a Class view and a Sequence view were constructed, and screen dumps of each diagram are shown below. The problems encountered in each phase were as follows.

Use Case View:

- It was hard to identify candidate Actors, and to distinguish between external and internal Actors. In the problem statement the only explicit candidates are 'particle', 'equation of motion' and 'gravity', and the lack of other explicit objects makes concrete and abstract classes hard to identify.
- Domain knowledge is needed to derive attributes for candidate classes: neither the methodology nor the software provide sufficient guidance for this. There was no particular reason for including the velocity attribute (other than it might prove to be useful), although the acceleration attribute is clearly necessary to form an equation of motion.
- Few Use Cases were identifiable. The usefulness of the ones listed in the Use Case diagram is doubtful other than to clarify concepts and ideas.

The result was a very sparse Use Case diagram (Figure 6.2) which did not lend itself to proceeding with the next stage: class construction.

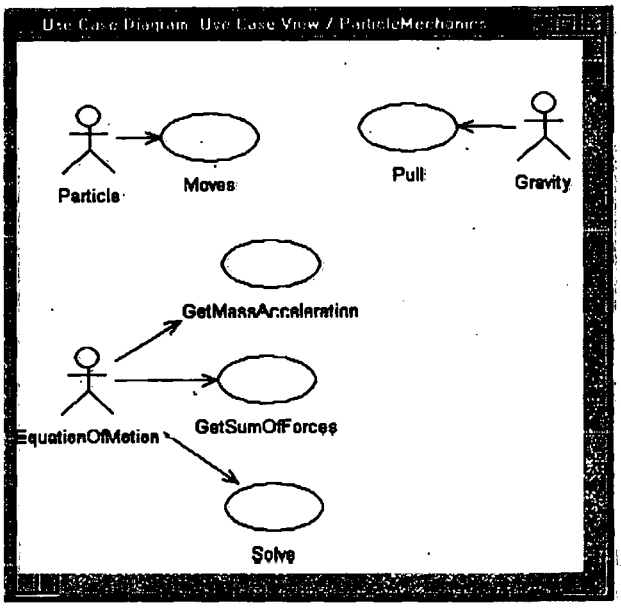


Figure 6.2

Class View:

- Several candidate classes were listed, including the 'obvious' Actors mentioned above, but the lack of Use Cases hindered progress with others.
- Ill-defined Time and Coordinate classes were identified (perhaps luckily) from operational heuristics and experience. The software provided little guidance about resolving methods, other than to remind the user that a typed argument list may be necessary.
- Placing and naming associations on the diagram provided some clues about which objects can be linked in a sensible way.

Some of the links in the Class Diagram (Figure 6.3) are intended to convey a sense of responsibility (e.g. Gravity and Particle are both responsible for constructing Weight)

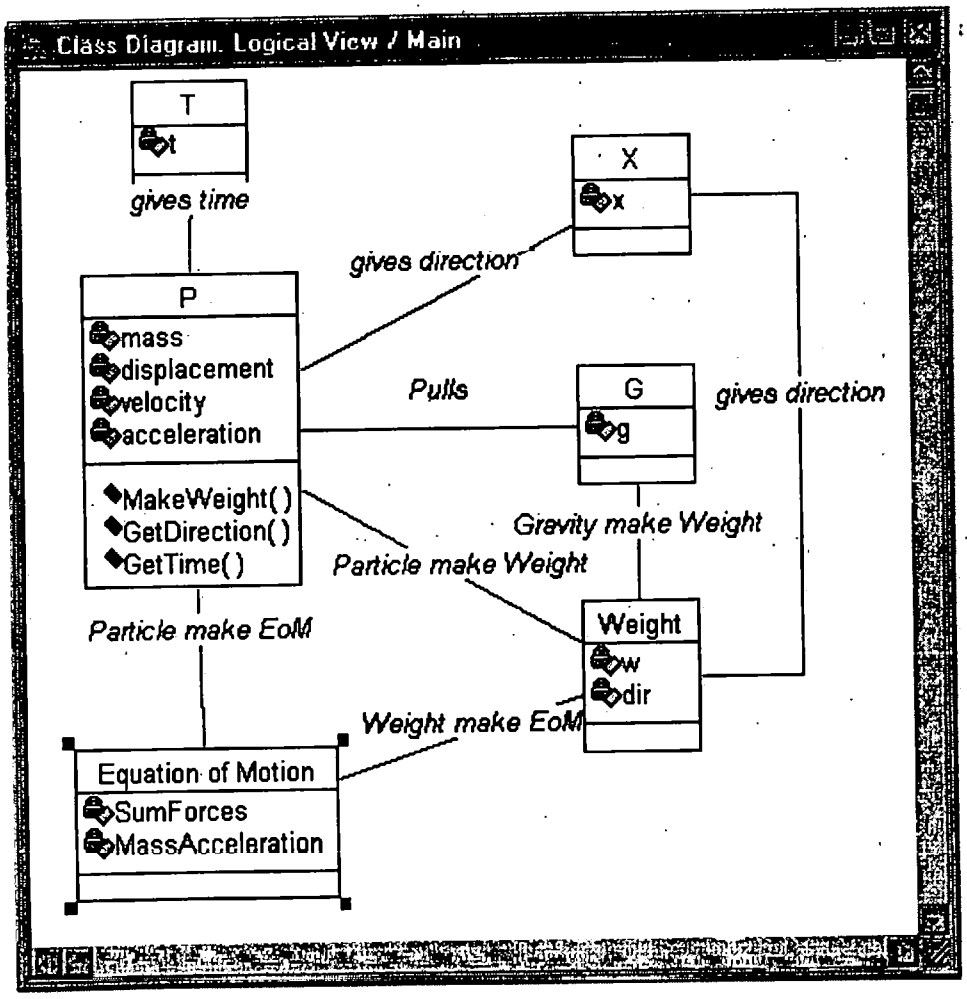


Figure 6.3

Sequence View:

The Sequence view (Figure 6.4) was the most useful view since it provided links between objects, which were essential for the modelling process. This assumed, however, that the objects in the diagram were placed in a sensible order. For example, the Weight object cannot be manipulated before it has been created.

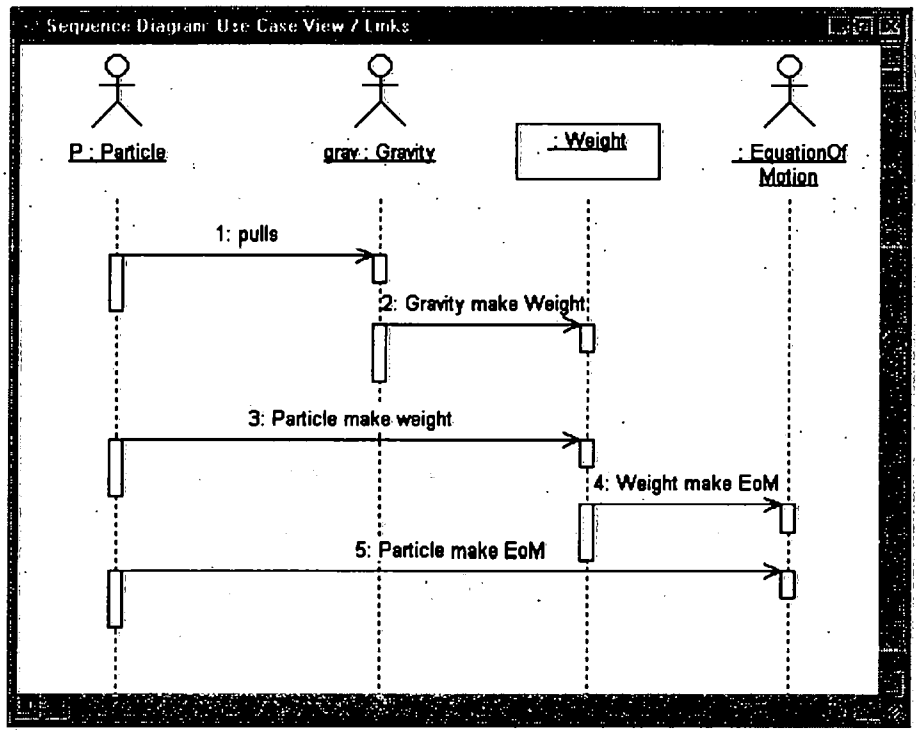


Figure 6.4

State Diagrams:

The only State Diagram (Figure 6.5) is of doubtful use. It merely identifies the cases $t = 0$ (initial velocity and displacement are zero) and $t > 0$, which is rather trivial. There is a possible application in providing initial conditions for solving a differential equation.

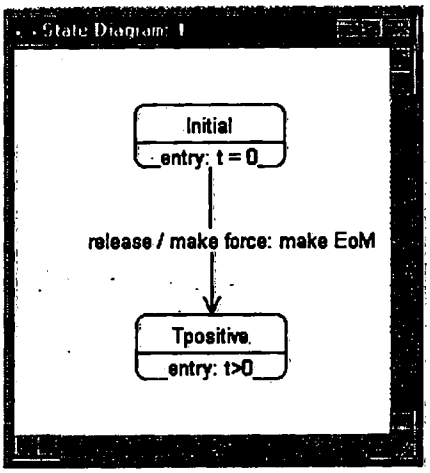


Figure 6.5

Overall, the UML method and associated software did not tackle basic problems with this application, which were to provide methods for doing the modelling, and to help to identify objects, their attributes and way they interact. Drawing a diagram, as would be done as a normal part of the modelling process, would provide these functionalities in a much more explicit way. It makes sense to make more use of this 'natural' feature of modelling.

6.8 Examples

In this section I illustrate the Diagram-Axiom methodology with a number of detailed examples. They show the essential processes of constructing and linking objects, and how these generate a mathematical model. Because the Diagram-Axiom methodology is ideally applicable to a modelling situation in which it would be normal practice to draw a diagram, I chose common modelling scenarios where a diagram would normally be present, but also others where a diagram would not normally be present. In all the examples, labels such as **[DA4]** show which components of the Diagram-Axiom methodology (section 6.4) are used, and the symbol ⇨ denotes 'results in'. The examples cover cases A-E in the list below.

A. Principal information source is the Diagram Strand: the model has an axiomatic basis.
B. Principal information source is the Diagram Strand: but few or no operational heuristics.
C. Principal information source is the Axiom Strand but the Diagram Strand is useful.
D. Principal information source is the Axiom Strand and the model is based on axioms rather than heuristics: diagrams not normally present.
E. Principal information source is the Axiom Strand and the model is based on heuristics rather than axioms: diagrams not normally present.

The first example contains more detail than subsequent examples in order to illustrate small details.

6.8.1 A detailed example: Newtonian particle mechanics

Consider a case of a particle of mass m subject to two forces F_1 and F_2, with known direction and magnitude or known components in 3-D. This is an example of Model Type A, since it would be normal to draw an annotated diagram in these cases. It is a simple extension of the earlier example in Section 6.5, but illustrates some modelling heuristics which are absent in the earlier example. A diagram can be constructed as follows.

First placement: particle, with attribute Mass, m **[DC1, DC2, DA3]**:

Figure 6.6a

Second placement: Force #1, with attribute Components {x1,y1,z1} (sufficient to characterise it up to an equivalence of vectors with these components **[DC1, DC2, DA3, DA1, DA5]**:

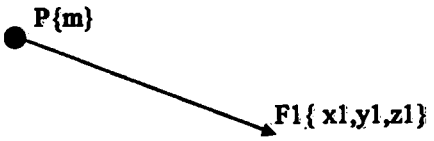


Figure 6.6b

At this stage it is possible to link the two objects **[DM2]**. The result is an equation of motion, which can be expressed as an *EquationOfMotion* class. However, it does not make sense then to combine the *EquationOfMotion* object with the second force **[DM3]**. The second force must therefore be added to complete the diagram.

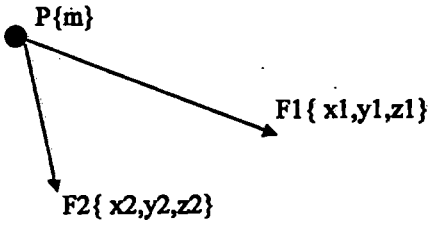


Figure 6.6c

This completes the Diagram Strand, leaving the Axiom Strand. The principal axiom is Newton's 2nd. Law of Motion (often written as simply F = ma), which indicates a link between forces, mass and acceleration **[AL1,AL2]**, and that one such link exists for each massive object in the system (there is only one here). Furthermore, the acceleration term points to the existence of further coordinate attributes of the particle, since these are connected to the mass term, which refers to the particle. Also, a special heuristic of this type of model is that a space-time coordinate system must exist **[AH1]**. Thus the final diagram is amended to:

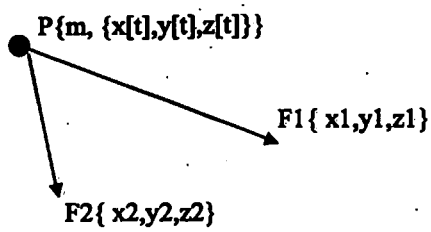


Figure 6.6d

Only certain links between class are sensible in this scenario. They are **[AH1, AH2]**:

- Force + Force = Force
- Particle + Force = Equation of Motion

The modelling process can then proceed as follows.

LinkObjects(F1, F2)	[produces a third (resultant) force F3]
LinkObjects(P, F3)	[produces an EquationOfMotion object EoM]
EoM.GetEquationOfMotion	[[**AL6**] method to return the text of the Equation of Motion]

F3 can be characterised as {Force, {x1+x2,y1+y2,z1+z2}} and
EoM can be characterised as

{ EquationofMotion, {m x''[t] = x1+x2, m y''[t] = y1+y2, m z''[t] = z1+z2}}.

The LinkObjects procedures must ensure that these details are returned correctly.

6.8.2 Further examples

The following examples provide less detailed analyses, but stress the role of constructors, and applications in other modelling contexts. Each covers one of the cases A-E (at the start of Section 6.8). They illustrate how to apply the *Diagram-Axiom* methodology, and how successful it is in generating the model.

6.8.2.1 Linear Models for Heat Transfer

This is another example of Model Type A. Its diagram is particularly useful because of the large number of symbols, relations and equations involved in a relatively small problem. Unfortunately, an O-O treatment does not help with this as the number of symbols in the system increases as it is decomposed into objects. The examples in this section are based on the discussion in (Berry 95), and the following calculation is typical (although it's not a modelling problem).

"A layer of insulation of thickness t and U-value U is fixed to the inside of a brick wall of thickness s and thermal conductivity k. The inside and outside temperatures are T_{in} and T_{out} respectively. Calculate the U-value of the combined wall and insulation."

This poses problems for an O-O approach because axioms and heuristics are not so clear cut as in Newtonian mechanics. Hence, the Axiom Strand is needed at the start of the analysis to clarify the precise nature of the classes and how they link. This is important in more complex examples, which can appear to be more complicated than they actually are if basic principles are unclear or wrong. One such case that involves convective heat loss will be discussed later. Useful assumptions for this type of system are:

- The Fourier linear heat transfer law: *Steady state rate of heat flow through unit area* = *U-value* × *Temperature difference* **[AL1, AL2]**
- The system is composed of layers
- The rate of heat flow through each layer has the same numerical value at any given time. This allows each object to have the same symbol for the rate of heat flow.
- In buildings which require insulation, it is reasonable to assume that the outside is colder than the inside. This allows standardisation of a 'hot' and a 'cold' temperature attribute for each *Layer* object, as discussed below.

In this case, the outside face of the brick wall corresponds to the 'outside', but this would not be the case if there were a convective layer immediately adjacent to the wall. A solution is to construct a separate object to represent the 'outside' (and also the 'inside') **[AH2]**. This is not strictly necessary but does allow for a more flexible object domain, even though it complicates the analysis. Thus, there are two principal classes in the model: *Region* and *Layer*. Modelling starts by drawing one of each, the outside (OUT) **[DC3]** and the wall (WALL) respectively **[DC1,DC2]**. Considerations

[DA1,DA3] decide the characteristics of each. Each surface in Figures 6.7 below represents one square unit.

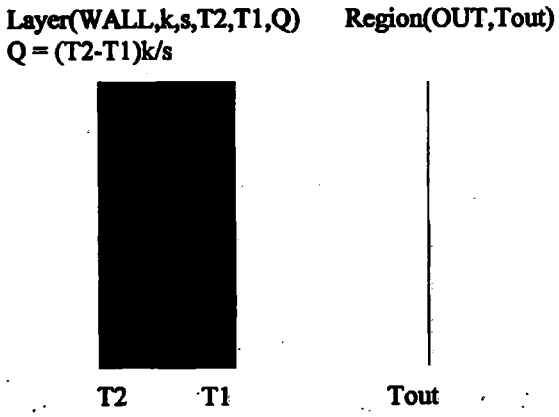


Figure 6.7a

Linking these two objects qualifies the attributes of WALL **[DM4, DM2]**:

LinkObjects(WALL, OUT) → Layer(WALL, k, s, T2, Tout, Q = (T2-Tout)k/s)

The next stage of the modelling sequence is to draw the insulation *Layer* object **[DC1, DC2]** and the inside *Region* objects **[DC3]**.

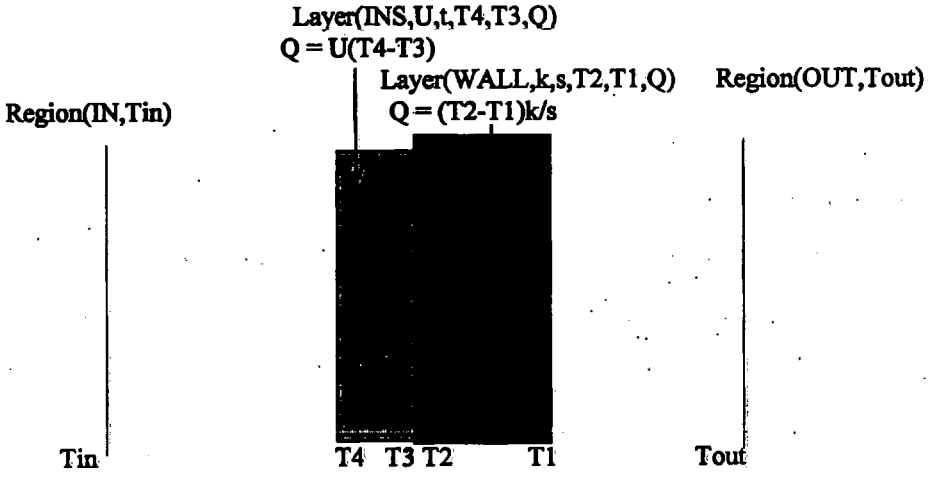


Figure 6.7b

Linking the IN and the INS objects has the same effect as linking the OUT and the WALL objects: it qualifies the INS object. Programming this link requires the temperatures {Tin, T4, T3, T2, T1, Tout} to be in the correct order since the correct temperature in the *Layer* object must be replaced when this link is made **[AH1,DM2,DM3]**. Hence:

LinkObjects(INS, IN) → Layer(INS, U, t, Tin, T3, Q = (Tin-T3)U)

It appears to be possible to link an 'inside' Region with an 'outer' layer (e.g. IN and WALL). This is possible in this theoretical analysis, but it does not make sense in practice. The Principle of Adjacency prevents errors **[PA]**. If links are developed at the same time as objects are constructed, this heuristic could be implemented in software by recording adjacency of objects and disallowing links between non-adjacent objects **[AH1,DM2,DM3]**.

A *Layer* object can have either a U-value or a thermal conductivity attribute one (only one is necessary). This can be expressed by having an attribute for both, and using a null value for the missing datum. In this analysis, which does not involve a formal implementation, the distinction between them is clear, despite the potential for confusion. Consequently, the same attribute is used for either the U-value or the thermal conductivity - whichever is appropriate. Similarly, the thickness of the insulation layer, *s*, is not strictly required since its role is subsumed in quoting a U-value. A second 'derived' attribute is the rate of heat flow, which is expressible in terms of other attributes **[AL7,DA1]**. Nevertheless it is a useful feature in this analysis because it stresses the fact that there is a rate of heat flow associated with a *Layer* object. It could be implemented as a GetRateOfHeatFlow method **[AL6]**.

A further link of the INS and WALL objects results in a new layer object which represents the combination of the two.

LinkObjects(INS, WALL) → Layer(COMB, Ucomb, t+s, Tin, Tout,
Q = (Tin-Tout)Ucomb)

The software must calculate Ucomb as follows:

From INS; $Q = (Tin-T3)U$

From WALL; $Q = (T2-Tout)k/t$

Link INS, WALL; $T2 = T3$

Eliminate T3; $Tin - Tout = Q(1/U + t/k)$

The last stage is to call a GetUValue method of a Layer object to retrieve the required U-value, Ucomb:

COMB.GetUValue() $\rightarrow$ $Ucomb = (1/U + t/k)^{-1}$.

Summarising:

- the O-O analysis is long-winded in this case, but it does necessitate analysis of the problem domain in terms of layers, which is useful for correct analysis of heat flows.
- A further heuristic is useful for this modelling strategy: link a *Region* with a *Layer* before linking two *Layers*.

To qualify this further heuristic, consider what might happen if the two *Layer* objects are linked before linking either one to a *Region* object first.

LinkObjects(INS,WALL) $\rightarrow$ Layer(COMB, Ucomb, t+s, T4, T1,
Q = (T4-T1)Ucomb)

LinkObjects(COMB,OUT) $\rightarrow$ Layer(COMB, Ucomb, t+s, T4, Tout,
Q = (T4-Tout)Ucomb)

LinkObjects(COMB,IN) $\rightarrow$ Layer(COMB, Ucomb, t+s, Tin, Tout,
Q = (Tin-Tout)Ucomb)

COMB.GetUValue() $\rightarrow$ $Ucomb = (1/U + t/k)^{-1}$ [as before]

The following sequence appears to give a correct result but is invalid because Ucomb is derived from an equation which is not formally based on an 'inside' and 'outside' temperature difference **[AH1]**.

LinkObjects(INS,WALL) $\rightarrow$ Layer(COMB, Ucomb, t+s, T4, T1,
Q = (T4-T1)Ucomb)

COMB.GetUValue() $\rightarrow$ $Ucomb = (1/U + t/k)^{-1}$ [as before,
but based on the equation $T4 - T1 = Q(1/U + t/k)$, not $Tin - Tout = Q(1/U + t/k)$.]

In order to prevent this type of error in software, two boolean attributes could be added to the *Layer* object. Each would be set to *True* if and only if a link with an inside or an outside *Region* object has been done **[AL4]**. The GetUValue() method would then be callable if and only if both boolean attributes have been set to *True*.

Adding a further feature to the model, suppose that in addition to the features already described, both inside and outside surfaces are subject to convective heat loss, and that the coefficients of convective heat loss are H_{in} and H_{out} respectively **[DA4]**. In the following analysis, extra *Layer* objects are constructed and added to the existing diagram. They represent layers in which convection is active. This aspect would not normally be drawn on a diagram, but it reinforces the model for convective heat loss. *Figure 6.7c* is therefore very similar to *Figure 6.5b*, but has the two new Layer objects and revised symbols for temperatures **[DC1, DC2]**. They make the scheme $T_{in} > T_i > T_j > T_{out}$ $(i > j)$ consistent.

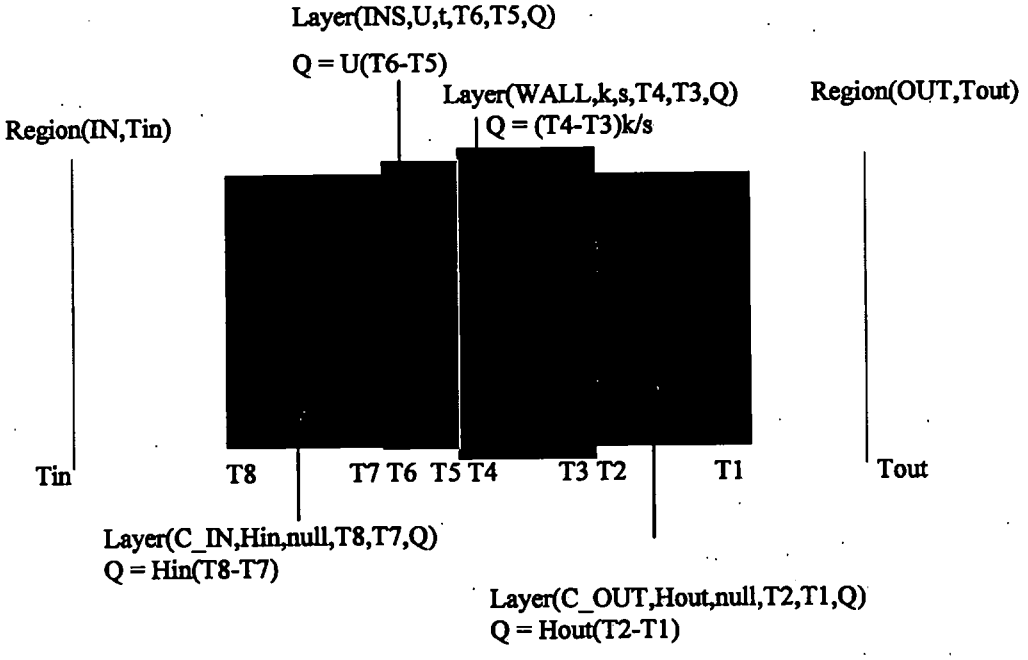


Figure 6.7c

The model then proceeds as follows. Each time two layers are linked, a new Layer object is created **[PA]**. These links are done working from inside to outside, although any sequence involving adjacent objects would be just as good.

Construct Region(OUT, Tout)

Construct Region(IN, Tin)

Construct Layer(WALL,k,s,T4,T3,Q = (T4-T3)k/s)

Construct Layer(INS,U,t,T6,T5,Q = U(T6-T5))

Construct Layer(C_IN,Hin,null,T8,T7,Q = Hin(T8-T7))

Construct Layer(C_OUT,Hout,null,T2,T1,Q = Hout(T2-T1))

LinkObjects(IN, C_IN) → Layer(C_IN,Hin,null,Tin,T7,Q = Hin(Tin-T7))

LinkObjects(OUT, C_OUT) → Layer(C_OUT,Hout,null,Tout,T7,
Q = Hout(Tout-T7))

LinkObjects(C_IN,INS) → Layer(COMB1,U1,t,Tin,T5,Q = U1(Tin-T5))
$U1 = (1/U + 1/Hin)^{-1}$

LinkObjects(COMB1,WALL) →Layer(COMB2,U2,t+s,Tin,T3,Q = U2(Tin-T3))
$U2 = (t/k + 1/U + 1/Hin)^{-1}$

LinkObjects(COMB2,C_OUT) → Layer(COMB3,U3,t+s,Tin,Tout,
Q = U3(Tin-Tout));
$U3 = (t/k + 1/U + 1/Hin + 1/Hout)^{-1}$

Three points are notable from this analysis.

- An attribute value 'null' has been given for the thickness of the convective layers, but the functionality of this parameter is included in the convective heat transfer coefficient. Physically, such a value makes no sense because the thickness of such a layer is not well-defined.
- In theory, a link such as LinkObjects(IN, C_OUT) is possible, but violates the Principle of Adjacency. An alternative strategy would be to designate a *Region* object as either an inner or an outer region by supplying a further attribute: Region(Unique_ID, Temperature, InOut_Type). Linking an 'inner' with an 'outer' object can then be prevented. This solution is clumsy because it may be difficult to supply a consistent definition of the concept of 'in' and 'out'. The Principle of Adjacency is more generally applicable and reinforces the way links can be constructed when objects are created.
- The result of this process is a standard expression for computing the U-value of a composite material. The point at issue here is the modelling process, not the simplicity or standard nature of the result.

6.8.2.2 Linear Heat Flow with Heating/Cooling

This section uses the results of the previous section to formulate a heating problem which involves a heat sink (or a heat source). This extension illustrates two points:

1. an existing object domain can easily be supplemented by more objects;
2. how the attributes of an additional object may be elucidated by considering how it relates to existing objects.

Consider the following problem.

"An object cooled to T_{init} degrees is placed in a cubic container, the walls of which consist of a rigid layer of thickness u and thermal conductivity p, lined by an insulating layer of thickness v and thermal conductivity q. Each face of the cube has area A. The outside of the cube is subject to convective heat loss with coefficient of convective heat trnasfer h. The task is to model the way in which the temperature of the cooled object varies with time."

Since this scenario involves heat transfer through a surface, it makes sense to consider a new object which either supplies or extracts heat. It would be possible to draw such an object on a diagram representing this problem, but it helps here to try the Axiom Strand first to see what its characteristics might be. Characteristics of a heat source are **[AH3, AL7,DA1,DA3]**:

- its mass m;
- its specific heat c;
- it is subject to the heat loss law:

  Rate of heat loss = *mass* × *specific heat* × *temperature gradient* **[AL1]**;
- temperature gradient is usually modelled by a differential dT/dt, in which the variables are temperature T and time t **[AL2]**;
- the initial temperature T_{init} of the heat source is constant.

These properties provide the attributes of a *HeatSource* object, which can be written as HeatSource(HS, m, c, T, T_{init}, t). Its methods must include one to link with a *Layer* object because a *HeatSource* object will be placed adjacent to a *Layer* object when the diagram is drawn **[DM4, PA]**. This link derives an equation (of state) for heat flow

[DM1,DM2], and it is convenient to model this as a *HeatEquation* object. The details of these two objects are apparent from the analysis which follows Figure 6.7d.

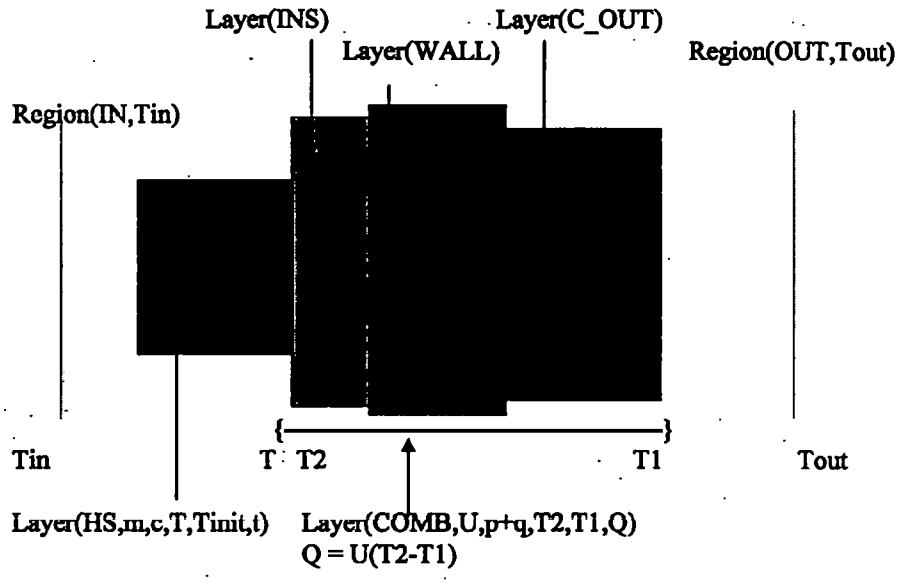


Figure 6.7d

To simplify the analysis, the three layers INS, WALL and C_OUT have been combined to make a composite *Layer* object COMB with U-value $U = (u/p + v/q + 1/h)^{-1}$ and a temperature profile as shown in Figure 6.7d. The diagram stresses the link with the heat source, which can either subsume the role of the 'inside' *Region* object, or be linked to the latter as follows.

Construct Region(OUT, Tout)
Construct Region(IN, Tin(t))
Construct Layer(HS,m,c,T(t),Tinit,t)
Derive Layer(COMB,U,p+q,T2,T1,Q= U(T2-T1)),
$U = (u/p + v/q + 1/h)^{-1}$

LinkObjects(IN, HS) → Layer(HS,m,c,Tin(t),Tinit,t)
LinkObjects(COMB,OUT) → Layer(COMB,U,p+q,T2,Tout,Q= U(T2-Tout))
LinkObjects(COMB,HS) → HeatEquation(HE,-mcD[Tin(t)-Tout,t],
6AU(Tin-Tout),{0,Tinit})

HE.GetHeatEquation → $-mc\frac{\mathrm{d}}{\mathrm{d}t}\left(T_{in}(t)-T_{out}\right)=6AU\left(T_{in}(t)-T_{out}\right)$

HE.Solve → $T_{in}=T_{out}\left(1-e^{-\frac{6AUt}{mc}}\right)+T_{init}e^{-\frac{6AUt}{mc}}$.

This solution is at least plausible **[DM2]**, since its functional form is as in *Figure 6.7e.*

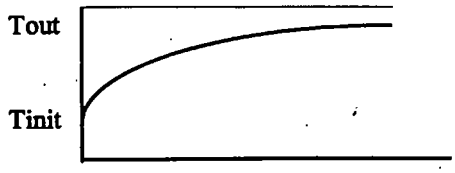


Figure 6.7e

The following points are notable from this analysis.

- Correct signs are needed in deriving the heat equation.
- The method is sufficiently robust to produce a correct result when Tin < Tout, which is the opposite way round to the first of the heat flow problems in this section (which used Tin > Tout). This stresses the geometrical nature of the problem, and why the diagram performs a key role.
- Arguably, it may be useful to add a *Direction_of_Heat_Flow* object to the diagram **[DA5, DM5]**. This would stress that the direction of heat flow is opposite to the direction of increasing temperature.

6.8.2.3 Newtonian Particle Mechanics: Spring-Dashpot systems

A diagram is a natural part of the modelling process and Newton's Laws form an axiomatic basis for analysis. This section concentrates on the way ambiguities can arise when constructing a diagram. The detailed discussion of the way the diagram drives the model, and how the O-O modelling process proceeds, is deferred until Chapter 7.

An ambiguity with a diagram arises with a spring-dashpot system **[AH1]**. Examples in (MST204 89) show that the two parts of Figure 6.8 are equivalent in the sense that they give rise to the same differential equation, $m\,x''[t] + r\,x'[t] + k\,x[t] = mg + ka$, where a is the unstretched length of the spring and the other symbols have their usual meanings.

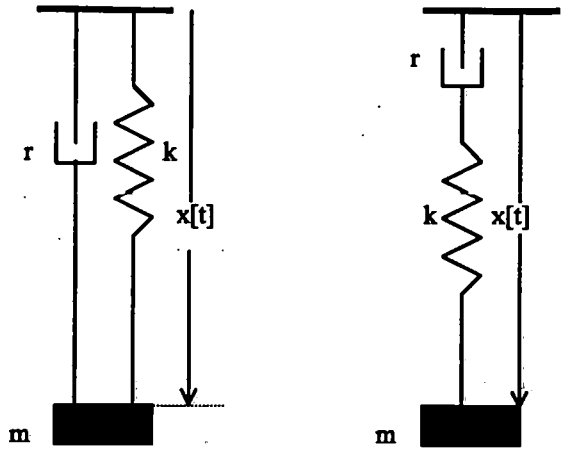


Figure 6.8

In addition to a *Particle* and a *Spring* object, there is a *Dashpot* object. The symbol *r* is new to the diagram (with its usual meaning), and this is the principal attribute **[DA5, DA3]** of the *Dashpot* object. The other is the coordinate x(t), although the precise nature of its involvement is not shown on the diagram. Modelling based on the left hand diagram in Figure 6.8 then proceeds by creating the relevant objects, linking the spring with the mass to produce a resistive force, and then linking the dashpot with the mass to produce another resistive force. An EquationOfMotion object results from linking the mass with the sum of forces concerned **[DM5,DM2]**.

```
Construct  Particle(P, m, x[t], mg)
Construct  Spring(S, k, a, x[t])
Construct  Dashpot(D, r, x[t])
LinkObjects(P, S)        → Force(T2, -k(x[t] - a) )
LinkObjects(P, D)        → Force(T2, -r x'[t] )
LinkObjects(P, T1+T2) → EquationOfMotion(EoM,
                              m x''[t] = mg - rx[t] - k(x[t] - a)  )
```

Using the right hand spring-dashpot diagram in Figure 6.8, it appears that either the spring and dashpot, or the particle and spring should be linked. The result of the first link would be a new *SpringDashpot* object which is not nonsensical but does not actually exist. The result of linking the particle with the spring will be a force, but will not change the fact that the dashpot and particle are not adjacent on the diagram. This would violate the Principle of Adjacency **[PA]**. Hence, neither of these alternatives is

preferable to the situation shown in the left hand part of Figure 6.8. Care must therefore be taken to draw even simple diagrams in an appropriate manner.

In section 7.10 of Chapter 7 I apply this simple spring-dashpot model to the context of a car suspension system. Townend (Townend 95) models an interesting variation in which the car suspension system consists of several coupled spring-dashpot pairs. The car is modelled by two particles, representing the front and rear axles. His domain knowledge results in a relatively sophisticated model which would be difficult to emulate with my O-O techniques. Linking pairs of objects would not be a problem, but there is no provision for combining a collection of objects to form a self-contained unit. If this were possible, such a unit could then be used like any other object in the problem domain. This is the essence of Townend's model: a 'unit' consists of a spring, a dashpot and a particle.

6.8.2.4 A Geometric Model which has no Axiomatic Basis

This section analyses a type B model: a diagram is essential because of the geometric nature of the problem. It is the 'firebreak' problem (MST204 92), which is discussed in detail in Chapter 9. The problem can be paraphrased as follows.

"In order to prevent excessive loss of timber in an area of forest, firebreaks are inserted such that in the event of a fire, the fire cannot spread too far. The task is to develop a model which predicts the optimal number of firebreaks such that the maximum number of trees can be harvested."

This problem has a number of difficulties in its formulation.

- There a no well-defined objective function to maximise. Part of the modelling process is to find one.
- Objects must be built from scratch, which is inefficient if they cannot be reused.
- The general problem-solving strategy is not fixed, despite the geometric solution presented here.
- The model must be capable of representing many or an indeterminate number of class instances on a diagram, without actually drawing them.

The geometric solution proposed is to consider a square area of forest, with a rectangular grid of firebreaks. It follows the treatment proposed in (Mitic 94). This simple geometry incorporates two opposing features **[AH1]**:

1. as the number of firebreaks increases, the forested area decreases;
2. as the number of firebreaks increases, the potential loss due to fire decreases.

Drawing a diagram to represent the situation is a good starting point for this type of model whatever strategy is adopted. The drawing consists of rectangles, so the first stage is to define an *Area* object, which will be used to represent forested and unforested areas **[DH1, DH5, DC2, DC1, DA4,DM3]**.

```
Area{
     attributes:
      Unique_ID;
      Forested/NonForested;  // boolean
      Area;
      NumberInstances; // the multiplicity of instances
     methods:
      GetArea();
    }
```

The model proceeds by constructing a forested area and firebreaks with the dimensions and multiplicity as shown in Figure 6.9(a -c).

Construct
Area(FOREST, Forested, s^2, 1)

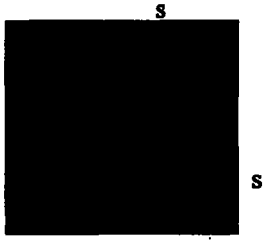


Figure 6.9a

Construct
Area(FIREBREAK1,
NotForested, ws, x)

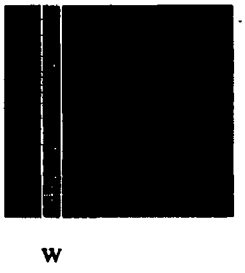


Figure 6.9b

Construct
Area(FIREBREAK2,
NotForested, A2, x(x+1))

$$A2 = \frac{(s - xw)w}{x+1}$$

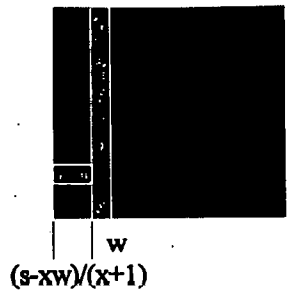


Figure 6.9c

This formulation constructs the FIREBREAK2 object such that it does not overlap a FIREBREAK1 object **[PA]**. It is algebraically simpler to construct a horizontal version of the FIREBREAK1 object and account for overlaps (i.e. don't count them twice), but this aspect is much harder to incorporate into an object model.

Linking objects modifies the 'area' of the forested region. Since it then no longer corresponds to the actual area of the object, an alternative strategy is to use these links to define a new object which defines a 'complement' area.

LinkObjects(FOREST, FIREBREAK1) → Area(FOREST, Forested, s^2 - wsx, 1)
LinkObjects(FOREST, FIREBREAK2) → Area(FOREST, Forested,
s^2 - wsx - (s-wx)wx, 1),

in which s^2 - wsx - (s-wx)wx = (s - wx)^2.

At this stage an objective function has to be defined. This has partially been done by computing the area of the remaining forested region. To complete this process, an additional method for the *Area* object can be added **[AH4,DM2,DM3]**, and used in the FOREST instance. This represents the result of losing N entire sub-regions (i.e. bounded by firebreaks and/or the edge of FOREST). This is followed by calling one of a variety of Solve methods, depending on what solution strategy is to be adopted.

FOREST.DefineObjectiveFunction1(N) → $A(x) = (s - wx)^2 - N\left(\frac{s - wx}{x+1}\right)^2$

FOREST.MaximiseAreaCalculus() → Solve A'(x) = 0,
rounding to the integer which gives a
maximal A(x).

OR

FOREST.MaximiseAreaDiscreteSearch(n) → Implementation of the algorithm:

```
{ maxArea = 0;
  For x = 1 to n do
  Calculate A(x)
  if A(x) > maxArea then
     maxArea = A(x)
  end_if;
  return maxArea
}
```

This completes the problem as posed in (Mitic 94). Overall, the O-O treatment is harder to formulate than a 'by hand' treatment, mainly because the modelling procedures are less clear then for a scenario with an axiomatic basis. The O-O approach has several difficulties.

- The nature of the problem formulation (geometric or otherwise), the method of solution and the construction of the objective function are all heuristics. The modelling methodology provides no guidance about any of these steps.
- Since there is no axiomatic basis behind the model, an object model has to be constructed from scratch. This is a large programming and/or modelling overhead.
- The problem of representing many object instances by drawing one class instance on the diagram is unsatisfactory from the viewpoint that 'every element drawn represents an object instance', which is central to the *Diagram Strand* of this methodology.
- It was proposed earlier that there is little likelihood of reuse of elements of this O-O model. This remains so, although descendents of the *Area* object, with additional methods, may be useful in some other geometric models **[DH2, DH3, DH4]**.

The advantages of the O-O approach are that it focuses attention on the following.

- How linking objects (i.e. constructing the diagram) assists in developing the model. It asks questions of the form "What is the effect of linking object A with object B?".
- Formulation of an objective function (also prompted by the word 'maximise' in the problem statement).

Developing the third of the above disadvantages, another possibility for implementing multiple instances of an object with one actual instance is to define a *MultipleInstance* object **[DH5,DM3]**. When linked with an instance of any other object, a *MultipleInstance* object would create as many instances of the target object as required.

In the firebreak problem, the *MultipleInstance* objects would then be linked with FOREST. Having constructed one FIREBREAK1 object, as above, Figure 6.9d could be drawn.

Construct
MultipleInstance(ID, FIREBREAK1, x)

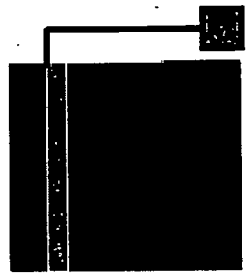


Figure 6.9d

LinkObjects(FOREST, ID) → Area(FIREBREAK1, NotForested, xws, 1)

This solution seems preferable since it is more consistent with the *Diagram Strand* principles. Overall, the construction of an object model for an ad hoc scenario such as the *Firebreak* problem is much harder than dealing with an axiomatic system. The methodology does not work as well. However, this is often the problem faced in large software projects: the scenario is unique and needs much analysis, most of which is domain specific.

6.8.2.5 Cash flow modelling

This is an example of Model Type C, and the Diagram Strand serves to clarify the role of payments at given times. (Wilson 89 and Bender 89) discuss problems of this type and deal with a sequence of payments made at given times. Wilson's basic problem is:

"An article may be purchased by a cash payment P at the end of Month 1, or by a lease-purchase scheme in which a payment Q ($< P$) is made at the end of Month 1, followed by 23 further payments of q ($<Q$) at the end of each successive month. The monthly interest rate is m%. Calculate the Internal Rate of Return (IRR) and the Annual Percentage Rate (APR) for the lease-purchase scheme."

One way of tackling this type of problem is to use the principle of *discounting* cash flows. If an amount x has to be paid in n months time, an amount $x_n = x(1+m/100)^{-n}$ must be held now **[AH1, AL2]**, because this amount, earning compound interest at m% per month for n months, will accumulate the required amount x in n months time. The quantity x_n is the *Net Present Value* (NPV) of x, and m is known as the *Internal Rate of Return* (IRR).

A time line is a natural starting point for a diagram **[DC1, DC2]**: it provides a *TimeLine* class (Figure 6.10a). Adding cashflows produces Figure 6.10b, in which $a = 1+m/100$.

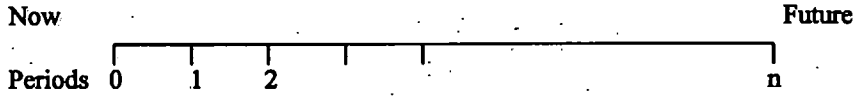


Figure 6.10a

```
Class TimeLine
{
 Attributes:
     n  Integer;  [AH3]
 Methods:
     GetTime();   // returns n   [AH4]
     SetTime(n)   // Constructor: defines n   [AH4]
}
```

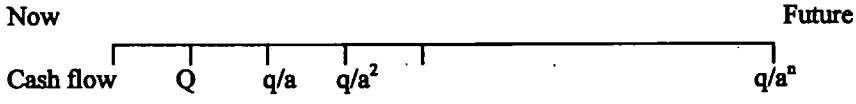


Figure 6.10b

Adding cash flows produces a class *CashFlow* **[DC1]**:

```
Class CashFlow
{
 Attributes:
     ID String;    // Unique identity
     a  Float;     // or m, the monthly interest rate
     Q Float;      // an initial payment
     q  Float;     // subsequent payments
     n  Integer;   // number of payments:
                   // total or excluding the initial payment
 Methods:
     NPV(r,q, a); // returns NPV of q for period r: q/a^r
}
```

This simple approach allows for a definition of NPVs, but the *Axiom Strand* is needed to needed to combine them.

The 'axiom' here is the rule: ΣNPV_i = Current Value (RHS to be clarified later) **[AL1, AL2]**. Wilson's problem (stated at the beginning of this section) is inaccurate: he has miscounted the number of payments. It is a matter for speculation whether or not a simple time line diagram would have helped in his analysis of this problem. Bender's discussion does not clarify the role of the NPV and the analyses of Bender and Wilson both lack applications in which discounted cash flows are used. One is discussed at the end of this section.

The LinkObects approach is awkward here because a *TimeLine* object (TL) must be linked with many NPV method calls of a *CashFlow* object (CF), in a construct like:

```
LinkObects(TL, CF.NPV(1), {CF.NPV(i), {i, 2, n}} )
```

(CF.NPV(1) is the NPV for Q at the end of month 1: separated from the other CF.NPVs because of its slightly different nature.)
The result of this link is the quantity ΣNPV_i **[AL6]**.

This approach is fundamental in that there is a formal separation of the *TimeLine* and *CashFlow* objects. The awkwardness of linking many objects, as above, can be avoided by extending the *TimeLine* object to subsume the *CashFlow* objects and provide the necessary summation by calling an appropriate method. This necessitates amending the existing definition of *TimeLine* (which is not really an appropriate name now: DiscountedCashFlow is better) **[DH2, DH3]**.

```
Class DiscountedCashFlow
{
 Attributes:
     ID String; // Unique identity
     n  Integer;// number of payments
     m  Float;  // the monthly simple interest rate
     M  Float;  // amount financed
     q  Float;  // subsequent payments

 Methods:
     GetTime();  // returns n
     SetTime(n); // Constructor: defines n
     CalculateTotalPayable(n,M,m);
                 // returns q = M(1+nm/1200)/n
     DiscountedCashFlowEquation(n,M,m,a);
                 // returns  dcf(n,M,m,a) = Σ_{i=1}^{n} q/a^i - M
     InternalRateOfReturn(n,M,m,a);
                 // returns Solve(dcf,a)
     APR(irr);   // returns (1+irr/100)^12-1 where
                 // irr = InternalRateOfReturn(n,M,m,a)
}
```

Although this *DiscountedCashFlow* object is less fundamental because it is a combination of more primitive objects, it is easier to use, and contains the necessary methods to compute a lease payment schedule, in the following way **[AH1, AL2]**.

Given a loan M for n months with annual simple interest rate m%,

1. Calculate the total payable using simple interest: $M(1+nm/1200)$
2. Compute payments $q = M(1+mn/1200)/n$.
3. Calculate ΣNPV, setting $dcf(n,M,n,a) = 0$, using an IRR r where $a=1+r/100$ %
4. Solve for r
5. Calculate the APR using $APR = (1+r/100)^{12}-1$

In O-O terms,

Construct DiscountedCashFlow(DCF, n, m, M) **[DC1, DC3]**

DCF.CalculateTotalPayable(n,M,a) → q

DCF. DiscountedCashFlowEquation(n,M,m,a) → dcf[a] **[AL6]**

DCF.InternalRateOfReturn(n,M,m,a) → a and IRR = 100(a-1)

DCF.APR(IRR) → APR = $(1+IRR/100)^{12}-1$

An alternative way of approaching problems involving capital expenditure and running costs is to compute the interest that would have been earned by investing the capital employed and adding this to the running cost. This approach can lead to over-simplified models because they typically consider one year's expenditure for running costs and no alternative to capital expenditure (e.g. leasing). This is the method used in (MST204 91).

6.8.2.6 Population dynamics

Population dynamics models (as in, for example, Berry 95) provide a mainstream modelling scenario where it would be highly unusual to draw a diagram. This is therefore an extreme example of a Type D model. It is made more extreme by the non-axiomatic basis for birth and death processes in population dynamics models. There are established techniques for formulating these problems, and these can be cast into O-O terms. The operational heuristics of the model are **[AH1]**:

- a variable *P(t)* (initialised) to represent the population at time *t*;
- Birth and Death functions;
- use of an Input-Output Principle to formulate either a difference or a differential equation for *P(t)* **[AL1]**;
- the discrete variable *P* is often approximated by a continuous variable (mostly without justification).

These heuristics identify some candidate classes: *Birth*, *Death* and *PopulationEquation*. It is harder to identify a 'control' class to encapsulate *P(t)*, but in the following analysis, it emerges that this is a useful class to have because both Birth and Death objects need to relate to it. Thus, consider a *PopulationState* class with attributes (Unique_ID, P(t), t, P_0) **[AL3]**. The Birth and Death classes are wrappers for the functional forms of the birth and death processes, and are therefore somewhat contrived. They contain functions $f_B(t)$ and $f_D(t)$ which are used in a link with a *PopulationState* object. This link is used to derive an equation which describes a state transition of the system (i.e. relates a change in *P* to a corresponding change in *t*) **[AL1, AL5]**. A skeleton O-O model is then as follows.

Construct PopulationState(POP,P(t), t, P_0)

Construct Birth(B, $f_B(t)$, t)

Construct Death(B, $f_D(t)$, t)

LinkObjects(POP, B, D) → PopulationEquation(PE,
Replace[f_B by P in $B(f_B(t), t)$],
Replace[f_D by P in $D(f_D(t), t)$])

The PopulationEquation object can represent a standard difference equation $P(t+1) - P(t) = B(P(t),t) - D(P(t),t)$, which can be cast, by calling a suitable method, into the differential equation $\frac{dP}{dt} = B(P(t),t) - D(P(t),t)$. Care has to be taken to supply correct functional forms $f_B(t)$ and $f_D(t)$ as problems are often phrased in terms of 'proportionate birth and death rates' **[DH1, DH4]**. In this case the right hand side has to contain appropriate forms corresponding to $\frac{1}{P}\frac{dP}{dt}$, or $\frac{P_{t+1} - P_t}{P_t}$ must appear on the left hand side.

Exponential growth (or decay) is given by:

Construct Birth(B, af_B, t)

Construct Death(B, bf_D, t),

and this gives rise to the difference equation $P(t+1) - P(t) = aP(t) - bP(t)$.

The case of logistic growth is given by:

Construct Birth(B, $kP_{max}f_B$, t) (where P_{max} is a constant)

Construct Death(B, kf_D^2, t),

and this gives rise to the difference equation $P(t+1) - P(t) = kP(t)\left(P_{max} - P(t)\right)$.

The conclusion from this example is that it is difficult to cast this problem into O-O terms such that the O-O analysis adds significantly to the problem analysis. The *Diagram Strand* has little relevance, and it is hard to conceive of any diagram which is not contrived. The method serves to draws attention to the components in the problem domain, the way in which they interact, and the nature of the resulting difference and differential equations.

6.8.2.7 Input-Output processes

Input-output systems are similar to population dynamics systems (and are of Type D), but pose difficulties for the *Diagram-Axiom* methodology. The physical situations described are often relatively straightforward, but can be difficult to represent in a 2-D diagram, require a way of representing infinitesimal lengths, and pose difficulties in capturing *events per unit time* on a diagram. It is also essential to be able to concentrate on actual inputs and outputs to the system.

To illustrate diagrammatic difficulties, consider how to represent flow through a tube with circular cross-section, and incorporate diffusion through the wall of the tube from outside to inside. This scenario is the prototype of many similar ones, and is described in (Herod 98) in the context of diffusion of a solute through a blood vessel. Figure 6.11a shows a transverse cross-section.

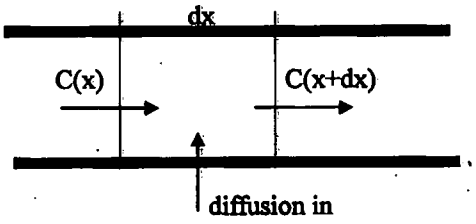


Figure 6.11a

Some key elements are missing in this diagram: the perpendicular (circular) cross-section and the area of the annulus of width *dx* (which is a rectangle when opened out). Elements of the *Axiom Strand* are needed to describe the surfaces concerned, with the aim of computing the number of units ('solute particles') passing through each surface in unit time **[AL1, AL2]**. Suppose that the speed of fluid through the tube is *u* and that solute diffuses into the tube at a rate proportional to the difference *C(x) - S*, where *C(x)* is the concentration of solute inside the tube and *S* is the constant concentration of solute outside. The object model has a *Flow* class, with attributes directly relevant to computing the number of units passing through a surface in a diagram in unit time **[AH3, DA2]**.

The prototype is:

Flow(Unique_ID, IN-OUT, Concentration(Position),
Surface_Area, Flow_Parameter)

The model proceeds as in Figures 6.11(b-d).

Construct Flow(Cin, IN,
C(x), πa^2, u)

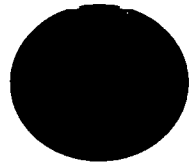


Figure 6.11b

Construct Flow(Cout, OUT,
C(x+dx), πa^2, u)

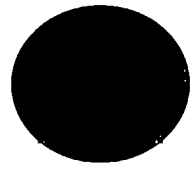


Figure 6.11c

Construct Flow(DIFF, IN,
S-C(x), $2\pi a$ dx, k)

Figure 6.11d

The disadvantage of this approach is that the surfaces drawn are not shown in context. Linking the two input flows to produce an overall 'total input flow' does not work well because artificial quantities have to be supplied for the Surface_Area and Flow_Parameter parameters. Instead, linking all inputs and outputs can result directly in an equation of state, the *FlowEquation* object below, in which left and right hand sides record the output and input flows respectively **[DM3]**. Calling a DifferenceEquation method then produces a 'balance' difference equation.

LinkObjects(Cin, Cout, DIFF) → FlowEquation(Feq, Lhs, Rhs)

Lhs = $C(x+dx)\,\pi\,a^2\,u$

Rhs = $C(x)\,\pi\,a^2\,u + (S - C(x))\,2\pi\,a\,k\,dx$

Feq.DifferenceEquation → $C(x+dx)\,\pi\,a^2\,u =$

$C(x)\,\pi\,a^2\,u + (S - C(x))\,2\pi\,a\,k\,dx$

Feq.DifferentialEquation → $\frac{dC(x)}{dx} = \frac{2k}{au}(S - C(x))$

The last step, in which a formal limiting process results in a differential equation (possibly a differential equation object), may require some tricky pattern matching in a detailed implementation in order to automate detection of the terms *C(x+dx)* and *C(x)*, and form their difference.

6.8.2.8 A model based solely on heuristics

In this section I consider a modelling scenario which has no axiomatic basis, and for which a diagram is not useful. Since it is a heuristics model, it is of Type E. This scenario is taken from (Townend 95), and I include it here because it appears that the *Diagram-Axiom* methodology will be of little use. Since the scenario is unusual, I quote the problem in full.

The goal of many students is to maximise the amount of knowledge gained from a course (or at least gain enough to pass the examination!) while at the same time minimising the intellectual effort expended in achieving it. Your own experience will tell you that different study strategies exist. Develop a mathematical model which could be used to establish a work schedule which would permit a student to achieve the above goal.

This problem presents difficulties for modellers because the features are unclear. Townend's analysis clearly indicates that domain knowledge is needed, and his model contains ill-defined features which are not explicitly in his features list. The problem for the O-O modeller is worse: a diagram is no help because the scenario has no physical

objects, and classes must be elucidated. Using the 'heuristics' part of the *Diagram-Axiom* methodology (section 6.4), the following features are potential classes or attributes.

- Amount of knowledge **[AH2]** (a potential class, rephrasing the problem in the active tense).
- Intellectual effort **[AH3]** (a potential attribute since this is the object in a sentence, although this feature seems more likely to be a candidate class).
- Study strategy. **[AH2]**
- Work schedule **[AH2]** (likely to be the same as 'study strategy').
- Minimise the intellectual effort **[AH1]** (part of the goal).
- Maximise the knowledge gained **[AH1]** (part of the goal).

The problem statement refers to "your own experience", which is ill-defined **[AH5]**. Experience is necessary to derive Townend's other features: *course duration* and *rate at which students forget knowledge*. The *Diagram-Axiom* methodology is only partially successful at finding features (in order to find classes and attributes), and fails to find methods (I did not find [AH4] useful - identify methods from verbs). Townend derives a differential equation containing the following components:

- $K(t)$ - Knowledge at time t;
- $E(t)$ - Effort at time t;
- m - Fraction of knowledge forgotten;
- T - Total study time;
- a, b, c - constants:
- Rate of Knowledge acquisition: cK^aE^b
- Rate of Knowledge loss: mK
- Differential equation for the model: $\frac{dK}{dt} = cK^aE^b - mK$

  (essentially an input/output situation)
- Goal - minimise $\int_0^T E(t)dt$

The terms cK^aE^b and mK must be formed in an O-O analysis by linking the classes *Knowledge* and *Effort*. The functional form of these terms is a further heuristic [AH5]. The precise result of such a link is uncertain. Similarly, the precise nature of the object relevant to the difference $cK^aE^b - mK$ is also uncertain. Overall, the Diagram-Axiom methodology imposes additional burdens on the user.

6.9 Discussion

In this section I discuss the strengths and weaknesses of the Diagram-Axiom methodology.

6.9.1 Strengths

1. The Diagram-Axiom methodology satisfies the principal requirements for mathematical modelling support, which are:
   a) to drive the modelling process;
   b) to identify classes and attributes with reasonable ease;
   c) to find links between classes, encapsulating them as methods or stand-alone operations;
   d) a simple system which does not need to support a complicated structure.

2. In providing guidelines for finding classes, the Diagram-Axiom methodology helps to clarify aspects of the system's behaviour, which is useful for finding features in a *generic* modelling analysis. An example is the *Study* problem (section 6.8.2.8). A noun-verb analysis of the problem statement helps to find classes and attributes (i.e. features). Having found features, the Diagram-Axiom methodology helps to define how they interact by finding what the methods for the classes are. This defines relationships in *generic* modelling.

3. The method of modelling introduced in section 6.5 is sufficiently flexible to solve a variety of models. I chose contexts which ranged from those where it is normal practice to draw a diagram, to those where a diagram would not normally be useful. These cases test the *Diagram* and the *Axiom* strands respectively. The methodology copes best when there is a well-defined diagram (for example, the simple Particle Mechanics problem in section 6.8.1) because elements on the diagrams help to define classes. It is also easy to find attributes because they correspond to properties of classes (for example *Mass* and *Position* for the class *Particle*).

4. The linking process generates models in an ordered way by allowing only certain objects to be linked at an appropriate stage. This reduces the risk of error by preventing inappropriate links (for example a *Spring* with a *Gravitational Field*). When applied to problems where drawing a diagram is not normal practice the *Axiom* Strand of the methodology also succeeds in generating a model (for example the *Cash Flow* model in section 6.8.2.5). Sufficient parts of this strand (section 6.4) exist to identify and define classes. In some cases, a suitable diagram can be drawn (the *Firebreak* problem of section 6.8.2.4), whereas in others a "generic" diagram (where icons represent conceptual objects, as in the *Input-Output* problem of section 6.8.2.7) can also be drawn. Linking objects with the aid of a diagram (which invokes the *Principle of Adjacency* of section 6.4) then solves the problem of relation generation (summarised in section 4.1 of Chapter 4).

5. The *Heat Transfer* problems in sections 6.8.2.1 and 6.8.2.2 show how if only a few classes are defined, they can be used as basic units in building a variety of models. While coding the *Heat Transfer* classes (Appendix 7H), I succeeding in formulating and solving problems involving differing numbers of surfaces, differing numbers of layers, with and without a heat source. The coded examples in Chapter 7 also illustrate this versatility in the context of Particle Mechanics.

6. It is possible to represent elements of a diagram in which are not actually drawn by defining object arrays (as in the *Firebreak* problem, section 6.8.2.4). This is an extremely versatile device because it represents a general case in which the presence of objects can be implied without those objects actually being present. This enables a large number of objects, or a number of objects expressed in terms of a symbolic parameter to be modelled. In the case of the *Firebreak* problem, the number of firebreaks in unit length can be expressed in terms of a symbolic parameter n.

6.9.2 Weaknesses

Only the first of the following four points is a serious shortcoming of the Diagram-Axiom methodology. The others can be addressed with further work.

Heuristics in the Axiom Strand

It is not easy to use the heuristic parts of the *Axiom Strand*, as the *Study* problem (section 6.8.2.8) shows. The tasks that must be done to find features in *generic* modelling and to find classes in O-O modelling are similar. In the *Study* problem, noun-verb analysis of the problem statement detected classes and attributes, and the same technique could have been used to find features. This was my approach in a *generic* modelling treatment of the knowledge problem. The basis for the relationship between classes and features is the association *Feature = Distinct Class*. Similarly, the association *Property of Feature = Attribute* serves to detect attributes. To find methods, the ways in which features (i.e. classes) interact must be defined. The same task applies for *generic* modelling. In O-O modelling there is the additional tasks of expressing everything in O-O terms. Hence, to create an O-O environment, more work has to be done than in a traditional analysis of the same problem. However, once a class has been defined it can be used in further analysis without repeating work unnecessarily.

The need for Aggregate Classes

There is currently no mechanism for grouping classes to form an aggregate class. An aggregate class can be useful in two ways. First an instance of such a class may be used in place of its constituent objects. This would make it easier to build a diagram on the screen and to approach modelling in a modular way. An example is to combine the *Spring* and *Dashpot* classes to make a new class (*SpringDashpot*) which would have the combined attributes and methods of its constituent objects. The model of Section 6.8.2.3 would then proceed as follows (with Figure 6.12 replacing Figure 6.8).

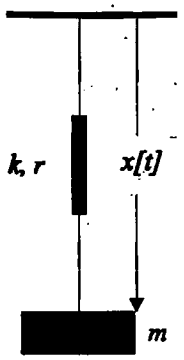


Figure 6.12

Construct Particle(P, m, x[t], mg)
Construct SpringDashpot(SD, k, r, a, x[t])
LinkObjects(P, SD) → Force(T, -k(x[t] - a) -r x'[t])
LinkObjects(P, T) → EquationOfMotion(EoM, m x''[t] = mg - rx[t] - k(x[t] - a))

The second way of using aggregate classes is to provide a means of creating objects which do not correspond directly to physical objects. An example is the *Study* problem (section 6.8.2.8) in which instances of classes *Effort* and *Knowledge* have to be linked to form a new object which represents the term cK^aE^b the in the differential equation for the model. This technique avoids the problem of deciding what the result of a non-obvious combination of classes should be: it is a class with the combined methods and attributes of its constituent classes.

Controls over how the Diagram is drawn

The diagram has to be drawn in a sensible order. If this is not done, appropriate links cannot be created. Some order is mandatory when drawing a diagram, so it is possible to find a reasonable order. However, this order is not controlled by the software, which makes the process error-prone. In practice, a useful way to order the diagram is to ask "what is the effect of linking..." when an additional object is added to the diagram, and to do this for each object already on the diagram. This process also determines links.

Ordering steps in the Diagram-Axiom methodology

The steps in the *Diagram-Axiom* methodology are not intended to be followed in any particular order. In order to produce the models in this chapter, I used whichever steps were appropriate. In principle, they need to be ordered and refined in order to be more generally accessible, and to stress the parts which prove to be most useful in practice.

Chapter 7

AMK: Object-Oriented Software for Particle Mechanics

7.0 Abstract

This chapter presents a computer algebra software implementation of the *Construct→Link→Invoke_Methods* modelling cycle, applied to the context of Newtonian Particle Mechanics. This context is widely studied, is non-trivial and forms the basis of more advanced techniques. It is therefore a starting point for developing more general techniques. There is a description of the Mathematica O-O environment, and the class hierarchy, derived using the *Diagram-Axiom* methodology. The treatment of space/time coordinates and the role of gravity are given as examples of problems which cannot easily be resolved by a methodology alone. These require domain knowledge, and I suggest alternative strategies. The method of linking objects is discussed in detail. This is significant for modelling because it modifies the generic modelling cycle by constructing and linking objects until an *EquationOfMotion* object results. Its *Equation* method can then be called to produce an explicit equation of motion, which is the goal of the cycle. Several further discussion points also arise. These include alternative ways to link objects, different class hierarchies and the role of multiple inheritance. The modelling examples that follow illustrate the power and scope of the methodology and software. They show two things. First, how important geometry is in problem formulation, and how this relates to the *Diagram-Axiom* methodology. Second, how problems of increasing complexity can be solved with only minor amendments to inputs. I indicate the limitations of the method in this context and also highlight problems with the syntax of input expressions, and when following the *Construct→Link→Invoke_Methods* cycle. In order to demonstrate the wider applicability of the software, I show how new classes can be created for the Particle Mechanics context, and develop and use a class hierarchy for a different context.

7.1 Overview and Purpose of Software

This chapter describes a concrete implementation of object-oriented modelling techniques, based on Newtonian particle mechanics. The implementation has three purposes. First, it demonstrates that a working software implementation is possible. Second, it illustrates that the system is capable of formulating a wide range of problems, despite a relatively simple rule-based basis. Third, it shows that the system can be easily extended and amended.

The minimum software required is a CAS in which an object-oriented environment can be built. When a prototype was first implemented (1991), Mathematica was the natural choice for the computer algebra engine because of its extensive programming capabilities and communication protocols. Mathematica is still the best choice for the computer algebra component for the same reasons that it was originally chosen. The Mathematica implementation of object-oriented Newtonian Particle Mechanics is called the *Applied Mathematics Kit* (AMK). An overview of it appeared in (Mitic 95A).

7.2 The Newtonian Particle Mechanics Context

Newtonian Particle Mechanics provides an environment which is based upon well-defined axioms, in which it is relatively easy to isolate candidate objects. The axioms of Newtonian mechanics provide a scientific model for mechanics, and have no parallel elsewhere. It is also standard practice to draw diagrams when solving problems, so that the *Diagram-Axiom* methodology can be exploited easily. Relating features is a problem that applies to particle mechanics, despite the axiomatic problem domain. A strategy for problem formulation is still not always clear to students, and Chapter 9 provides evidence of this. The software introduced in this chapter is limited to 1-D and 2-D Newtonian Particle Mechanics. It need not be restricted to this domain, and Appendix 7H shows how it works for a heat transfer problem.

Existing undergraduate courses on mechanics provide many examples of how to solve problems which involve particles, forces, strings, springs and other similar elements. These elements are mentioned implicitly, and problem-solving is often done by example. This approach has disadvantages. Students can have difficulty in determining a starting point for finding a solution, in isolating the necessary steps which are needed

to obtain a solution, and in applying the correct techniques and processes at the appropriate time. Generic computer algebra systems have been used to solve problems in particular contexts by providing a specific environment for problem solving. Dubisch (Dubisch 90) constructed an environment for solving problems in Newtonian Mechanics using Mathematica. This approach relies on a toolkit of templates, which works well if a template is available. If not, one must be created. This can lead to a large number of specific cases. Viklund and Fritzson (Viklund 92) created a formal O-O environment in order to do finite element computations. They supplement Mathematica by ObjectMath, with which objects can be defined and manipulated, but only used computer algebra for prototyping and coordinate transformations.

7.3 The Object environment

The O-O environment used is that developed by Maeder. Its principles are discussed in (Maeder 92 and Maeder 93). These papers discuss general features of implementing message passing and using methods in Mathematica. The implementation used here is from (Maeder 94A), but Gray (Gray 94) gives a clearer explanation of it. It is a full object-oriented environment, in which methods associated with object instances are stored using *up-values* (i.e. a rule for applying a method is attached to a symbol - the unique identifier for the object instance). The generic procedure *Class* defines classes, with methods and attributes. A constructor procedure, *new*, creates instance variables, and message passing is implemented as a rule for applying functions to objects. The destructor *delete* frees memory when objects are no longer needed. It is rarely necessary to do this, even though destroying objects would be more consistent with a revised modelling cycle, as described below. For example, once a *Particle* has been associated with a *Spring*, a *Force* (the tension in the spring) can replace the *Spring*, which is no longer needed. Meader's environment is presented in the form of a Mathematica package, *Classes.m*. In addition to implementing O-O features, this package:

- integrates the O-O features into an environment where symbolic and numerical manipulation are possible;
- permits communication with other Windows applications.

In principle it would have been easier to define and maintain a class hierarchy in C++, but the overriding consideration was a need to perform symbolic manipulations. This meant that an algebra engine was essential. Maple or Macsyma are potential alternatives

to Mathematica, but no object-oriented environment packages are available for them. It would also have been difficult to write one with the available primitives. A more complicated alternative is to maintain the class hierarchy in C++, holding all attributes as strings. These could then be passed to Mathematica using the MathLink protocol, converted to expressions, manipulated, and then returned to C++. This approach is likely to be much more efficient in terms of execution time but is much more complicated to implement.

7.4 Objects and the Object hierarchy

A search through standard texts on mechanics, such as (Milne 48) and (Dyke 92), reveals consistency in the objects (not in the O-O sense) that appear and in the situations in which they appear. Many objects in this context share the same characteristics. The most fundamental shared characteristic is a coordinate system. The class *CoordinateSystem* forms the superclass for the majority of the other classes in AMK. *CoordinateSystem* encapsulates a complete implementation of 2-D polar and cartesian coordinates. Details of *CoordinateSystem* attributes are given in (Mitic 95A). Appendix 7FS contains a functional specification of all the AMK classes and auxiliary procedures.

The class *CoordinateSystem* is itself a subclass of the abstract class *CoordinateTransformations*, which provides primitives for transformations between cartesian and polar coordinates in terms of rewrite rules. This is an efficient implementation for the class hierarchy because transforming between coordinate systems is a common operation and is applicable to any object that uses coordinates. An alternative would have been to program methods for transforming coordinates for each coordinate system. I tried and rejected this less efficient option because of the programming overhead (having to repeat parts of the code). There was a clear role for creating an abstract superclass.

Most other classes are subclasses of *CoordinateSystem*, and have no subclasses themselves. For example, the class *Particle* has the extra attributes *Mass* and *Time*, and has, among other methods, *Velocity* and *Acceleration*. The latter two are obtained by differentiating the space coordinates with respect to time. Such mathematical processes make it an advantage to maintain the object hierarchy within the Mathematica environment. An example of a subclass of *CoordinateSystem* which has its own subclass

is *HorizontalPlane*, which has two subclasses, *InclinedPlane* and *CircularSurface*. Each has one additional attribute: *Slope* and *Radius* respectively. This implementation makes modelling motion on a circular surface particularly easy, despite the difference in physical objects. It would have been possible to implement an abstract class *Plane*, with concrete subclasses *HorizontalPlane*, *InclinedPlane* and *CircularSurface*. The classes *HorizontalPlane* and *Plane* would have been so similar that it was not worthwhile making the change, despite being neater in O-O and conceptual terms. The class *EquationOfMotion* is not a subclass of *CoordinateSystem*. It is the only class that does not need to inherit any methods of *CoordinateSystem*. The class *CoordinateSystem* has an attribute *Info*, which provides a generic way of inserting information into the system without the need to define a specific method. It is used mainly to provide information and symbol names for later use, and to supply initial conditions.

Figure 7.1 shows the class hierarchy, and comments on it follow.

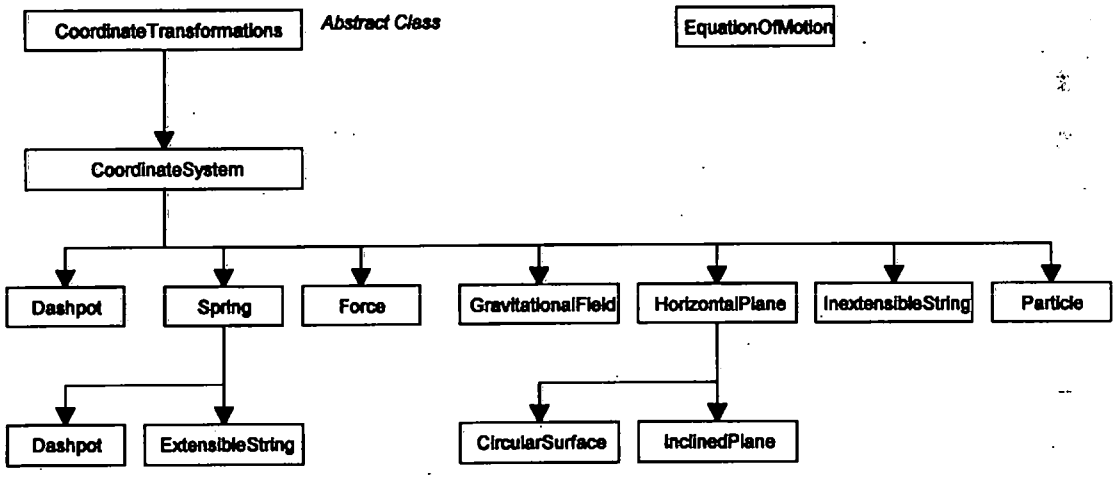


Figure 7.1

7.4.1 Dashpot

The *Dashpot* class is shown in two places in the hierarchy. This reflects two methods of implementation, the secondary one shown with the dotted link. This class is discussed in detail later.

7.4.2 Gravity

Several implementations are possible. One possibility is to associate a symbol g, and a nominal "down" direction with the *Particle* class. The consequence is that *Particle* becomes a vector, which is conceptually unsatisfactory. It is convenient to be able to derive the particle's weight by calling a method, but problems are often formulated in the absence of gravity. The nominal direction "down" should also relate to the coordinate system for the problem domain. A better alternative was to create a distinct *GravitationalField* class, which has two attributes: a magnitude g and a direction, which is inherited from *CoordinateSystem*.

7.4.3 Extensible String

Programming the *ExtensibleString* class proved to be very awkward because, when associated with a *Particle* object, it behaves like an extensible spring when stretched, and a null object when slack. Its tension T when its length is x is $T(x) = \begin{cases} k(x-a); x \geq a \\ 0; \quad x < a \end{cases}$, where the unstretched length is a and the stiffness is k. These conditions can only be fulfilled after the model is formulated and solved. The only alternative is to write the tension as a function of time, which also requires a pre-solved model. Furthermore, whether or not the string is slack depends on initial conditions. In a simple case, if the initial displacement is greater than the equilibrium extension, the string will become slack. A multi-stage process must be used to solve extensible string problems "traditionally". An O-O analysis does not lend itself easily to multi-stage problems, and implementation problems persist. An alternative is to define a class which can switch between a *Particle* and an *ExtensibleSpring* under given conditions. The problem of when those conditions are fulfilled still remains.

7.4.4 Pulleys

No *SmoothPulley* class was implemented. A pulley (or peg) is a means of redirecting a force. These objects are not as common in problems as others in AMK and could be replaced completely by a pair of forces. A direct implementation is to define three sets of coordinates, which are the attributes of the *SmoothPulley* class. The first coordinate is the position of the pulley. The others allow the direction of a force and of a redirected force to be defined. These forces typically take the form of tensions in strings. It is less clear how a pulley object would operate in practice. Since its effect is to signify that the tensions in two parts of a string are equal, a LinkObjects function could be defined which formally equates the tensions in two *InextensibleString* objects:

```
LinkObjects(P, {IS1(T1), IS2(T2)}) → {IS1(T1), IS2(T1)}
```

This approach could then easily be used to define a descendent class *RoughPulley*. An additional method would be required which defines the relationship between the tensions in the two inextensible strings which are linked with the pulley.

7.4.5 Point Objects

No *Point* class currently exists in AMK. Instances of a *Point* class could be used, typically, to define fixed coordinates such as the point of suspension of a simple pendulum, or an initial position. The role of *Point* objects is subsumed into attributes of other classes that use coordinates, and I did not find them necessary. However, they may be useful if AMK is extended to include relative motion. In this case, the concept of a fixed point (or a fixed reference frame, which can be defined in terms of fixed points) is much more important.

7.5 Changes to the Generic Modelling Cycle: Creating and Linking Objects

In Chapter 1 I described the *Generic* modelling cycle. The way in which objects are created and linked in AMK modifies the *Formulate Model* and *Set Up Model* stages of this cycle (Figure 1.5 in Chapter 1). This section outlines the necessary changes.

When learning how to solve applied mathematics problems 'by example', a number of steps are implicit. First, which entities exist in the problem domain. Second, how to use the Newtonian axiomatic model to produce a mathematical model. Third, what the necessary steps to solve a problem are. AMK uses predefined objects which interact using their methods. This thesis uses the term 'linking' for this. AMK modelling is therefore done by constructing instances of objects and specifying which of them interact. The software handles the way in which they interact automatically. Linking constructs a new object or objects. Information about any given object can be obtained at any time by invoking a method of that object. The user has to know a general strategy for problem solving and has to be aware of the consequences of the interaction of two objects. The pseudocode below summarises this aspect of the modelling cycle.

```
For each  physical object in the system
   Define  an instance of that object
End_For
For each  Particle in the system
   Repeat
      Link Particle to appropriate other objects
   Until the Particle and a Force are linked
   Obtain the equation of motion from EquationOfMotion
End_For
```

This algorithm concentrates on the *Particle* object because particles are the massive entities which are an essential part of Newton's Laws. The term 'physical object' in the algorithm refers to any object other than the *CoordinateSystem*. The algorithm replaces the 'Formulate Model - State Relations' stage of the Penrose 7-point Modelling Cycle (see Chapter 1). Details of appropriate links appear in the next section.

7.6 Object Links

This section contains a discussion of two points. First, the relative merits of two distinct design alternatives for implementing object links. Second, I state the feasible links for objects in AMK.

Linking objects formally formulates and solves modelling problems. Two ways of doing this are apparent. The first is to define an appropriate method for each class, which takes another object as its argument. For example, in order to create a *Force* (the tension T) by linking an *ExtensibleString* S with a *Particle* P, the following method could be defined (detailed arguments are omitted):

```
P.LinkWith(S)   → Force(T)
```

This would mean defining a suitable method for each object, and having to redefine the same link in a construct such as `S.LinkWith(P)` → `Force(T)`. This approach is consistent with the O-O paradigm, but the duplication of methods proved to be awkward. An alternative approach is to define a global polymorphic procedure, *LinkObjects*, which takes the objects to be linked as its arguments. Thus, to link P with S, the procedure `LinkObjects(P, S)` is called. This approach also has the advantage that if no *LinkObjects* template exists for any given set of objects, no link between those objects is possible, and a null result is returned. The programmer is responsible for constructing suitable *LinkObjects* templates, and for defining what links are meaningful. There should be no links for objects for which the resulting combination is not meaningful, such as a *Spring* with an *EquationOfMotion*.

Only a limited number of links are meaningful, so the number of templates required is small. The ones implemented in AMK are:

Particle + GravitationalField = Force;

Particle + Spring = Force;

Particle + InextensibleString = Force;

Particle + HorizontalPlane = Force;

Particle + InclinedPlane = Force;

Particle + CircularSurface = Force;

Particle + Dashpot = Force;

Particle + Force = EquationOfMotion.

"Force + Force = Force" makes sense but is implemented in a different way to allow for overloading of the " + " symbol in expressions such as TotalForce = Force1 + Force2. A calculus for combining forces using the "+" operator means that there is no need for a LinkObjects procedure for two *Force* objects. Polymorphic definitions determine the *Sense* of the resulting force from the inputs. Forces implemented in this way make it possible to write 'natural' links such as `LinkObjects(P, W + T)`, in which a *Particle* P is linked with the sum of a weight W and a tension T. All the procedures for combining forces assume that the forces concerned are given with respect to the same coordinate axes. If this is not so they cannot be combined unless one coordinate system is rotated such that it coincides with the other. The example of the interaction of a particle with an inclined plane shows this, and is discussed in Section 7.9.

In order to test for the type of objects which are being linked, the environmental primitive *isa* implements boolean functions such as `ParticleQ(P)` (is *P* a Particle?). In some cases, linking two objects triggers a response which is particularly appropriate for the situation. For example, if an inextensible string is linked with a particle, the result is a force (the tension in the string). If the coordinates of the particle were given in terms of polar coordinates, the resulting force is also given in terms of polar coordinates. Otherwise, both are given in terms of cartesian coordinates. This trick anticipates the wishes of the user. For example, polar coordinates are often most useful for a simple pendulum system.

7.7 An alternative Class Hierarchy using Multiple inheritance

Even though the object-oriented environment does not support multiple inheritance, there are some advantages in using it, but there are more compelling reasons for using single inheritance only. Two examples of how multiple inheritance could be used follow.

The first is in implementing the kinematic and geometrical properties of particles. A *Coordinate* class can define space and time coordinates. The only attribute of the *Particle* class is then *Mass*, because it does not inherit coordinates attributes from the *Coordinate* class. Any kinetic properties of the particle must then be derived from both the *Coordinate* and the *Particle* classes. This is the purpose of a *MassAcceleration* class, which defines the product of mass and acceleration in Newton's 2nd Law. The result is a *Force* object, which has to have dimensional information and must therefore be descended from the *Coordinate* class. The *MassAcceleration* object can then be linked with the *Force* object to produce an *EquationOfMotion* object. This gives the class hierarchy in Figure 7.2.

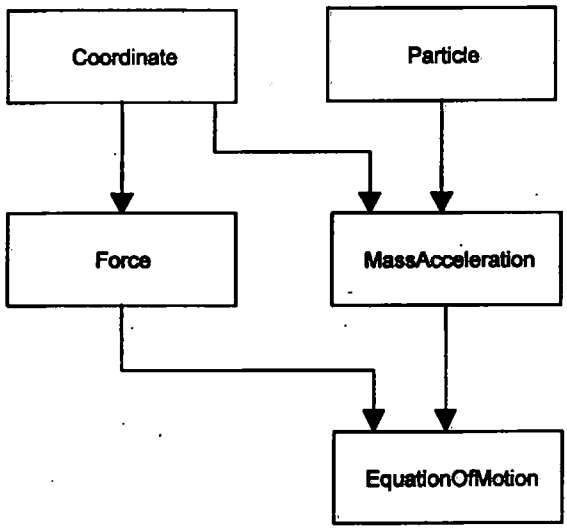


Figure 7.2

If this hierarchy is adopted, the acceleration component of the *EquationOfMotion* object can be inherited from the *Coordinate* object, but a copy of the relevant information is also held as an attribute for the *Force* object. This is an unnecessary complication.

A second example is fundamental to implementing coordinate systems. In principle, it would be useful to define a superclass *Coordinate* with two subclasses, *PolarCoordinate* and *CartesianCoordinate*. It would then be possible for a subclass such as *Particle* to inherit methods from both the classes *PolarCoordinate* and *CartesianCoordinate*. If multiple inheritance were available in this way it would simplify some of the code considerably. In particular, coordinate transformations would have been easy.

Given the theoretical difficulties in using multiple inheritance in some circumstances, and the practical difficulty of implementing multiple inheritance in the object-oriented environment, it was easier and more practical to stick to single inheritance. This approach drove the *CoordinateSystem* class, from which most other AMK classes descend.

7.8 Initial Conditions and Solving the Equation of Motion

The software introduced in this chapter is limited to deriving an equation of motion using Newton's Second Law. Solving the resulting equations of motion is relatively trivial using standard Mathematica primitives (in italics in this paragraph), with some minor complications. *Dsolve* must be used to solve differential equations, and *Nsolve* and *NDSolve* are their counterparts for finding numerical solutions. *FindRoots* is a particular function for finding numerical solutions to polynomial equations, but more generally, *Solve* can be used for systems of equations. Many dedicated numerical and symbolic techniques are available for given circumstances. In principle, all that is necessary is to choose a technique and call the relevant function. Solutions and objective functions are therefore not implemented.

In order to solve the equation of motion fully, initial conditions have to be supplied. Although the initial conditions are usually an essential part of the model, they are only needed to solve the equation of motion. Since the current implementation of AMK does not do this, most of the models developed in this chapter do not include initial conditions.

A mechanism already exists for setting initial conditions using the *Info* attribute. For example, initialising the velocity of a particle to u is possible as follows.

```
P = new[Particle, Cartesian[{{x[t]}, 0, 1}],
          {InitialVelocity -> u}, t, m]
```

However, there is no mechanism to require the user to supply initial conditions, and no mechanism for creating a default if the user does not specify them. The subsequent treatment of initial conditions is best done by propagating them to form an attribute of the *EquationOfMotion* class. As such, it is easy to integrate them into a Solve[] method of the *EquationOfMotion* class.

7.9 Modelling Examples

The following examples illustrate how AMK can solve modelling problems of varying complexity. I consider four problems areas: particle-force interactions, spring-particle systems, contact problems and the simple pendulum.

7.9.1 Particle-Force interactions

Particle-Force Problem 1

This simple problem illustrates the general principle of the AMK method: construct objects, link objects until an *EquationOfMotion* object is created, and call its *Equation* method. A particle P is subject to a 1-D force F1 of magnitude e^t (Figure 7.3). The InitialVelocity parameter u can be retrieved for later use. Extending to 2-D is trivial.

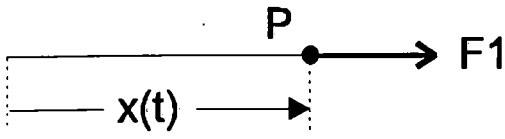


Figure 7.3

```
P = new[Particle, Cartesian[{{x[t]}, 0, 1}],
          {InitialVelocity -> u}, t, m]
F = new[Force, Cartesian[{{E^t}, 0, 1}]]
E1 = LinkObjects[P, F]
Equation[E1]
```

The result is

```
                t
{m x''[t] == E }
```

7.9.2 Particle-Spring systems

Particle-spring systems illustrate the versatility of AMK: only a few basic toolkit elements are needed to formulate and solve relatively complex problems. One key modelling point forms the basis of the *Spring* class: the spring only has a tension when something is attached to it. Theoretically, and independent of any link, a tension arises when the current length of the spring is not equal to its unstretched length. A Tension method (which returns the symbolic value of the tension) is coded but is not used. The tension is obtained from a *Particle-Spring* link.

The class *Spring* needs attributes to record the position of its two ends. They are determined relative to the coordinate system and are inherited from the parent class, *CoordinateSystem*. A spring has to have a nominal 'direction', which is the direction of a coordinate increasing. This ensures that the tension has the correct direction. There is no need to use the concept of a fixed and a non-fixed end for the spring since, in many situations, there is no fixed end. Hence there is no need for a *Point* class, as proposed in Section 7.4.5. The major advantage of this strategy is that the user does not need to determine the sense of the tension because the relevant LinkObjects method does it automatically, provided that users use a useful heuristic. This is to treat every spring as extended. This is not the only way but is often the simplest. It also works even if it is not physically possible for a spring to be extended.

7.9.2.1 *Spring Problem 1*

In this problem, a particle of mass m is suspended under gravity on the end of a spring of unstretched length L and stiffness k from a fixed point (Figure 7.4).

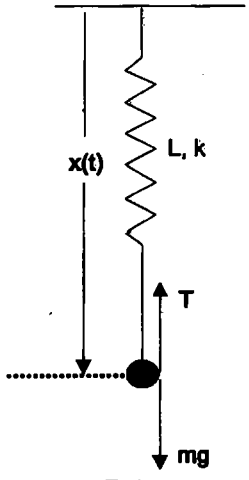


Figure 7.4

There are three objects: the particle P, the gravitational field GF and the spring S. Suitable links produce forces, and the sum of these forces, when linked with the particle, produces the equation of motion.

```
P=new[Particle,Cartesian[{{x[t]},0,1}],{},t,m]
S=new[Spring,Cartesian[{{0},0,1}],{},
        Cartesian[{{x[t]},0,1}],L,k]
GF=new[GravitationalField,Cartesian[{{g},0,1}],{}]
W=LinkObjects[P,GF]
T=LinkObjects[P,S]
```

```
ESP=LinkObjects[P,T+W]
Equation[ESP]
```

The result is:

```
{m x''[t] == k L + g m - k x[t]}
```

7.9.2.2 *Spring Problem 2*

This problem illustrates a simple extension to a 2-particle-3-spring system, without gravity (Figure 7.5). Problems like this justify the implementation of a separate *GravitationalField* class. Care has to be taken to use correct coordinates for the particles. Appropriate links with the springs produce the forces, and the equations of motion follow by linking forces with particles.

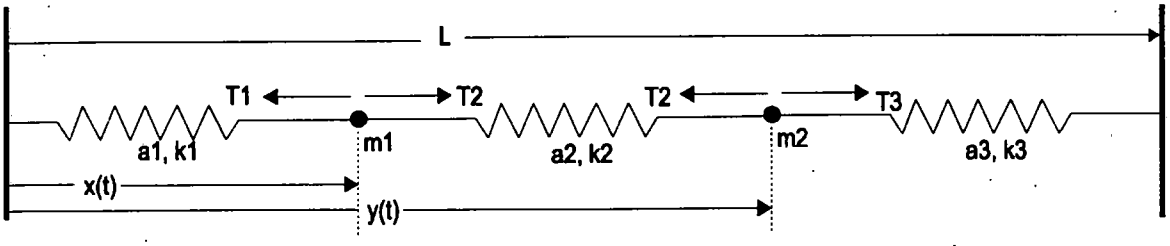


Figure 7.5

Construct the objects:

```
P1=new[Particle,Cartesian[{{x[t],0},0,1}],{},t,m1]
P2=new[Particle,Cartesian[{{y[t],0},0,1}],{},t,m2]
S1=new[Spring,Cartesian[{{0,0},0,1}],{},
      Cartesian[{{x[t],0},0,1}],a1,k1]
S2=new[Spring,Cartesian[{{x[t],0},0,1}],{},
      Cartesian[{{y[t],0},0,1}],a2,k2]
S3=new[Spring,Cartesian[{{y[t],0},0,1}],{},
      Cartesian[{{L,0},0,1}],a3,k3]
```

Link the first particle with the appropriate springs (using the *Principle of Adjacency*, these are the springs adjacent on the diagram):

```
T1=LinkObjects[P1,S1]
T2=LinkObjects[P1,S2]
EoM1=LinkObjects[P1,T2+T1]
Equation[EoM1]
```

Link the second particle with the correct springs:

```
T22=LinkObjects[P2,S2]
T3=LinkObjects[P2,S3]
EoM2=LinkObjects[P2,T3+T22]
Equation[EoM2]
```

The results are:

```
{m1 x''[t] ==
   a1 k1 - a2 k2 - k1 x[t] - k2 x[t] + k2 y[t], True}
{m2 y''[t] ==
   a2 k2 - a3 k3 + k3 L - k2 x[t] - k2 y[t] - k3 y[t],
True}
```

The advantage of the O-O approach is, again, the consistency of the toolkit approach. There is an unfortunate side-effect: T2 and T22 are not the same. Their magnitudes have opposite signs and their directions are equal, as the queries below show. This is consistent, but undesirable.

```
Magnitude[T2]
Magnitude[T22]
Direction[T2]
Direction[T22]

-(a2 k2) - k2 x[t] + k2 y[t]
a2 k2 + k2 x[t] - k2 y[t]
0
0
```

There was no problem in incorporating gravity into the 2-particle-3-spring scenario. There are clear extensions to determining normal modes.

7.9.2.3 *Spring Problem 3*

This problem is more unusual, and is much more difficult in computational terms. A particle is suspended, under gravity, from two springs which have different characteristics (Figure 7.6). The task is to find the equation of motion of the particle in two dimensions.

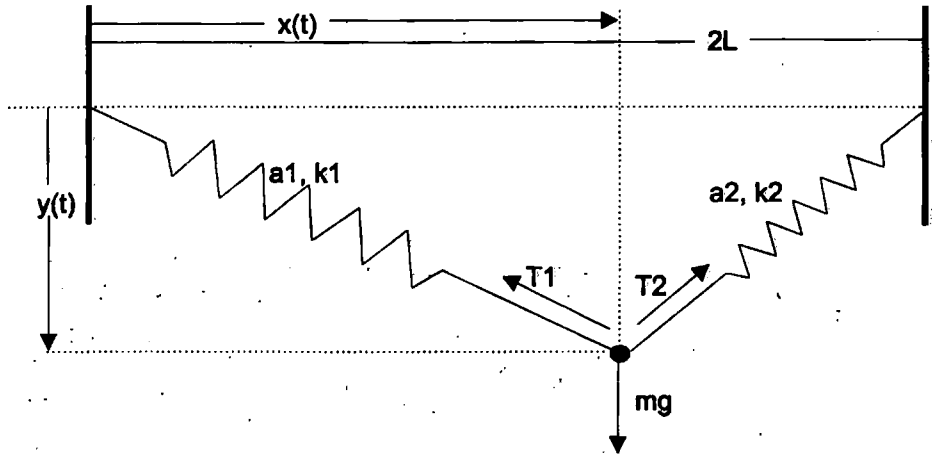


Figure 7.6

In AMK, the problem is no harder to formulate and solve than a relatively simple problem. The code runs slowly, but the links do all the heavy computation, which is considerable for this problem. This example has $a1 = 4$, $k1 = 1$, $a2 = 3$, $2L = 10$, $k2 = 2$.

```
P=new[Particle,Cartesian[{{x[t],y[t]},0,-1}],{},t,m]
GF=new[GravitationalField,Cartesian[{{0,g},0,-1}],{}]
S1=new[Spring,Cartesian[{{0,0},0,-1}],{},
    Cartesian[{{x[t],y[t]},0,-1}],4,1]
S2=new[Spring,Cartesian[{{x[t],y[t]},0,-1}],{},
    Cartesian[{{10,0},0,-1}],3,2]
T1=LinkObjects[P,S1]
T2=LinkObjects[P,S2]
W=LinkObjects[P,GF]
EoM=LinkObjects[P,T1+T2+W]
Equation[EoM]
```

The result is:

$$\{mx''[t] == 20 - 3x[t] + \frac{4x[t]}{\sqrt{x[t]^2+y[t]^2}} - \frac{60}{\sqrt{100-20x[t]+x[t]^2+y[t]^2}} + \frac{6x[t]}{\sqrt{100-20x[t]+x[t]^2+y[t]^2}},$$
$$my''[t] == gm - 3y[t] + \frac{4y[t]}{\sqrt{x[t]^2+y[t]^2}} + \frac{6y[t]}{\sqrt{100-20x[t]+x[t]^2+y[t]^2}}\}$$

7.9.3 Contact problems

Contact problems show how the implementation of the class hierarchy is transparent to the user, and how the modelling problems are essentially very similar. One planar surface problem was described in (Mitic 95A). The geometric aspects of this type of problem are important. This is particularly so for problems in which objects have coordinates of different types or orientations. In order to deal with these, coordinates must be transformed. This is equivalent to resolving forces, but is easier because all that is required is a statement of the structure of a new coordinate system.

7.9.3.1 *Contact problem 1*

Consider, first, a particle P of mass *m* moving on a rough horizontal plane. It is pulled by a force Q of magnitude *q*, inclined at an angle ϕ to the horizontal. The contact between P and Q gives rise to a normal reaction, R, and a frictional component, μR. All are referred to a Cartesian coordinate system (Figure 7.7)[1]

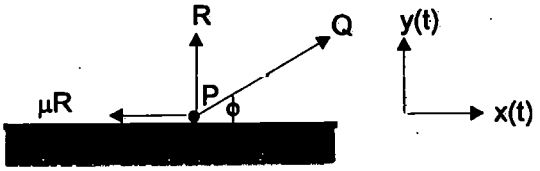


Figure 7.7

Four objects are constructed: the particle, P, the gravitational field, GF, the plane, HP, and the force which pulls the particle, Q. In accordance with the modelling cycle, links are made to produce forces, and the particle is then linked with these forces to produce

[1] There is an implicit assumption in Figure 7.7 that $Q\cos\phi > \mu R$.

an equation of motion. P is linked with HP to produce the contact force, CF, which is a vector (R, μR). The computations in this step are done automatically. The symbol R for the normal reaction must be supplied. Similarly, the mechanics of combining Cartesian and polar forces are automatic. The required inputs are:

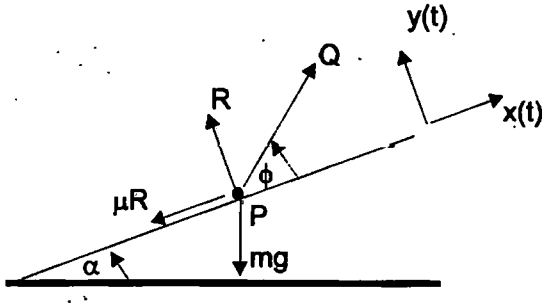
```
P=new[Particle,Cartesian[{{x[t],0},0,1}],{},t,m]
GF=new[GravitationalField,Cartesian[{{0,g},0,1}],{}]
HP=new[HorizontalPlane,Cartesian[{{x[t],y[t]},0,1}],
                    {NormalReaction->R}, mu]
Q=new[Force,Polar[{{q,phi},0,1}]]
CF=LinkObjects[P,HP]
W=LinkObjects[P,GF]
EoM=LinkObjects[P, W+CF+Q]
Equation[EoM]
```

The result is a list.

```
{m x''[t] == -(mu R) + q Cos[phi],
          0 == -(g m) + R + q Sin[phi]}
```

7.9.3.2 *Contact problem 2*

Few changes are needed to solve the same type of system, but with a plane inclined at an angle α to the horizontal. The natural coordinate system for this situation has axes aligned along the line of, and perpendicular to, the line of greatest slope of the plane (Figure 7.8). In this problem the particle moves up the slope. [2]



(Figure 7.8)

The same objects as before are defined, except that the plane, IncP, is an instance of the class *InclinedPlane*. The syntax of the term `Cartesian[{{x[t],0},alpha,1}]`

[2] There is an implicit assumption in Figure 7.8 that $Q\cos\theta > \mu R + mg\sin\theta$.

means that a right handed set of coordinate axes is rotated clockwise through an angle α. It is natural to use the x and y axes as stated for all objects except the gravitational field, which is expressed in terms of non-rotated axes. In order to combine the weight W with other forces, a common coordinate system must be used. This is the purpose of the cast `ToCartesian[W,alpha,1]`, which creates components of weight referred to the axes *x* and *y*. This process is slightly awkward, but no formal force resolutions are needed.

```
P=new[Particle,Cartesian[{{x[t],0},alpha,1}],{},t,m]
GF=new[GravitationalField,Cartesian[{{0,-g},0,1}],{}]
IncP=new[InclinedPlane,Cartesian[{{x[t],y[t]},alpha,1}],
                    {NormalReaction->R}, mu,alpha]
Q=new[Force, Polar[{{q,phi},alpha,1}]]
CF=LinkObjects[P,IncP]
W=LinkObjects[P,GF]
W1=ToCartesian[W, alpha,1]
EoM=LinkObjects[P, Q+CF+W1]
Equation[EoM]
```

The output is another list.

```
{m x''[t] ==
          -(mu R) + q Cos[phi]- g m Sin[alpha],
 0 == R - g m Cos[alpha] + q Sin[phi]}
```

7.9.3.3 *Contact problem 3*

This is a standard problem in which a particle P moves down the line of greatest slope on the surface of a circular cylinder, CS (Figure 7.9). In principle, the AMK analysis is very similar to the previous two contact problems, except for an AMK equivalent of resolving forces. This involves a tricky coordinate change.

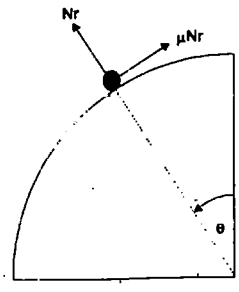


Figure 7.9

First, create the relevant objects.

```
P = new[Particle, Polar[{{R0, th[t]}, Pi/2, 1}], {}, t, m]
GF = new[GravitationalField, Cartesian[{{0, -g)},0,1}], {}]
CS = new[CircularSurface, Polar[{{R0, th[t]}, Pi/2, 1}],
      {NormalReaction -> Nr}, mu, R0]
```

Next, compute the weight and resolve it along the line of the radius vector. This resolution is unavoidable. It happens automatically for the reaction, which gives its result in Cartesians.

```
W = LinkObjects[P, GF]
W1 = ToCartesian[W, th[t] + Pi/2, 1]
Reaction = LinkObjects[P, CS]
```

The last stage is standard. The process uses polar acceleration components implicitly. This is both an advantage, because computational complexity can be avoided, and a disadvantage because it evades the detailed content of the interaction.

```
EoM=LinkObjects[P,Reaction+W1]
Equation[EoM]
                    2
{-(m R0 th'[t] ) == Nr - g m Cos[th[t]],
 m R0 th''[t] == -(mu Nr) + g m Sin[th[t]]}
```

7.9.4 Simple Pendulum

In an earlier remark on the *InextensibleString* class, I mentioned the problems of implementing a simple pendulum model in polar coordinates. This is technically much harder to program and to model than in Cartesian coordinates. Both versions are provided here for comparison.

7.9.4.1 *Simple pendulum: polar coordinates*

The angular coordinate is measured as in Figure 7.10.

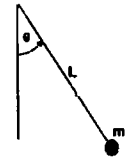


Figure 7.10

```
P = new[Particle,
     Polar[{{L, th[t]}, th[t] - Pi/2, 1}], {}, t, m]
IS = new[InextensibleString,
      Polar[{{-Te, 0}, th[t] - Pi/2, 1}], {Tension->Te},
      Polar[{{L, th[t]}, th[t] - Pi/2, 1}]]
GF = new[GravitationalField, Cartesian[{{0, g},0,-1}], {}]
W = LinkObjects[P, GF]
F = LinkObjects[P, IS]
Fr = ExpressAsCartesian[F, th[t] - Pi/2]
Wr = ToCartesian[W, th[t] - Pi/2, 1]
E1 = LinkObjects[P, Wr + Fr]
Equation[E1]
```

The solution obtained is the expected:

```
                2
{-(L m th'[t] ) == -Te + g m Cos[th[t]],
   L m th''[t] == -(g m Sin[th[t]])}
```

Several disadvantages are apparent with this formulation.

1. The negative tension was necessary to ensure correct directions and consistency of the coordinate system.
2. The polar coordinate initial line, `th[t]-Pi/2` is awkward, but was needed for consistency with a general polar coordinate system.
3. The method `ExpressAsCartesian` had to be used instead of `ToCartesian` to effect a conversion to the correct form of Cartesian coordinates. The latter did not work correctly because it could not cope with a rotated coordinate system.

7.9.4.2 *Simple pendulum: cartesian co-ordinates*

The formulation is much easier, but the answer is in an unusual (but correct!) form.

```
P=new[Particle,Cartesian[{{x[t],y[t]},0,-1}],{},t,m]
GF=new[GravitationalField,Cartesian[{{0,g},0,-1}],{}]
IS=new[InextensibleString,Cartesian[{{0,0},0,-1}],
        {Tension->Te},Cartesian[{{x[t],y[t]},0,-1}]]
```

```
T1=LinkObjects[P,IS]
W=LinkObjects[P,GF]
EoM=LinkObjects[P,T1+W]
Equation[EoM]
```

$$\{m\,x''[t] == -\left(\frac{Te\,x[t]}{Sqrt[x[t]^2 + y[t]^2]}\right),$$

$$m\,y''[t] == g\,m - \frac{Te\,y[t]}{Sqrt[x[t]^2 + y[t]^2]}\}$$

7.10 Extension: new Classes

In Chaper 6 I indicated that it was not difficult, in theory, to add new classes to a problem domain, and to override methods in existing classes by deriving subclasses from them. In order to test this in practice, I amended the original AMK code some time after it was originally written. The AMK system, as originally programmed, contained no *Dashpot* class. It proved to be easy to add a *Dashpot* class by amending the code for *Spring* and LinkObjects(Spring, Particle). The *Spring* and *Dashpot* classes are similar: the major difference is the nature of the resistive force that results when associated with a particle. This means that *Dashpot* can be implemented in two ways: as a subclass of *Spring* or as a subclass of *CoordinateSystem*. Both worked successfully. It is easier conceptually to contemplate the latter, so I prefer this marginally. Some problems of defining subclasses are discussed in the section on override classes. With the addition of the Dashpot class, the problem domain could be extended to cover Spring-Dashpot systems. A simple system that results in a powerful model is shown in Figure 7.11. It models a car suspension system with a rough road. A coordinate y[t] provides a generic form for the road surface (y[t] = $A\sin(wt)$ is a common simplifying assumption).

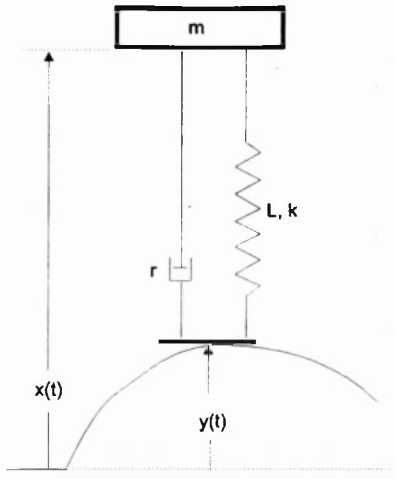


Figure 7.11

The AMK input script for this problem is:

```
P=new[Particle,Cartesian[{{x[t]},0,-1}],{},t,m]
S =new[Spring,Cartesian[{{0},0,-1}],{},
     Cartesian[{{x[t]-y[t]},0,-1}],L,k]
DPot =new[Dashpot,Cartesian[{{0},0,-1}],{},
     Cartesian[{{x[t]-y[t]},0,-1}],r,t]
GF=new[GravitationalField,Cartesian[{{g},0,1}],{}]
W=LinkObjects[P,GF]
F=LinkObjects[P,DPot]
T1=LinkObjects[P,S]
Res = T1+F
TotalForce = Res + W
ESP=LinkObjects[P,TotalForce]
eq=Equation[ESP]
```

The output was (correctly!):

```
{m x''[t] == k L - g m - k x[t] + k y[t]
                - r x'[t] + r y'[t]}
```

Three points arose from this experiment:

1. It was tricky to get the geometry right in this problem. There were two particular problems. First, since the coordinate $x[t]$ is measured 'up', `S.End1 = 0` and `S.End2 = x[t]`, not the other way round. Second, the direction of the

gravitational field opposes the direction of x increasing: hence the choice of GF.Sense.

2. Combining the three forces using the overloaded '+' operator technique is far too slow. This is why the intermediate results `Res` (resistance) and `TotalForce` were computed first: it was faster than computing `T1+F+W`, which would have been preferable. The reason for the slowness of this operation is unclear, but the most likely reason is circular references in coordinate transformations.
3. Mathematica 'remembers' errors, so any that occur in defining new classes and procedures remain in memory until they are cleared. This leads to confusion when subsequent amendments appear to have no effect: Mathematica is using the first versions defined instead of the latest versions...

7.11 Extension: override objects

In order to test the practicalities of amending a property of an existing object, I implemented a *NonLinearSpring* class. When linked with a particle, the tension in the non-linear spring is $k(x-L)^p$ instead of $k(x-L)$, where p is an additional attribute of the new class. The process involved:

- Defining a class *NonLinearSpring*, descended from *Spring*;
- Defining a function `NonLinearSpringQ` and amending `SpringQ` to

```
SpringQ[x_] := isa[x,Spring] &&! (isa[x,NonLinearSpring])
```

so that *Spring* and *NonLinearSpring* can be distinguished;

- Writing a `LinkObjects[Particle, NonLinearSpring]` function, identical to `LinkObjects[Particle, Spring]` except for an amended force magnitude.

The equation of motion for a mass suspended from a fixed point by this more exciting spring emerged correctly:

```
                                    p
{m x''[t] == g m - k (-L + x[t]) }
```

7.12 Extension: other problem domains

In Chapter 6 I presented a theoretical account of the *Diagram-Axiom* modelling methodology for several contexts. The purpose of this section is to demonstrate that this theory can be translated into working software. To do this, I applied the same techniques that were used in AMK to program a class hierarchy for the simple heat transfer problems of Chapter 6. The purposes of this exercise were to demonstrate:

- that the AMK techniques are applicable to other contexts;
- direct method calls to link objects (i.e. use ObjectA.LinkWith(ObjectB) instead of LinkObjects(ObjectA, ObjectB));
- support for the *Diagram-Axiom* modelling methodology;
- inclusion of a *SolveHeatEquation* method for a *HeatEquation* class, as an example of a 'solve' method.

Heat transfer problems constitute a small problem domain with few classes, which is ideal for a simple demonstration of these points.

I successfully developed the software implementation, HEAT, and an example problem is solved in Appendix 7H. Although no major problems were encountered, programming the detail was tricky, and could not be done by inexperienced programmers. Linking objects immediately after creation provides support for the Diagram-Axiom methodology, although the modelling sequence can consist of constructors, followed by links, and then the method calls to get the results. Using direct method calls to link objects worked successfully but the external LinkObjects procedure results in more readable (and hence maintainable) code. The only essentially different treatment was in linking a *HeatSource* object with several *Layer* objects. The result is advantageous because a construct such as

```
LinkLayer[HeatSource1,{Layer_1,Layer_2,…,Layer_n}]
```

immediately performs the necessary computation to produce a heat equation, independent of the number of layers listed.

7.13 Evaluation

This chapter has demonstrated that AMK can solve certain problems successfully. Here I summarise AMK's strengths and also describe classes of problems that are difficult or impossible to solve with AMK.

7.13.1 AMK: Limitations

AMK relies on creating and linking objects. If this cannot be done it is not possible to formulate a model. Superficially, a situation where no objects are created and no objects interact constitutes an empty problem domain. However, there are well-defined problems with an essentially geometrical nature which AMK cannot solve. They depend on a class which characterises a geometrical situation. Such a class does not exist in the present implementation of AMK. Alternatively, geometrical properties might be included as attributes and methods of *Particle*. An example is the brachistochrone problem, described in, for example, (Dettman 62).

The brachistochrone problem requires a particle to travel from point $O(0,0)$ to $Q(a,b)$ along an undetermined path $y(x)$ in minimum time. Two aspects of this problem are pertinent. The first comes from particle mechanics, and is the energy balance equation $\frac{1}{2}mv^2 = mgy$. This can be coded as a method of *Particle*. The second describes the path's geometry: ds = v dt, with $ds = \left(1 + y'^2\right)^{\frac{1}{2}} dx$. This geometrical aspect is absent from AMK. It could be coded as an attribute of *Particle*, but can also be considered as an attribute of either *ReferenceFrame* or of a *Geometry* class (not yet defined). The result of linking *Particle* with *Geometry* is unclear. The requirement is to substitute $v = \sqrt{2gy}$ (from the energy equation) into ds = v dt, and minimise over all possible paths $y(x)$ to obtain the functional equation $t = \underset{y(x)}{Min}\left(\int_0^a \left[\frac{1+y'^2}{2gy}\right]^{\frac{1}{2}} dx\right)$. It does not seem appropriate to code all these aspects as methods of *Particle*: the constraint on the path implies the presence of a surface which never enters into the analysis. A further aspect of this problem which can cause problems is to recognise that solving the functional equation must be done by Euler-Lagrange techniques.

The main classes of solvable problems with which AMK has difficulty are those for which it is difficult to draw a diagram. These are likely to be heuristic models with no axiomatic basis. Alternatively, there may be no or only a limited number of physical objects in the problem domain. An example is the study habits problem of Chapter 6. In order for AMK to solve this type of problem, symbols must be assigned to concepts (for example the time spent studying, depth of study, or amount remembered). This process determines classes, to which attributes and methods must be assigned. Since all of this analysis must be done from scratch, an object-oriented analysis is of dubious value: a traditional analysis suffices. However, the *Axiom Strand* of Chapter 6 may be useful to define classes. Once the analysis has been done, it may provide base classes for further analysis. It is only at this stage that the O-O analysis can add significant value.

Applying O-O techniques to the context of Newtonian Particle Mechanics involved transforming between Cartesian and Polar coordinates. It was difficult to design a class library that did this, and also eliminated mutual recursion. The class library has a reasonable structure for this context but the undesirable effects of recursion make it very slow in certain circumstances. Recursive side effects are exacerbated by repeated evaluation of expressions. There is therefore a conflict between structural and functional efficiency.

AMK does not cope well with problems involving Polar coordinates. It was difficult to code without retaining complete generality because classes had to cover specific situations (such as the sample pendulum or motion on a circular surface). This problem could be eased by providing more general attributes and methods which can cope with motion on a curve.

There is currently no functionality in AMK for dealing with multi-stage problems, in which there is a sequence of interactions in time. The problem of the extensible string has already been discussed in this chapter. The significant point about this situation is that once the string becomes slack (slackness being a property of the string), the state of the particle changes. At the point when the string becomes slack, the only force acting on the particle is its weight. Another example is a ball bouncing on an elastic surface. The height of each bounce reduces by a factor e at every bounce, where e is a coefficient of restitution. At each stage there is an interaction between a particle and a surface. The output parameters at each interaction become input parameters for the next interaction.

In order to cope with this situation efficiently, AMK would need to include this sequence of interactions in a loop. However, the details of each interaction (forms of the inputs and outputs) must be known in advance.

7.13.2 AMK: Strengths

AMK's strength is that relatively complicated mathematical models can be constructed using a small number of generic classes, provided that they are linked in a meaningful way. Only 9 classes (Section 7.6) provide inputs for problems, and there is only one class for outputs: *EquationOfMotion*. Section 7.6 also shows that there are only 9 feasible links. The *Particle-Spring* systems in Section 7.9.2 are good examples of how complex systems can be built from only three classes: *Particle* and *Spring* and *GravitationalField*. The *Spring-Dashpot* systems of Section 7.10 show that even more complex problem domains can be tackled with the addition of only one more class: *Dashpot*.

This method of doing particle mechanics forces the user to think about the geometry of the situation and about the objects in the system. The user links the objects in a way which parallels how a diagram showing forces and coordinates would be drawn. Mathematica deals with the mechanics of producing new objects. The system works with a wide variety of situations in particle mechanics, without needing a dedicated template for each conceivable situation. Automation can disguise algebraic manipulations which, although tedious, should be something that the user can do if necessary.

The present implementation is restricted to given classes within the context of Newton's Second Law of Motion. The conceptual framework of the system can be extended to include momentum, work and energy, and relative motion, which would provide scope to solve a wider class of problems. In addition, a means for the user to define new objects would be advantageous.

Chapter 8

Front-end Support for the Modelling Cycle

8.0 Abstract

Mathematical modelling methodologies allow the user to deviate from the constraints of using the methodology. This chapter illustrates how a front-end for the modelling process constrains the user to use the methodology rigorously. In doing so it provides three components. First, it supplies an algorithm for the whole modelling process. Second, it automates production of links between objects. Third, it hides the awkward syntax required for the computer algebra components. The problems associated with producing and using Mathematica scripts as described in Chapter 5 can be eased by building a user interface. Two interfaces are discussed here, and both are assessed within the context of support for a modelling cycle. Both are designed to force the user to use the *Construct→Link→Invoke_Methods* modelling cycle. The first interface produces Mathematica scripts, and is implemented in Visual Basic. It provides a simple but effective means of communicating with Mathematica, forcing the user to follow a well-defined algorithm in the crucial relation-building part of a generic modelling cycle. I discuss the advantages and disadvantages of this front end in achieving this aim, and evaluate alternative ways of implementing it. The second interface is a graphical front end, implemented in C++. The user can build a diagram on screen using icons which represent objects. Links are then generated by placing the icons sufficiently close to each other, which is a direct application of *the Principle of Adjacency*. The process corresponds more closely to a diagram which would normally be drawn as part of model construction, and is more consistent with a *Construct→Link→Invoke_Methods* modelling cycle. The second interface is not as wide-ranging as the first: it implements only a subset of the AMK classes, attributes and methods. It is sufficient to illustrate that a practical solution to the problem of explicitly generating a model as the user draws a diagram is possible.

8.1 Rationale

In Chapter 7 I gave examples of modelling using the Mathematica component of the AMK software. Modelling was done by constructing Mathematica scripts which implement an O-O methodology. Some examples indicated that it is possible to vary the order of expressions in those scripts. The main variations are to:

1. construct all objects first and then define the links until an *EquationOfMotion* object results;
2. construct a limited number of objects and then link them, repeating these Construct→Link sub-loops until an *EquationOfMotion* object results.

This freedom allows the user to make errors and stray from the O-O modelling process. The primary purpose of the ideas in this chapter is to use a front-end to support the *Construct→Link→Invoke_Methods* modelling cycle. The front-end exists to constrain the user to follow the modelling cycle, thereby ensuring that it works correctly and sensibly.

There are further purposes for a front-end. The modelling process of Chapter 7 also depends heavily on the geometry of the problem domain in many modelling scenarios. Constructing the necessary objects reflects this: coordinates have to be supplied. In some cases, signs must be supplied. The Mathematica inputs are also heavily dependent on syntax and accurate knowledge of the functional forms of the internal expressions. These three factors can make it difficult to use the Mathematica input scripts as they were presented in Chapter 7. The secondary purpose of the front end is therefore to produce those scripts without heavy reliance on knowledge of syntax.

8.2 Computer Algebra System front-ends

Computer algebra systems are generally designed as a kernel, which contains the algebra engine, and a user interface, which allows communication with the kernel. There is usually a dedicated communications protocol for doing this. This architecture allows separate development of kernel and user interface functionality, and also makes it easier to write cross-platform applications.

Ideally, a user interface for a CAS ought to communicate directly with the kernel. Several significant interfaces which do this have been written, and Kajler (Kajler 90) summarises some issues of interface design for 'mainstream' CASs such as Reduce and Maple. Third party interfaces have been written for established computer algebra systems. Examples include MathScribe (Smith 86), which is for Reduce, and CAS/PI (Kajler 92), for Maple, Sisyphe, Ulysse and two graphics packages. These general purpose interfaces are designed to be versatile, system-independent and extendible. The two main issues highlighted in (Kajler 94) are relevant to the design of an interface for AMK. First, communication should be *efficient*: only active parts of an expression should be interchanged. This is known as 'expression' sharing and is used extensively in Maple (Leong 86). The first interface described in this chapter uses *inefficient* communication because entire expressions have to be interchanged. Kajler's second issue is to find ways in which an interface can activate mathematical processes. Bonadio (Bonadio 89) does this in Theorist: manipulating icons on the screen initiates 'solve' processes automatically. This is the idea behind the second interface described in this chapter.

The CASE tool SPADE (Seppanen 91) performs a similar function to the software of this chapter (but not for a CAS) in that it guides the software design process. SPADE does this by producing and displaying views for an established design methodology, but produces a software design specification for large software systems, not a model. It cannot therefore abstract requirements for mathematical modelling easily.

The interfaces described in this chapter are designed to communicate with the front-end of Mathematica. This is easier than attempting to communicate directly with the kernel, and is sufficient from the point of view of model development. At an early stage in the investigations for this thesis, I programmed a menu-driven interface in Pascal for MuMath, using the method in (Rickhuss 93). MuMath is, in effect, the kernel for DERIVE, although the current DERIVE kernel has advanced since DERIVE first appeared in 1991. Choosing menu items in my MuMath interface generated a MuMath expression in a text file, *msgfile.txt*. An example is `int(x^2*#e^x,x,0,a)`, for $\int_0^a x^2 e^x dx$. The DOS command `MUMATH.EXE < "msgfile.txt" > "ans.txt"` then invoked the MuMath kernel, reading the expression in *msgfile.txt*, evaluating it,

and depositing the result in a second text file, *ans.txt*. The interface could then display the contents of *ans.txt*.

I used a similar technique to build a menu-driven interface for Mathematica 1.2, which was command-driven and DOS-based. The essential difference between the Mathematica and MuMath interfaces was that the Mathematica kernel called its interface as a sub-process, rather than the other way round (as described in the previous paragraph). Three commands were involved: `Run[]` to activate a sub-process, `ReadStrings[]` to read the text file generated by the interface, and `WriteStrings[]` to write the result to a second text file.

These exercises demonstrated the following points.

- Direct communication with a kernel could be slow without the use of a dedicated communications protocol.
- Running a sub-process (either a CAS kernel or an interface program) frequently caused program failures due to insufficient memory.
- Message-passing between a kernel and its front-end can necessitate lower level (often recursive) programming constructs. It is more productive to use higher level programming constructs and not communicate directly with a kernel. [1]

As a result of these early attempts at interface design, I used independent Windows applications for the interfaces of this chapter and the CAS. This improved the stability of all programs concerned. It also allowed Mathematica procedures to do all the mathematical manipulations, which improved efficiency. I retained the text file communications method: it was stable, was sufficiently fast, and was easy to program.

[1] The DERIVE kernel is now available as a DLL, and can be linked into compiled applications. Communication with it is easy: function calls pass DERIVE commands strings, but the problem of awkward programming constructs remains.

8.3 Menu-driven Interface

A brief account of this first user interface appears in (Mitic 95B). In this chapter I refer to it as IF1 (Interface 1).

8.3.1 Support for the Modelling Cycle

IF1 reinforces the *Construct→Link→Invoke_Methods* modelling cycle by incorporating three principal stages: Construct, Link and Methods. I illustrate these processes with some sample screen dumps in Section 8.3.5. The *Construct→Link→Invoke_Methods* modelling cycle specifies a nominal sequence in which objects are constructed, linked and any necessary methods are called. The final stage is to call an *Equation* method, which generates the required equation of motion. In principle, it is possible to intersperse construction and linking. This order of operations is sensible, but not necessary. An alternative is to construct all objects before any of them are linked. In some cases this is not possible. For example, coordinate transformations are sometimes needed in order to combine forces, and methods can be accessed at any time to retrieve information. A top level scheduler form initiates each of the three stages, and this gives rise to a modelling sub-cycle as below. The output of the grouped *Construct→Link→Invoke_Methods* unit is the *EquationOfMotion* object whose *Equation* method is called before sending the script to Mathematica.

IF1 avoids editing and manipulating expressions except in a very elementary way, and does not require a command language. An event-driven environment replaces commands, the principal event being the mouse click. As each stage proceeds, the interface constructs Mathematica inputs. They can be viewed in a window (for reference) as they are created. When complete, the script can be sent to Mathematica. The result is automatically returned to the interface software as well as appearing in Mathematica's own front-end. Its implementation allows the user to be presented with relevant data input forms at appropriate stages in building a Mathematica script. This forces the user to stick to the modelling cycle in Figure 8.1.

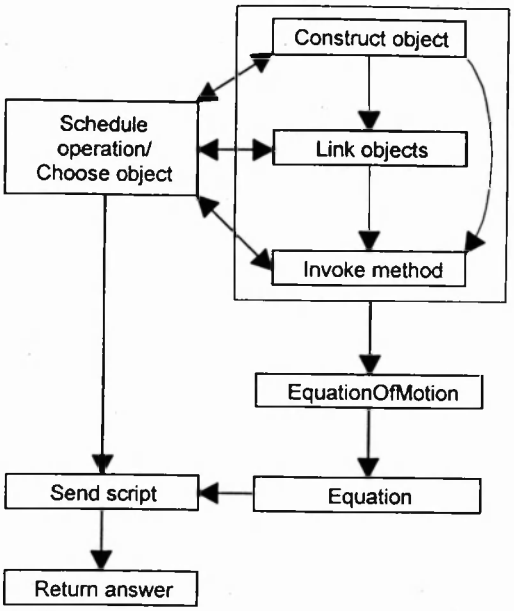


Figure 8.1

Error-trapping devices ensure that:

- disallowed link combinations are impossible (e.g. *Spring* with *GravitationalField*);
- links cannot be made with non-existent objects;
- only available methods can be invoked.

These error checks not only reduce errors: they direct the modelling process by reinforcing the general strategy. In particular they stress the overall aim of getting an *EquationOfMotion* object by emphasising *Particle-Force* links.

8.3.2 Implementation

Visual Basic provided the necessary tools for programming this interface:

- an environment for placing dialog boxes, command buttons, picture boxes and similar items on a form;
- a mouse-driven environment;
- sufficient functionality to communicate with Mathematica for Windows.

Since IF1 is used for data capture (in order to construct an input script for Mathematica), it need not reflect the class structure of AMK. This was not even possible in VB (version 1), which cannot support an O-O environment. Only events (generated by the mouse and the keyboard) correspond to the AMK actions of *Construct*, *Link* and

Invoke_Methods. The precise response to such an event is determined by using *Select Case* constructs liberally. These are hard coded, which makes it difficult to maintain the interface as changes in the AMK object model occur.

In the current version of Mathematica (Version 3.0) it is possible to create command buttons and launch Mathematica procedures from them. Unfortunately the command button environment is insufficiently rich to emulate the VB front-end functionality, appearance and versatility.

8.3.3 Communications Protocol

The communications protocol employed is relatively unsophisticated but serves the purpose of providing a workable link without an excessive programming overhead. The VB interface sends a Mathematica script via the Windows clipboard. VB provides constructs which make this particularly convenient. Mathematica output cells cannot be placed on the clipboard without specifically selecting them, but this was not possible in the original Mathematica (Version 2) implementation.[2] Outputs remain in the Mathematica notebook. VB has the particular advantage that it can control Mathematica operations using its *SendKeys* procedure. This sends keystrokes to external applications, and can control Mathematica's menu system.

There are two alternatives to using the clipboard. The first is Rickhuss' method (Rickhuss 93), which is slower. The second is MathLink. MathLink would have been preferable, but was not available when I started this interface. Furthermore, it was uncertain (and still is!) whether or not VB could use MathLink successfully.

[2] It is possible in Version 3

8.3.4 Editing and Filing

There are minor editing and filing facilities. In particular, input scripts can be saved and reloaded. They can also be edited to some extent. Providing minor editing facilities is a design feature, and forces the user to consider each construction, link and method. Editing is useful only if there is a clear similarity between a script before and after editing. For example, it is possible to copy and correct a minor error, and delete the original. A more useful purpose is to create an object identical to an existing object except in one respect. Similarity and symmetry are supported in this way. The *AMK* interface editor thus functions like the editor described for GI/S, a front end for Macsyma (Young 87). This controls the history of individual expressions by allowing for recall and editing of previous expressions. However, they differ in that the *AMK* editor cannot manipulate sub-expressions and relies on linear input. Both of these problems were pointed out in (Kajler 94), from which it is clear that considerable development work is needed to solve them.

IF1 is programmed to cope with a limited number of AMK classes. It is extendible, but only by re-coding and recompiling. A clear advance would be to provide a 'class editor' in which existing class attributes and methods can be amended, and new classes can be created.

8.3.5 The Interface in use

This section shows how IF1 is used to construct a script to send to Mathematica. The model in this example is simple:

"A particle of mass m is subject to two forces: its weight and a force $F(t)$, which acts vertically upwards. With a coordinate y for the upward vertical direction, show that the equation of motion is $m\ddot{y}(t) = F(t) - mg$. "

The main form of IF1 acts as a 'Scheduler'. It enables the user to activate Mathematica, construct objects, link them, call their methods, and send a script to Mathematica for evaluation. Control is returned to this 'Schedule' form after each activity.

The first stage after activating Mathematica is to choose an object to construct using the 'Schedule' form. This activates the 'Construct' form. Completing a 'Construct' form is the most time consuming process since attributes have to be specified in full. There are three in this problem: a *Particle* P1, a *Force* F1 and a *GravitationalField* GF. Figure 8.2a shows the completed *Construct Object* form for the particle *P*.

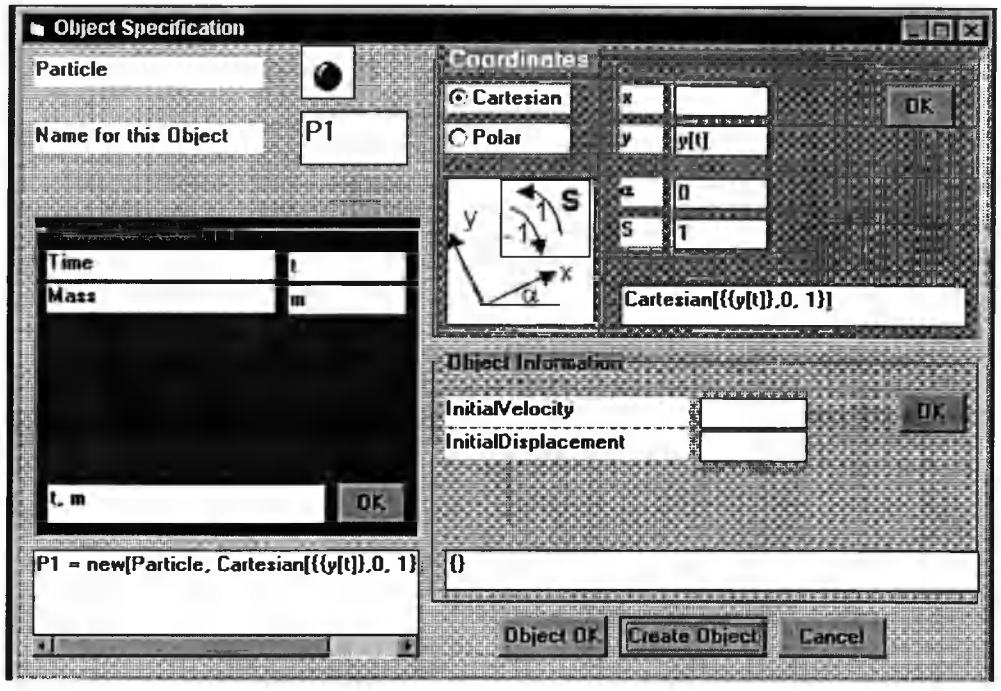


Figure 8.2a

The 'Link' command button is used three times to make the links

```
Weight = LinkObjects[P1, GF]
TotalForce = Weight + F1          and
EoM = LinkObjects[P1, TotalForce] .
```

Figure 8.2b shows the inputs for the *Weight* object. Creating the sum `TotalForce = Weight + F1` is less straightforward because a "Sum of Forces" option must be explicitly chosen, and "Weight + F1" has to be typed explicitly. This problem can be solved with further programming. The user has to choose which objects to link, and there is no control over which objects to link, and when to link them. The only constraint is that the interface prevents impossible links (e.g. *Spring* with *EquationOfMotion*).

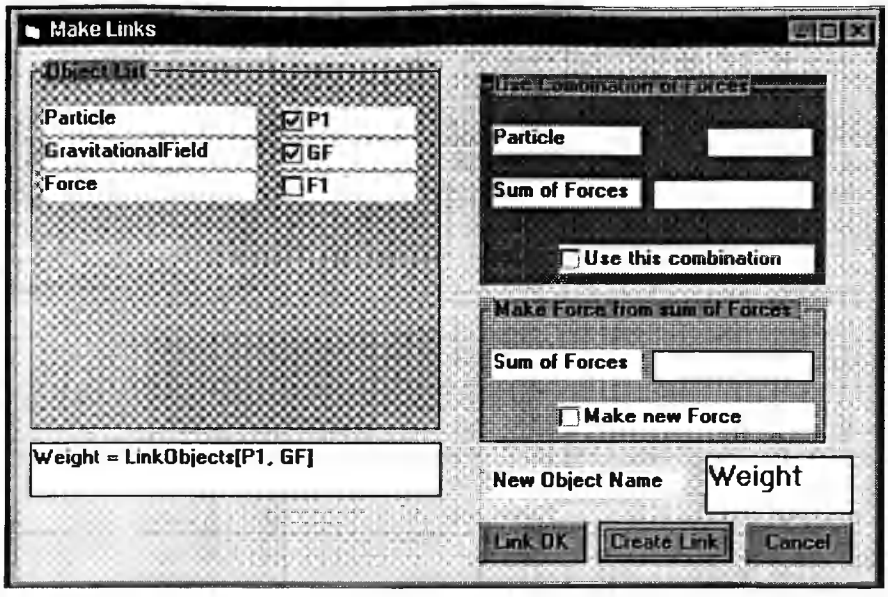


Figure 8.2b

Figure 8.2c shows the *Methods* screen, which is used to invoke the *Equation* method of *EoM*.

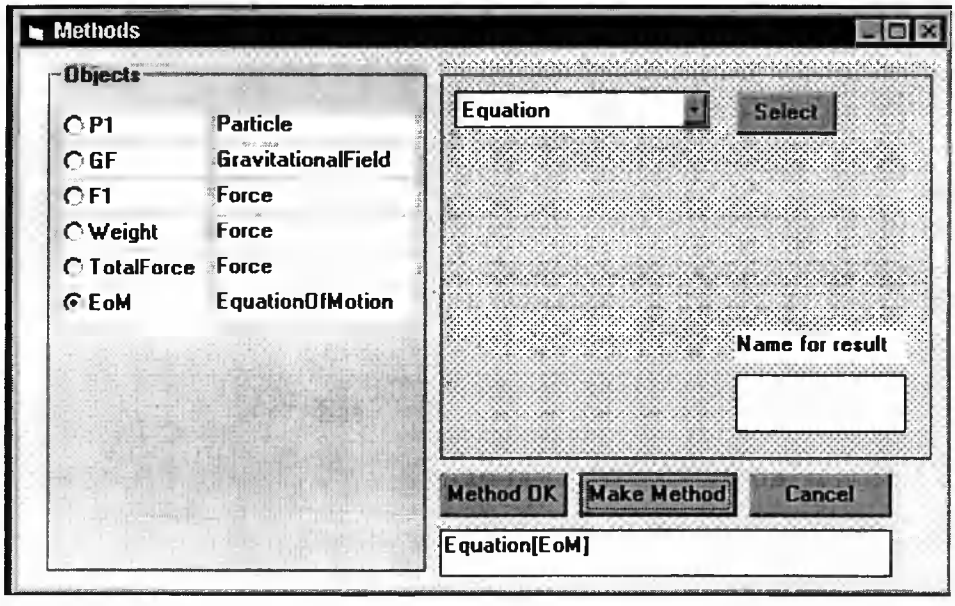


Figure 8.2c

The order in which the parts of a given form are filled in is immaterial, and amendments can be made at any stage. Figure 8.2d shows the result of all the above operations, with the Mathematica window in the background, the script produced by the interface, and the evaluated result.

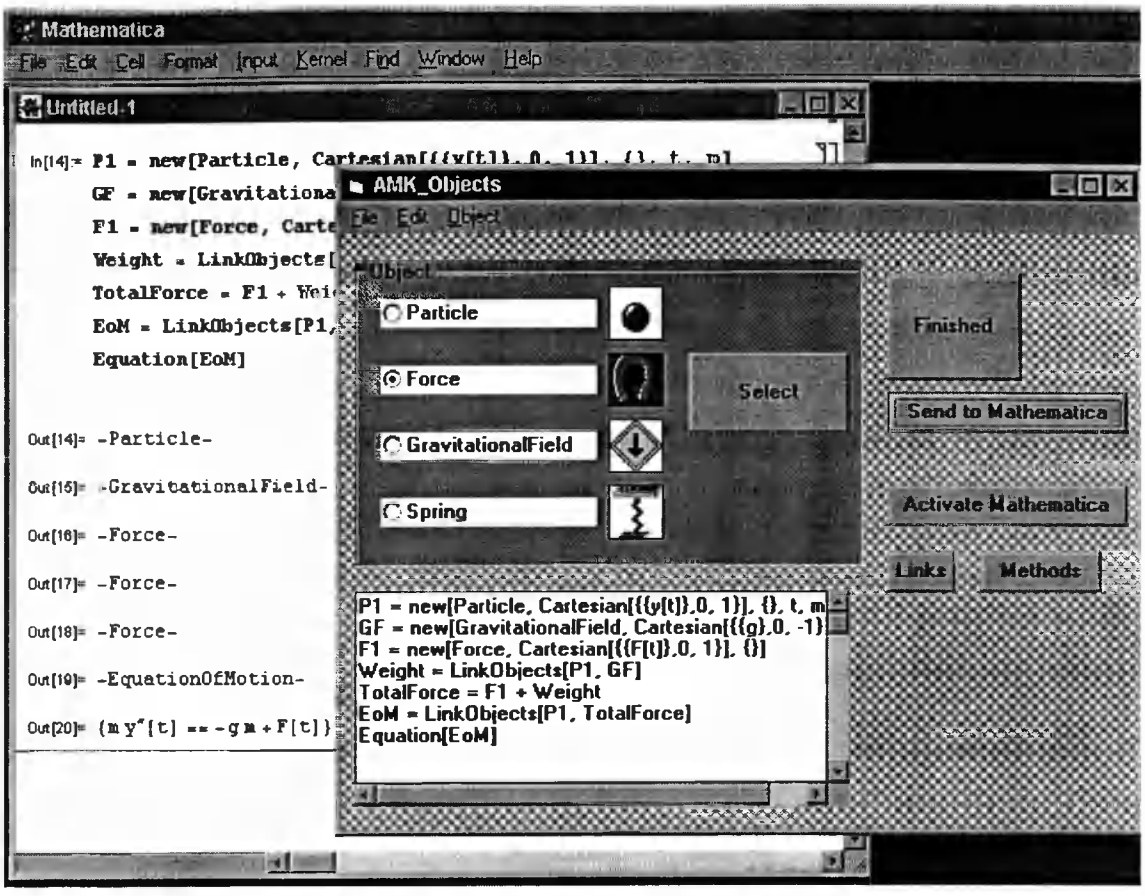


Figure 8.2d

8.3.6 Evaluation of IF1

IF1 forces the user to use the *Construct→Link→Invoke_Methods* modelling cycle, which is the essential component of this O-O technique. The simplest form of this cycle is to create all the objects first, and then make the links. The user can deviate from this sequence by linking objects at the first possible opportunity after they are created. Both are valid strategies. The interface also forces the user to specify the geometry of a situation accurately, since without geometrical considerations coordinates cannot be specified reliably. To some extent this detracts from the activity of modelling. Details can be tedious to fill in, but the editing facility eases this. There is a certain degree of error protection. For example, only methods relevant to the object in question can be chosen. There is additional protection in specific cases. For example, an attempt to link two objects for which there is no meaningful outcome, such as a *Particle* and an *EquationOfMotion*, is detected and aborted.

8.4 Icon-driven Interface

In this section I describe a second interface which I refer to as IF2 (Interface 2). It is designed to further enhance the ideas in the *Construct→Link→Invoke_Methods* modelling cycle by providing a 'graphics-driven' environment, mainly controlled by mouse clicks. It is not as wide-ranging as IF1 in that it caters for fewer AMK classes, and only for motion in one dimension. Nevertheless, it illustrates important principles (discussed in Section 8.4.1), and is a significant improvement over IF1 in terms of support for the modelling cycle.

IF2 is written in C++ and emulates the underlying object model. This makes it easy to create and manipulate 'screen' objects. It also allows for a potential MathLink communications protocol.

8.4.1 Purpose and Support for the modelling cycle

The purpose of IF2 is to:

- emulate the 'natural' modelling technique of drawing a diagram;
- support the Principle of Adjacency;
- support *Construct→Link* sub-cycles.

The idea behind IF2 is to produce elements of the object model simultaneously with drawing a diagram on the screen. The 'screen' diagram consists of moveable icons, each one of which corresponds to an element of a 'paper' diagram. Objects can be created in the same way as for IF1. When the user selects a 'screen' object and clicks on the *Link* button, IF2 searches for adjacent objects and suggests a link with any object for which a link is appropriate. This is a direct application of the *Principle of Adjacency*. When two adjacent objects are linked, a new object is created and replaces one or both of the original objects. Objects which take no further part in the modelling process are no longer visible, although they are still technically in the problem domain because their *Destroy* methods are not called. The user can invoke a method for a selected object by clicking on the *Methods* button and choosing from a menu. Each *Create*, *Link* and *Invoke_Methods* operation adds a further line to the script for Mathematica.

8.4.2 Implementation issues

In this section I discuss implementation issues which have a bearing on the way in which IF2 operates and supports the modelling cycle.

The object structure of IF2 does not mirror the AMK object structure exactly. It only needs to assemble the Mathematica script and capture data for creating objects. Class methods are therefore less relevant than class attributes. IF2 requires values for all attributes, but only needs to 'know' the names of methods and what the results of the possible links are.

The principle that after certain links, objects which are involved in that link may take no further part in the modelling process, can cause problems with 'persistent' objects, which need to be there all the time. Objects which are not needed after any given link are marked as 'inactive' and, although retained in memory, are invisible and hence inaccessible to the user. An important example is the *GravitationalField* object. It can be linked with any one *Particle* to produce a *Force* object: the particle's weight. After this link the *GravitationalField* object should have no further involvement with the *Particle* object, and should be marked as inactive. It certainly should not link with the same particle twice, which would be mathematically incorrect. I decided to make *GravitationalField* object inactive after a link with a *Particle* to avoid an incorrect second link with that *Particle*. However, if it is inactive, it cannot link with any other *Particle*. This problem can be solved by creating further *GravitationalField* objects: one for each other *Particle*. This is also undesirable because all the *GravitationalField* objects represent the same physical object. Alternatively, a *Particle* could be marked as 'linked with a *GravitationalField*' to prevent subsequent erroneous links. Nevertheless, the implemented strategy still emphasises the *Particle* + *GravitationalField* = *Force* equation.

'Allowed' links can be held as methods of the IF2 classes, or external to those objects, as in the AMK Mathematica code. For example, the *Particle* class can have a *LinkWithGravitationalField*(GF) method, where GF is the unique identifier of a *GravitationalField* (as discussed in Chapter 6). In principle, any property of this problem domain can be cast as a class method in this way. It was simpler not to implement such methods: the interface is for data capture and its object structure does

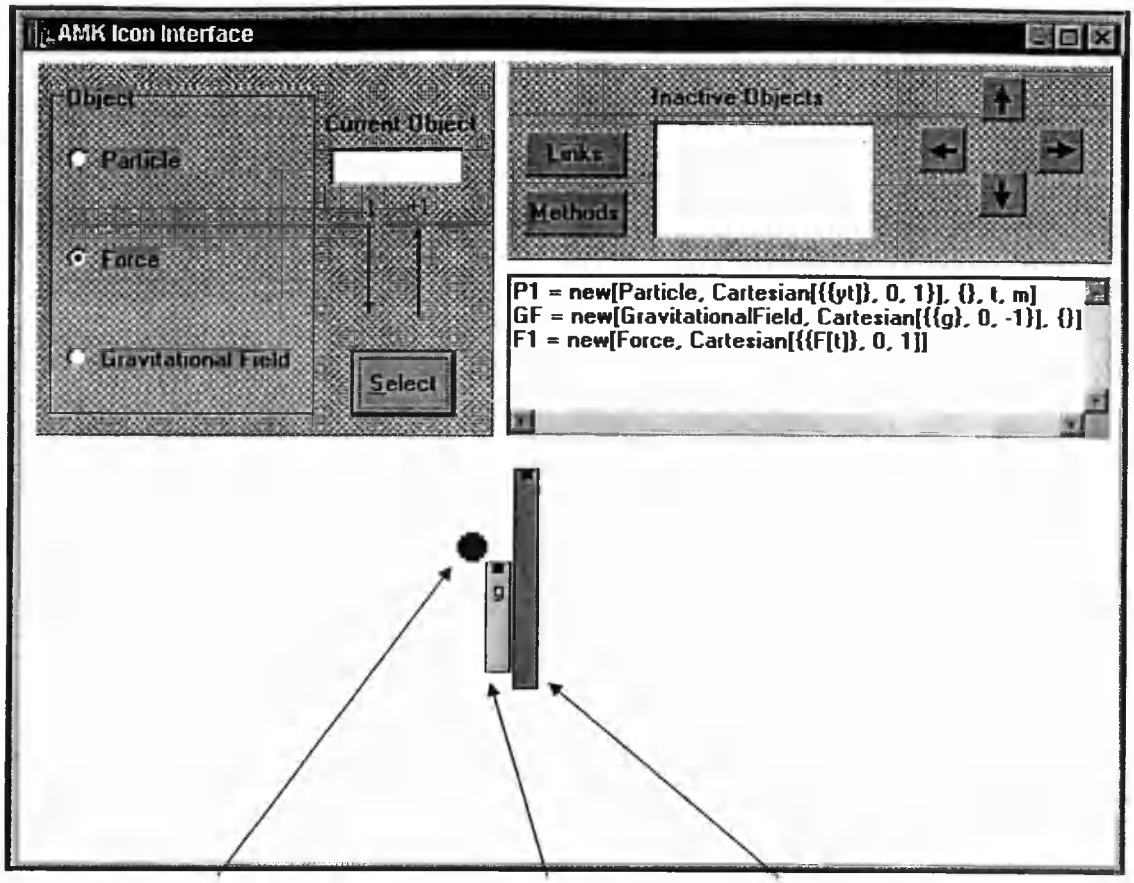
not need to reflect the AMK object structure faithfully. In IF2, a database holds details of the allowed links, and the consequences of those links. A sample record is:

ItemNo	Object1	Object2	Result	ResultCode	Hide1	Hide2
1	Particle	GravitationalField	Force	F	N	Y

This record states that the result of linking a *Particle* with a *GravitationalField* is a *Force*, for which a code letter is *F*. The fields Hide1 and Hide2 describe what happens after the link. Object1 (the *Particle*) remains active (Hide1 = 'N') and Object2 (the *GravitationalField*) becomes inactive (Hide2 = 'Y').

8.4.3 The Interface in use

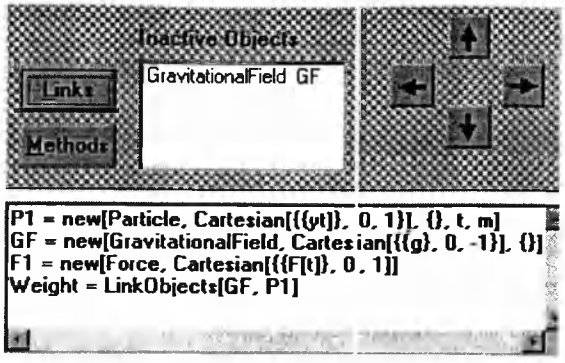
The sequence of screen dumps, Figures 8.3a to 8.3e, shows how IF2 operates. It illustrates the same problem as in Section 8.3.5. Figure 8.3a shows the entire screen, after creating the three objects P1, GF and F1 (as in Section 8.3.5). They can be dragged anywhere in the 'modelling' area, which is the blank part of the form. Fine adjustments of position are possible by using any of the four 'adjust' buttons (Figure 8.3a, top right). The 'Mathematica script' section of the form shows the current state of the model in terms of what has been created and linked, and which methods have been called.

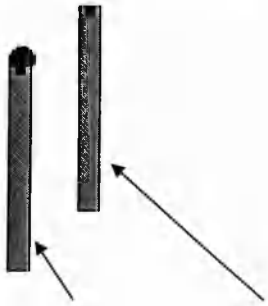


Particle P1 *GravitationalField* GF *Force* F1

Figure 8.3a

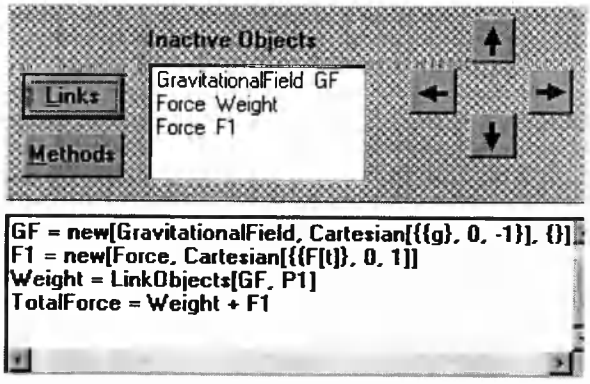
Figures 8.3b and 8.3c show a portion of the screen after two links. After the first of these (Figure 8.3b), a new *Force* (Weight) replaces *GF*, which is recorded as 'Inactive'. After the second link (Figure 8.3c) *TotalForce* results from *Weight* + *F1* and is then the only active force. It is not readily apparent which force is which, and a clear improvement to IF2 would be to label each object with its 'name'. I did not do this in the current implementation because the identity of each object is clear for simple problems.

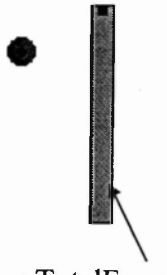




Force Weight *Force* F1

Figure 8.3b

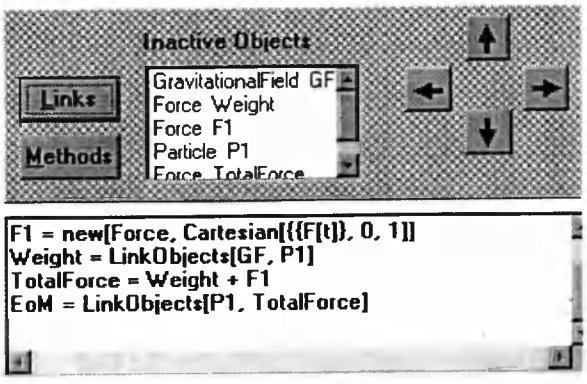




Force TotalForce

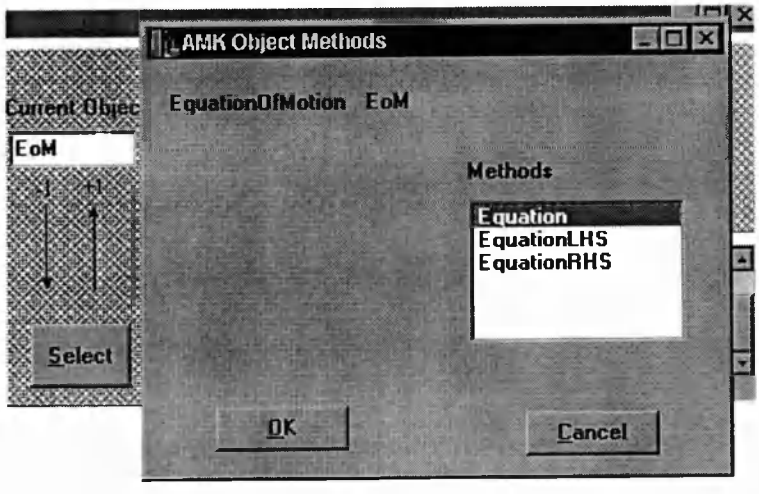
Figure 8.3c

The next stage in the modelling process is to generate the *EquationOfMotion* object by linking P1 and TotalForce (Figure 8.3d). Ellipses represent *EquationOfMotion* objects in IF2: they are easily distinguishable from other objects. Figure 8.3e shows the Methods screen, which adds the line `Equation[EoM]` to the Mathematica script.



EquationOfMotion EoM

Figure 8.3d



EquationOfMotion EoM

Figure 8.3e

The result of this sequence is the following script.

```
P1 = new[Particle, Cartesian[{{y[t],0,1}], {}, t, m]
GF = new[GravitationalField, Cartesian[{{g,0,-1}], {}]
F1 = new[Force, Cartesian[{{F[t],0,1}], {}]
Weight = LinkObjects[P1, GF]
TotalForce = F1 + Weight
EoM = LinkObjects[P1, TotalForce]
Equation[EoM]
```

IF2 does not send this script to Mathematica.[3] Its purpose is to show how an interface can support a modelling cycle.

[3] The result of evaluating this Mathematica input is, correctly, {my''[t] == -gm + F[t]}

8.4.4 Evaluation of IF2

IF2 supports the *Construct→Link→Invoke_Methods* modelling cycle and the *Principle of Adjacency* directly by propagating the model as a diagram is drawn. It is more fundamental than IF1 because the modelling process and 'good practice' (the diagram!) are intimately connected: drawing a diagram generates the model. In doing so, it reinforces fundamental processes within the modelling domain. Specifically these are:

- what elements are in the problem domain;
- what their characteristics are;
- how they relate to each other.

The user needs to supply the first of the above characteristics by creating suitable objects. Object characteristics are encapsulated in the object model, and the user only needs to supply values (symbolic or numeric) for parameters. The user also needs to have an overall strategy for advancing the model, but the details of individual interactions between objects are held in the object model. IF2 only needs to 'know' what the relevant interactions are. Within the constraints of the classes supported by IF2, IF2 can generate Mathematica scripts successfully and quickly. Like IF1, IF2 frees the user from some of the Mathematica syntax, which a necessary component of the AMK object model.

IF2 is capable of development into a generic modelling tool. To do this, a facility for amending existing objects and creating new ones is needed. The exercise of extending AMK to another problem domain in Chapter 7 showed that producing an object model that works is not trivially easy, and needs considerable forethought.

Two further additions (the two bullet points below) would improve IF2.

- The first is to allow 3 or more objects to be linked. For example, two *Forces*, F1 and F2 could be linked with a *Particle* P in the construct `LinkObjects[P, F1+F2]`. The present implementation lends itself to this approach: a multi-select facility, with a suitable means of recognising allowable combinations of objects, could be added.
- The current icon library is limited to generic icons. This causes problems if the object can take many forms. The *Force* class is one of these. Several icons would be useful: an arrow pointing to each of the eight points of the compass would provide

the flexibility to represent 2-D modelling scenarios. Resizeable icons would extend this capability further.

Since the principal ingredient of IF2 is a diagram, it cannot be used if no diagram can be drawn. I address this problem in Section 8.5.

8.5 Modelling scenarios where there is no 'Natural' Diagram

Since it is not possible to use IF2 with no diagram, modelling scenarios where there is no 'natural' diagram present more of a problem. A diagram could be contrived (see the discussion of population dynamics in Chapter 6), but this process can turn into a somewhat inflexible template approach. In principle, more research is necessary to solve this problem, but a generic template could be constructed as below. In this template, the icons bear no relation to the physical objects which they represent. Considering a population dynamics (or, more generally, an input-output) problem, the object model could have a *State* class and a *Change* class (Figures 8.4a and 8.4b).

Class State{UniqueName, InOut, Q[t], t, Info}
represented by:

Figure 8.4a

Class Change{ UniqueName, Modify[Q[t], t], dt}
represented by:

Figure 8.4b

Providing a template to capture a new `Class` instance is relatively simple, but the simplicity is limited to capturing attributes and providing `Get` and `Set` methods for them. The *Modify* method in this example is more specific. In this model, Q and t are the state variables. State.InOut = In represents a *State* object before the effect of linking with *Change* objects. This action encapsulates the birth and death processes. The *Modify*[] method of the *Change* class contains the functional form of the birth or death process and the *Info* attribute contains details of the birth and death processes when InOut = Out. Linking several *Change* objects produces a sum of the overall changes. Linking an object which represents the sum of overall changes with a *State(In)* object then produces a *State(Out)* object, from which a *StateEquation* can be exported. The

model can then proceed, within the context of an IF2 screen, in the following stages (Figures 8.5a to 8.5c).

Stage 1: Construct *State(In)* and *Change(Birth)* objects.

Figure 8.5a

Stage 2: Construct a *Change(Death)* object and link with the *Change(Birth)* object to produce a *Change(Diff)* object.

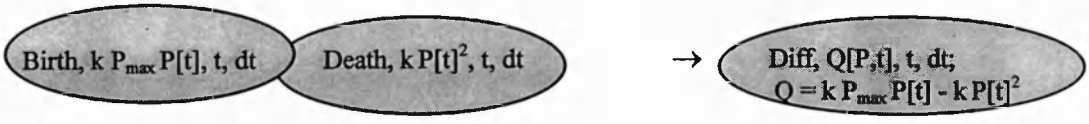


Figure 8.5b

Stage 3: Link the *State(In)* object with the *Change(Diff)* object to produce *State(Out)* object.

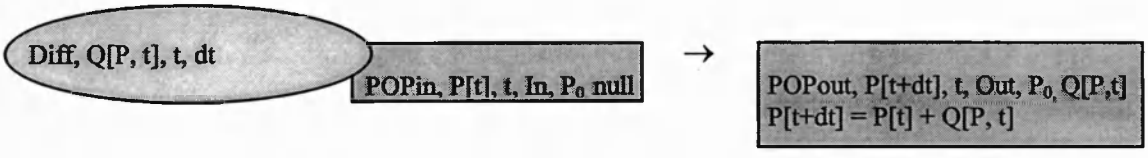


Figure 8.5c

Stage 4: Call POPout.StateEquation to produce the population difference equation $P(t+1) - P(t) = kP(t)\left(P_{\max} - P(t)\right)$.

Chapter 9

Validation

9.0 Abstract

I assess current trends in assessment of mathematical modelling, and this analysis is used to show that there is a lack of research into assessment of model formulation. The emphasis is on more peripheral issues: formation of relations is not emphasised. There is also no objective measure of how good a model is in terms of its complexity or simplicity. How a model may be assessed in terms of relations between its variables remains an open question. An analysis of errors made in 61 modelling projects provides a classification of error types. It then shows that 35% of those errors are in relationship formulation. Of these 35%, 19% were unable to start the process, and a further 71% made material errors in relationship formulation. These statistics justify the case for developing the modelling concepts and software in this thesis. The review in Chapter 3 showed that any direct comparison of a computer approach to mathematics with a 'traditional' approach is flawed because they are fundamentally different. In order to validate the O-O approach, an assessment is given, for each modelling project, of how the O-O approach would have been beneficial in alleviating difficulties. Subjective judgments have to be made in this assessment. In order to reduce the element of subjectivity, I require the O-O method to be a positive improvement, and penalise it when it presents difficulties. The details are recorded in an extensive Appendix. Scores representing the 'added value' of the O-O technique are then still statistically significant. There is a variety of statistical tests, and I draw attention to conditions under which they are applicable.

9.1 Research on Model Validation

This section analyses problems of model validation, and concludes that little work has been done on processes central to the modelling cycle: identification of features and formulation of relations.

Jablonka (Jablonka 97), makes the valid point that model validation is not merely a simple process of comparing numerical results from the model with experimental data. She considers that evaluation of the effectiveness and usefulness of the model depends on collection of suitable data and specialist knowledge about the problem domain. The latter implies that the mathematical sophistication of the model is important, but this view is inconsistent with Jablonka's. Jablonka considers three principal steps in evaluating the effectiveness of the model, and these are related to steps in a generic modelling cycle.

The first is to investigate the mathematical properties of the solution: the solution must behave in a functionally correct way. She does not say that in order to do this it is necessary to produce an algebraic model. The second stage is to use data which have been assessed for quality and accuracy, but she does not say how this may be done. The third step consists of checking results and assumptions. Comparing the results predicted by the model with reality should check for the possibility that they were obtained by chance. There is no clear way to do this without statistical trials. Similarly, although she says that alternative models should be checked, it may not be possible or feasible to produce such a model. Jablonka also provides two warnings. The first is that it may be very difficult to validate any model which is based on non-axiomatic principles. I suggest that the essence of solving this problem is to find a well-defined and stable empirical relationship between any variables concerned. The second warning is not to extrapolate too far, which is quite sensible.

Jablonka does not discuss several important points. These are how to:

1. validate features, in number and in importance;
2. validate assumptions for reasonableness and effectiveness;
3. validate relationships between features in the model;
4. measure the effectiveness of revising a model by changing assumptions, features and relationships between variables and parameters;
5. measure completeness of a model.

Hanna (Hanna 89) approaches model validation by considering inputs and outputs, which can seek ill-conditioning in the model. His comments are only applicable to data models. Hanna suggests a preliminary inspection to detect outliers, potential error margins and pathological cases. This process serves to 'clean' data, and to understand the variables and relationships concerned. He follows this by hypothesising relationships between variables, estimating numerical values for the parameters of the model and applying appropriate statistical tests to determine goodness of fit. This can easily result in over-fitting. He suggests a potentially useful process for model revision: "isolation of the important variables". To do this, variables are gradually added to a simple model and the effect of changes in numerical values of inputs on the outputs are examined. In parallel, variables are gradually removed from a complicated model and the same input/output analysis is performed. The processes stop when there is no further statistically significant change. This method looks promising but it pre-supposes that it is possible to add and subtract variables in this way, and that data are available. An algebraic version of this process would be more useful.

These papers shed some light on the validation process, but more work is needed in order to clarify issues arising. There are particular unresolved issues of assessing validity for models which do not have an axiomatic basis, and in finding an optimal complexity of a model.

9.2 Current Issues in Assessment

The aim of this section is to demonstrate that the thrust of current research activity into assessment of modelling (in particular, using a *generic* modelling cycle) is not directed to problems of finding and relating variables. The effect of this is to exacerbate problems such as not being able to find relevant variables, not relating features to symbols in the model, not being able to generate relations, and generating incorrect relations.

Current research in assessment of modelling exercises indicates that finding and relating features is not an area that has attracted attention. Battye and Challis (Battye 97) propose learning outcomes for mathematical modelling, but they correspond broadly to the 7-point generic modelling cycle. The "Abstract and represent relevant features" and

"Identify the mathematical nature of the problem" categories are relevant to finding and relating features. They do not analyse these points further. This paper concentrates on many general issues in assessment of modelling, as well as mathematical issues. These include subject knowledge, comprehension, ability to apply techniques, presentation of results and placing the problem in a wider context. Similarly, Hirstein (Hirstein 95) says "mathematize" and "use mathematics to solve". These criteria are too general, as are the descriptions of degrees of mathematisation etc., and are open to differing interpretations. They also provide no guidance as to how they should be applied. Houston (Houston 97) discusses assessment of poster presentations. He gets further in requiring assessment of factors which are more relevant to finding and relating features, but assessment criteria are often ill-defined:

1. Consider all relevant facts and information ("All" and "relevant" are ill-defined.)
2. Explain critical assumptions (They should demonstrably simplify features.)
3. Explain critical relations. (This does not say how a relation may be found.)
4. Make sensible use of personal knowledge and experience. (This is subjective and age-dependent.)

Haines and Izard (Haines 95) have a more objective analysis of descriptors for each assessment category. This is a process of stepwise refinement in which assessors and students agree, after a period of convergence, on appropriate assessment criteria. This approach is interesting in that it contains, effectively, in-sample and out-of-sample criterion testing. This process still does not address minute detail of the modelling cycle. Examples of the descriptors are: *M2 Identify features*, *M4 State variables*, *M5 Explore relations*. Although these criteria have been found in a more objective way, how to apply them is still subjective. In particular, the term "explore" can result in much effort but little result.

The Open University's assessment scheme (MST204Project, up to 1993) is more explicit in quantifying the relative importance of the parts of the generic modelling cycle. In the project outline 15% of marks are allocated to defining the problem, 20% to discussion of data, 30% to Assumptions/Features and 35% to formulating the model. The result is that it is easy to earn the first 65% of marks, but the 35% for model formulation is harder, and a 'reasonable' score can mask an ill-formulated model. At the model presentation stage, only 40% of marks are allocated to model formulation and

solution. The thrust of this assessment scheme is to examine the overall modelling cycle, not the complexity or integrity of individual stages.

Current activity in modelling concentrates on broader themes (MathSkills 98). Two out of the seven themes discussed are relevant to model formulation: *Electronic Teaching Innovations* (Theme 3) and *Using Technology, Multi-media and distance learning techniques* (Theme 7). The first of these comprises discussion of topics such as computer algebra systems in modelling, critical reviews and comprehension tests. The second comprises links to specific WWW sites where details of (mostly dedicated) software (e.g. MathWise modules) may be found. Activities reported on the MathSkills Web site are consistent with the assessment aims discussed in this section: problem identification, model formulation, validation and presentation of results. They address broad issues and are not concerned with problems of model formulation.

9.3 Empirical Evidence for Problems with Relationship Formulation

Potari's (Potari 93) discussion is significant because he provides evidence of problems with relationship formulation which mirror the empirical findings of this thesis. He reports results for a modelling project on travel costs, for which he aims to assess processes for model building. Although Potari's study is not extensive, it is big enough to draw some conclusions, even though lack of data inhibits statistical analysis. Out of 19 groups, 7 "did not appear to have any mathematical treatment of the data - no mathematical concepts, skills or methods were used." Thus 37% of groups were unable to formulate the problem. In the other 12 projects, there were attempts to explore variation, but few successes in relating variables. This compares with near equivalent figures in this chapter: 35% (59 out of 167) were unable to formulate relationships and 19% (11 out of 59) had trouble with formulating relationships. There were therefore few justifiable conclusions, and graphical descriptions were common. Potari does not suggest ways to improve this situation other than to use more group work (which appears to have failed here), and more open evaluation and discussion. This study is limited in the range of modelling situations, the number of students concerned and their ability. It does expose an extreme form of the problem tackled in this thesis, to the extent that none of Potari's students produced a successful model.

9.4 Validation Requirements

A fundamental problem noted in Chapter 3 is that validation of methods to aid the modelling process using experimental and control groups is virtually impossible because it is extremely difficult to isolate one factor only in a controlled trial. The factors that need to be isolated are:

1. Embedding mathematical modelling in a computer environment.
2. Applying an O-O method to mathematical modelling.
3. The computer implementation of the O-O method.
4. An O-O design methodology applied to mathematical modelling.

9.5 Validation Strategy

The validation strategy of this thesis, which avoids problems encountered in trials, is to assess whether or not an O-O methodology would have been beneficial, had it been available. Using modelling examples from (MST204 89), covering the years 1990-92 and a range of 'good' to 'poor' scripts (as judged by scores out of 20), student errors were categorised according to the classification below. I prepared outline O-O solutions for the same problems, and made an assessment of what could have been achieved had the O-O method been available. This is prone to subjectivity, and in order to be as objective as possible, I fixed established test criteria in advance, and the samples were tested against the O-O solution according to these criteria. Necessary assumptions are that documentation of methods in the class library for the problem exists, and that the student can use the O-O method. The comparison criteria mirror the main categories in the next section. Of these, it was rarely possible to test correctness of the solution because the aim was to indicate the solution method, not to produce a full solution.

A scoring system provides a crude statistical measure of the effectiveness of the O-O techniques. For each sample script and for each of the above test criteria, scores were recorded according to the rules:

- if the O-O technique could have helped, score = +1;
- if the O-O technique would have hindered, score = -1;
- if the O-O technique would have had a neutral effect, score = 0;

To differentiate between 'important' and 'marginal' criteria would have introduced further subjective elements in assessing their relative importance. Hence I did not do this. However, I would classify *Correct model formulation* and *Appropriate features and symbols* as more important than the others (these are the target for this thesis), and *Appropriate domain knowledge required* as less important, because this is less under the control of the student or the teacher.

9.6 A Classification of Problems encountered in Mathematical Modelling

The following section catalogues problems encountered in applying a generic modelling methodology. It was derived from a subset of the scripts. This classification broadly follows steps in this modelling cycle.

9.6.1 Inappropriate use of technique

1. *Wrong technique used.* The most common example of an inappropriate solution technique was using calculus to optimise with respect to a discrete variable. This can be justified (non-rigourously), but this is rarely done. Another example is to attempt to solve an ill-conditioned system of linear equations without using partial pivoting.

9.6.2 Problems with model formulation

1. *Unable to start.* This is a worst case scenario. The modeller is able to state the problem in words, list features, assumptions and variables, but is then unable to construct a model based on what has been done so far.
2. *Missing model.* Another extreme case is where a model and a solution are presented together, but with no justification for the solution. There is confusion between assumptions and a modelling objective. This occurred in a "sleeping policemen" script, and more often in "firebreak" scripts.
3. *Mathematically incorrect model.* A model contains relations which are not justified or are demonstrably wrong (often on dimensional grounds).

4. *Misunderstanding of model fundamentals.* These indicate that the student found it difficult to link relevant features. It usually related to a linear approximation instead of a differential in heating/cooling problems.
5. *Unclear way of relating features.* There is no justification for equations (or what is stated is not understandable). This is a particular problem in a contexts with no axiomatic basis, such as the "firebreak" problem.
6. *Inappropriate use of theory.* Theory is applied, but there are indications, from omissions, irrelevancies and errors, that the student has not understood it. The "kitchen scale" problem provides examples.

9.6.3 Problems with compilation and use of features and symbols

1. *The features list is very long.* This can make it difficult to reduce to a manageable list.
2. *The reduced features list is either too small or too large.* A lengthy discussion of features which are not to the point can be indicative of either insufficient understanding of the problem domain or how variables in the features list are related. If the features list is too small it is likely that material features are missing.
3. *Features are modelled but are not included in the features list.* The model is then divorced from the features list. This was common in heating/cooling problems.
4. *Symbols are ambiguous or incorrect.* Symbols in the model do always correspond to items in the features list, and are sometimes used incorrectly. This was also common in heating/cooling problems, where there were many similar symbols.

9.6.4 Problems with compilation and use of assumptions

1. *Assumptions are missing but used.* Such assumptions often remain hidden. For example, it is not strictly necessary to state that a car is modelled as a particle. Validation of the model may fail because such an assumption may not be justified.
2. *An assumption is contradicted in the model.* This error is possible if the assumption has no explicit effect on the model formulation, or if the model formulation is divorced from any statement of the assumptions.
3. *Irrelevant assumptions are stated.* An unused assumption will not affect the model, but indicates unclear thinking.

4. *Data and assumptions are confused.* Data introduced at an early stage in the modelling cycle can limit the possibility of exploring functional variation, and can hide singular cases. This error, in effect, masks an unstated assumption.

9.6.5 Problems with model solving

1. *Wrong solution method.* An inappropriate solution method may be used because of conceptual misunderstanding. Calculus errors with graphs is one category.
2. *Errors in solution.* It is easy enough to make routine algebraic and arithmetic errors, even when being careful! Using software produces different errors: programming, procedural and conceptual.

9.6.6 Data problems

Only the first two of the following are fundamental to the modelling cycle but for completeness, others are listed.

1. *Wrong data used.* Numeric values for parameters and variables must be reasonable.
2. *Inappropriate Fitting.* A model, fitted to empirical data, should not, in principle, be over-complicated. A linear fit is often all that is needed, and more complicated fits may not project onto subsequent data well.
3. *Over-fitting.* Using all data available does not allow a data model to be validated. It is better to use some of the data for in-sample fitting, and use the values of parameters obtained to calculate out-of-sample ordinates for comparison with the rest of the sample data.
4. *Cannot find data for validation.* This makes validation harder.

9.6.7 Problems with knowledge of the problem domain

1. *Insufficient knowledge base.* Finding relevant features and selecting the most important ones is easy for someone who is familiar with the problem domain. Davis (Davis 85) supports this view, arguing that modellers use old representations for models to make new ones. These permit efficient use of relevant heuristics,

procedures and knowledge. The study of Boekaerts, Seegers and Vermeer (Boekaerts 95) supports this view from a cognitive point of view.

2. *Domain expertise necessary.* Usher and Henderson (Usher 97) give an example of how input from domain experts in an oncology model produced a relatively sophisticated model: a simpler one would have been inadequate.

9.7 Extent of modelling problems

The purpose of this section is to categorise and quantify the extent of the errors identified in the 61 sample scripts (Table 9.1). The categories below are the broad categories in Appendix 9S. They reflect not only the predominance of the first three categories, but also the number of subdivisions within each category (there are more in the first three). The methodology and software developed in this thesis aims to address exactly these first three categories, although it does have an impact on the others.

Year	*1990*	*1991*	*1992*	*All*	*% All*
Total students	20	22	19	61	
Error category					
Features list	16	15	10	41	24.6
Relation formation	20	26	13	59	35.3
Objective/Method of Solution	15	17	13	45	26.9
Assumptions	6	9	1	16	9.6
Data	1	1	0	2	1.2
Other	3	1	0	4	2.4

Table 9.1

The category 'Other' represents cases where some aspect of the model presented was seriously at fault. These mainly relate to use of diagrams. There are cases in Appendix 9S which are instances where a diagram would have helped with an O-O formulation. These are not included in the above error analysis. Table 10.1 establishes that there is a *prima facie* case for investigating problems with relation formulation in particular.

The extent of problems at the 'equation formation' stage is much greater, and much more important, than problems elsewhere in the modelling cycle. Table 9.2 analyses subdivisions in this category. It shows that in a significant number of scripts, students

were unable to form equations at all. Most of these had had reasonable success in earlier stages of the modelling cycle.

Relationship formulation subdivision	Total	%
R1: Cannot start	11	19
R2: Material modelling error	42	71
R3: Features error	5	8
R4: Excessive complication	1	2

Table 9.2

These results show that the predominant problem is when a modeller makes some attempt to relate features, but makes a serious error in the process. The O-O solution addresses this problem by exporting class methods that describe relations between objects. For example, when a *Particle* object interacts with an *InclinedPlane* object, the result is a *Force* object, the components of which are correct. A common error in this situation is for the 'normal reaction' component of this force to be vertical. Error R1 is potentially more serious, because the modeller is unable to start to formulate a model. These cases reflect a complete failure in a generic modelling cycle. The O-O solution addresses this problem by ensuring that a model develops in parallel with drawing a suitable diagram, aided by heuristics and axioms.

9.8 Test Criteria and results

Appendix 9S1 contains a summary of the modelling errors in the sections above. These relate to the principal areas where an O-O methodology could be useful. For each test criterion in each script in the test sample, I assessed whether or not application of an O-O methodology could have been of use. If so, a score of +1 was awarded. The total score for each script was then noted. Allocating these scores is necessarily subjective, and in order to offset the effects of any subjectivity, the following steps were taken.

1. In cases where an O-O approach would not have added to the modelling exercise, a score of -1 was allocated, thus positively penalising the O-O approach.
2. In cases where a score of +1 was allocated, a justification of why that score was allocated was required.

3. Scripts were assessed against a prototype O-O solution and methodology, in which some alternatives were considered.

Appendix 9P contains O-O outline solutions for the problems in the sample scripts. Detailed scores are in Appendix 9S. Cases where no score was allocated are not listed. Appendix 9S2 contains a summary of scores per script, and the statistical analysis is mainly based on this.

9.9 Statistical analysis

The statistical analysis of the results consists of two principal strands: a crude non-parametric test and a more sophisticated one based on the Normal distribution. The purpose of the former is to measure the usefulness the O-O technique by a binary *improved/unimproved* measure, abandoning a numerical score. This is to mask bias in reinforcing factors which could have inflated scores. *z*-, *t*- and *Sign* tests then provide more sophisticated measures, but some risk including bias. The 61 items in the sample is large, and covers three student cohorts (1990-92), with a full range of TMA marks awarded and a reasonable spread over a range of modelling problems. They are all the scripts allocated to me in those years. I assume that this sample is typical of scripts taken at random from a background population of submitted TMA04s. A pre-test is inappropriate for this since there is no possibility of testing a script twice, and there is no before-and-after situation.

9.9.1 Justification for approximating discrete data by a Normal distribution

The O-O scores I allocated to these scripts means that they cannot be Normally distributed: they are discrete and have a limited range in practice (-4..7 in this case). The frequency distribution in Figure 9.1a confirms this.[1]

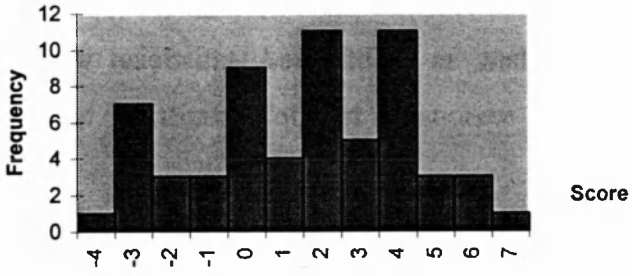


Figure 9.1a

Statistical tests which should only be applied if the background distribution is Normal are widespread in the comparisons described in Chapter 3 (e.g. Smith 94 and Mayes 97 for ANOVA and *t*-tests). However, none of the authors cited in Chapter 3 tested the normality of their data. Consequently, their results require further justification. In order to apply a statistical test, it is only necessary to test whether or not the data could have come from a Normal distribution, ignoring the known origin. I apply several such tests in the following paragraph. If the data pass these tests, statistical techniques which use tests that depend on a Normal distribution can be used.

Chatfield (Chatfield 88) makes several points which are appropriate for validating a Normal model. First, a detailed test for Normality is not always necessary as gross departures from Normality are often 'obvious by eye'. This is so here, as Figure 9.1a shows. Despite this, the raw data pass two stringent tests. They pass a Lilliefors Normality test (Dudewicz 88) at the 99% confidence level (although not at the 95%), and a Bowman-Shenton test (Wetherill 86) at the 95% confidence level. In addition, they pass a less stringent test (Hair 84), which depends on measuring the skewness of the distribution, at the 95% confidence level.

Secondly, a *t*-test is robust with respect to 'moderate departures from Normality'. In this case, what 'moderate' means depends on how the data are viewed. Krzanowski

[1] If the categories are grouped in pairs [(-4,-3), (-2,-1), ..., (6,7)] the profile of Figure 9.1a looks much more like a Normal distribution, but skewed slightly to the left.

(Krzanowski 98) suggests looking at the number of points on a Normal probability plot for which $\Phi^{-1}(x)$ lies outside the range (-1.96, 1.96). No more than 5% should be in this category. Only 2 raw data points (3.3%) actually fall in this category. Fitting a Normal distribution with the same mean and standard deviation as the experimental data was unsuccessful, and failed a χ^2 test for goodness of fit. Grouping the data improves matters, but insufficiently. The considerations in this paragraph show that, despite the clear non-Normality of the raw data, an uninformed statistician would not find the situation so clear. It is therefore reasonable to apply statistical tests that require a Normal distribution, albeit tentatively.

The same conclusion also follows from visual analyses of the data. A Normal Probability plot, (Daly 95), (with the ordered score on the vertical axis, and the ordinates $\Phi^{-1}(i/62)$, i = 1..61 on the horizontal axis) indicates that a Normal fit is not totally unacceptable. It seems reasonable to draw a straight line through the points in Figure 9.1b, despite the obvious score groupings.

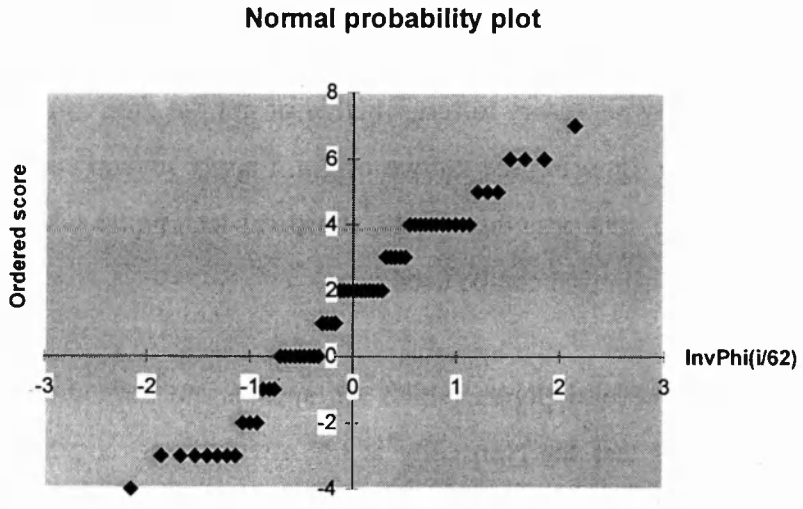


Figure 9.1b

Collett (Collett 91) suggests considering normality of residuals as an indicator of model adequacy. Following Collett's method, Figure 9.1c shows a plot of $X = \Phi^{-1}\left(\frac{i+n-\frac{1}{8}}{2n+\frac{1}{2}}\right)$ $i = 1..12$, $n = 12$ (for the 12 datapoints -4 to $+7$), where X is the expected value of the half-Normal order statistic, against the ordered absolute residuals Y (deviations of observed frequencies from ordinates on a Normal distribution

with the same mean and standard deviation as the data in Figure 9.1a). The points are enclosed between 2 lines: the simulated envelope of (Atkinson 81). These are half-Normal ordinates fitted to the minimum and maximum frequency in each category for a 19-run simulation of the distribution in Figure 9.1a. Three points lie outside the envelope, with probability 0.054, and systematic deviation from the mid-point line of the envelope indicates over-fitting, and hence an inappropriate model. Against this, the plotted points lie tolerably near the mean line of the envelope. These plots are subjective in construction and interpretation, and the simulated envelope is unclear near the origin. However, this analysis does indicate that tests based on the Normal distribution must be used with caution.

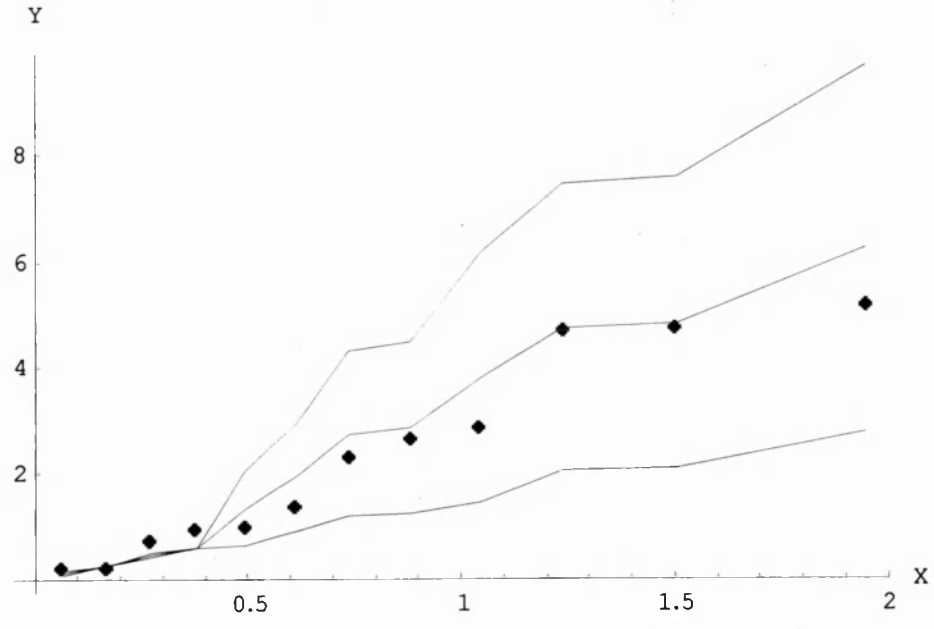


Figure 9.1c

9.9.2 Sign test

The sign test makes no assumptions about the background distribution and lessens the effect of inflated scores because it is a test for the population median. It is not sensitive to any finer points used to derive the scores, and the only point at issue is whether or not the overall O-O analysis could improve a script. A sign test is therefore a useful test here because the *t*-test is not entirely satisfactory.

Let S be the number of scripts (out of 61) that could be improved by an O-O analysis. The null and alternative hypotheses are:

NH: $p = \frac{1}{2}$;

AH: $p > \frac{1}{2}$ (since the O-O technique should improve matters).

A summary of the results in Appendix 9S4 is:

S(+) = 38; S(-) = 14; S(0) = 9.

Zero results in a Sign test are usually discounted, leaving a sample of size 52 in this case. The sample size is large enough to justify approximating the Binomial(52, ½) distribution by a N(52×½, 52×½×½) distribution. With this approximation, calculating P(S≥38) results in z = 3.19. [2] This is significant at the 0.1% level, and the null hypothesis can be rejected. In order to be hard on the O-O method, the (-) and (0) scores are combined. This is a more stringent test. It tests the number of improved results against the number of non-improved results, and thereby requires that the O-O method does provide a positive improvement. Approximating the Binomial(61,½) distribution by a N(61×½, 61×½×½) distribution and calculating P(S≥38) results in z = 1.79. [3] The null hypothesis can then still be rejected at the 5% significance level.

[2] This calculation uses a continuity correction. If the continuity correction is omitted, the value of the test statistic is 3.32, which is also significant at 5%.

[3] This calculation uses a continuity correction. If the continuity correction is omitted, the value of the test statistic is 1.92, which is also significant at 5%.

9.9.3 *t*-test and *z*-test

The sample size (61) is sufficiently large to allow a *z*-statistic based on the Normal distribution to be used. A *t*-test is still more appropriate since the standard deviation is not known in advance, and, more significantly, it is more robust when applied to data for which the assumption of a Normal background distribution is dubious. Despite this reservation, I concluded in Section 9.9.1 that it is not totally inappropriate to use a *t*-test. Appendix 9S3 contains the computational details. The null and alternative hypotheses are:

NH: mean population test score = 0;
AH: mean population test score > 0.

These hypotheses test the claim that the O-O analysis can improve the submitted analysis (AH) against the counter-claim that the O-O analysis has no effect (NH).

The results, $t = z = 4.21$, are both significant at the 0.1% level. The conclusion in rejecting the null hypothesis is that the O-O analysis is likely to have a beneficial effect. This conclusion is subject to constraints:

1. there are elements of subjectivity in allocating scores;
2. it is assumed that the student would have been able to make some use of the O-O facilities, had they been available;
3. the O-O component, rather than a more general focus on aspects of the modelling cycle, is responsible for any potential improvement;
4. Normality of the background population is not assured.

The first of these points is covered by the second element of the statistical analysis, below. Another possibility is for an independent assessor to allocate scores to ensure comparability. It is possible to support the second and third of the above points in two ways. First, drawing a diagram (which is possible in most cases considered) generates relationships between features whenever two objects are placed next to each other (the *Principle of Adjacency*). Drawing a suitable diagram is a normal part of modelling practice, so the modelling process can be advanced. Second, the process of defining object methods that describe how two objects interact forces the user to consider which objects can interact and how they do so. The fourth point can be justified in two ways.

First, the Normal Probability plot provides weak justification for approximating the data by Normal distribution. Second, the *t*-test is robust enough to cope with data for which the Normal distribution is a poor fit (Daly 95 gives examples).

9.9.4 Analysis of Fault category R1

"Fault R1" scripts are the ones which contained no attempt to formulate a model. Analysing this error category is important because it measures the potential efficacy of the O-O technique in initiating a model where no model existed previously. Overall, a higher proportion of positive scores were allocated to "non-Fault R1" scripts than to "Fault R1" scripts. This is a reflection of the number of other categories involved and does not measure the effect of the O-O technique on "Fault R1" scripts. Table 9.3 summarises data extracted from Appendix 9S5, and records the overall scores (not just the score for R1 only, which is clearly impossible if category R1 is absent).

	(+)	(-)	(0)	(+) Proportion
Fault R1 scripts	17	9	7	17/33 ~ 0.52
non-Fault R1 scripts	21	5	2	21/28 = 0.75

Table 9.3

The following unconventional analysis attempts to isolate the effect of the O-O technique on "Fault R1 scripts". I argue that 33 out of 61 "Fault R1 scripts" should show no improvement under a null hypothesis that the O-O and traditional treatments are equivalent. The measured proportion was 23 out of 61, which is lower than expected. The summary in Appendix 9S5 isolates the 33 instances where the score for R1 is non-zero and non-null. Of these, the scores R1 = +1 represent instances where the O-O technique can advance the model and the scores R1 = -1 represent instances where the O-O technique cannot advance the model because the result of the O-O analysis has already been done by traditional methods. Let R be the number of scripts (out of 61) that have Score(R1) = -1. From Appendix 9S5:

R(+) = 10; R(-) = 23; R(0) = 0.

The basis of the null hypothesis is that a proportion 33/61 of scripts would have produced the same analysis as the O-O technique in ideal circumstances (i.e. 'perfect

traditional' modelling). The experimental proportion is 23/61. Using a Binomial(61, Π) model for the number of scripts where the model formulation coincides with the O-O formulation:

NH: $\Pi = 33/61$; AH: $\Pi < 33/61$.

Approximating the Binomial(61, 33/61) distribution by a N(61×33/61, 61×33/61×28/61) distribution, calculating P(R≤23) results in z = 2.44. This is significant at the 1% level, and the null hypothesis can be rejected. This provides evidence that the O-O analysis can have a beneficial effect in model formulation.

9.9.5 ANOVA analysis of problem categories

If the 61 data in Appendix 9S are grouped by modelling problem, a cursory glance indicates that the O-O technique is much more successful with certain problems than with others. If this were so, it might indicate that the O-O technique may only be applied successfully to certain categories of problem. Appendix 9S6 shows the Appendix 9S data organised by modelling problem, and the ANOVA analysis below shows that the concern raised is unfounded. Appendix 9S6 also contains details of the ANOVA computations. The test hypotheses are:

NH: $\mu_{PB} = \mu_{CP} = \ldots$ where μ_P is the mean score allocated for problem P,

and $P \in$ (PB,CP,W,G,S,F,K).

AH: Not all the μ_P are equal.

Table 9.4 is the ANOVA table.

Source of variation	DF	Sum sq.	Mean sq.	F
Between samples	6	76.831	12.806	1.81
Within samples	54	382.415	7.082	
Total	60	459.246		

Table 9.4

The value of F obtained does not exceed the critical 5% value F(6,60) = 2.25. Hence the null hypothesis is rejected at the 5% significance level: there is no significant difference between the mean scores allocated to each modelling problem. This conclusion is subject to the assumption that the variances of the scores allocated within each

modelling problem area are the same, and that the background population is Normal. Earlier comments indicate that the latter assumption is dubious, and a supplementary test (Kruskall-Wallis) is needed.

A Kruskall-Wallis test is more appropriate because, being non-parametric, it is a distribution-free equivalent of ANOVA. It is therefore a weaker but more general test (Daly 95). The details in Appendix 9KW follow the method of (Mendenhall 86). In Appendix 9KW, categories W, G and S had to be combined to produce sufficient data in each category for a χ^2-test, leaving 5 categories. The result $H = 7.94$ does not exceed the critical 5%, 4 DF χ^2 value of 9.49. Hence the same null hypothesis as for the ANOVA can be rejected.

9.9.6 Experimental Control

Fortunate circumstances (detailed in this paragraph) allow a statistical comparison of an experimental group with a control group. This allows the O-O technique to be tested more directly against a non-O-O technique. In marking students' scripts, I sometimes supplied outline modelling strategies when the students had failed to produce one, or had a very inadequate one. These outline solutions form the basis of the control group. They were intended to provide ideas for the students, not to 'give' them a model. However, it is still possible to extract important principles from them, and compare these principles with O-O equivalents. There are also only seven of them, covering some of the modelling projects which have O-O solutions in Appendix 9P. Since I produced them, the ideas in them are comparable to my O-O solutions. They can also be regarded as 'good' solutions.

In addition to these outline solutions, I also used a different modelling scenario (a parachute jump: Appendix 9P, Problem PJ) for tutorials. I would produce a 'good' modelling strategy, using and amending ideas from the students. The result was a complete but considerably abbreviated TMA04. My O-O solution for this problem is directly comparable with the non-O-O solution. Appendix 9PJ contains the problem and O-O solution.

Two notable points arise from the development of the PJ model.

1. The model included a reasonable resistive force for vertical motion with an open parachute. Initially there was no resistance term for the horizontal motion. This led to the situation where the horizontal speed of the parachutist on landing was the same as the parachutist's speed on leaving an aircraft (e.g. 150 knots). When this error was realised, a (constant) horizontal resistance term was added. This was probably not a good way to model the total resistance to motion, but was simple and gave a reasonable result. I doubt that the O-O technique could have been beneficial in this circumstance.
2. The minimum height of an aircraft from which a parachutist could jump and land safely had to be supplied by a student with experience of parachuting. It may be possible to encapsulate such domain knowledge in an O-O model, but it would be hard then to apply the model to more general circumstances. Again, the O-O technique is unlikely to be useful.

Since the non-O-O solutions should be 'good', the scores allocated to control scripts should be negative. This reflects the fact that the O-O technique cannot usefully add to already adequate solutions. In general, this is the case. The statistical analysis in Appendix 9C justifies the statistical claim that the O-O technique adds value to the modelling process by comparing the mean score obtained from the 61 experimental scripts with the mean score obtained from the 8 control scripts. The statistical hypotheses are:

NH: $\mu_{expt} = \mu_{control}$;

AH: $\mu_{expt} > \mu_{control}$;

where μ_{expt} is the mean score for the population of scripts produced by students and $\mu_{control}$ is the mean score for the population of scripts produced by me.

A 2-sample *t*-test shows that the value of *t* obtained, 2.98, exceeds the critical 1% value of *t* (2.66) with 67 degrees of freedom. In rejecting the null hypothesis, the conclusion is that the O-O technique does add value, as measured by an improved score in the experimental group. This conclusion must be seen in the light of the reservations about subjectivity of allocating scores and the applicability of a *t*-test for these data.

9.9.7 A Descriptive Analysis of TMA scores

The purpose of this section is to demonstrate that the sample of 61 scripts does not represent an extreme of the ability range. The most objective way of doing this is to consider the original TMA scores for each script, since those scores were originally allocated according to an objective basis. No sophisticated statistical analysis is needed provided that the frequency distribution of the original TMA scores covers the theoretically possible range 0-20 comprehensively. Figure 9.2 shows this. [Summary statistics: mean = 13.66, SD = 3.36, Skewness (3^{rd} moment) = -0.47]

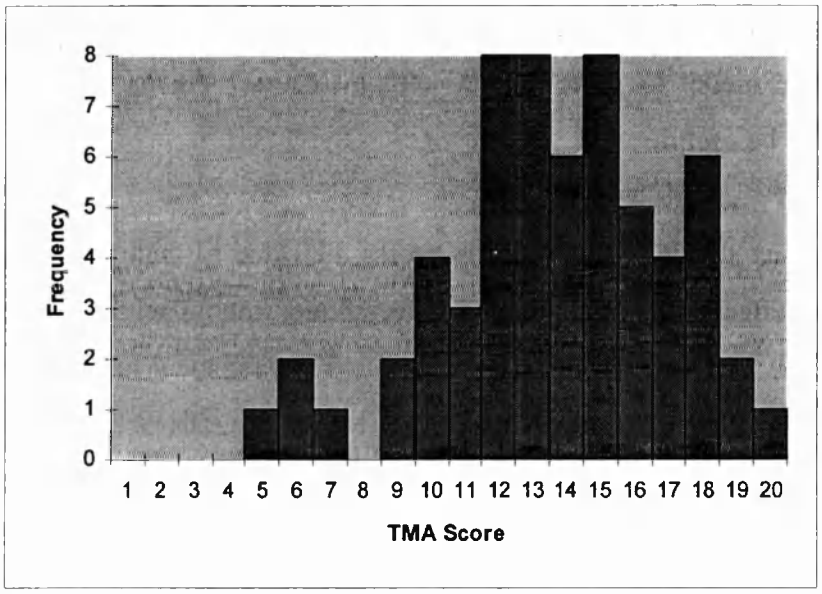


Figure 9.2

The original TMA scores are not a reliable measure of modelling ability for the analysis in this thesis because they reflect activities (e.g. stating the problem, describing data) which are peripheral to the process of relationship generation.

9.9.8 A Comparison of TMA scores with O-O scores

A plausible hypothesis for the O-O analysis is that using O-O techniques benefits weak students more than others. The rationale for this is that stronger students manage well with existing techniques. A simple graphical analysis with minimal computation shows that this conjecture cannot be substantiated to any useful degree. Figure 9.3 is a scatter plot of TMA scores for the 61 sample scripts with corresponding O-O scores (using data from Appendix 9S). A broad inverse correlation is apparent, but it is not striking. The measured correlation coefficient is -0.57 and best linear fit line has equation `OOscore = -0.47*TMAscore + 7.9`. A significance test based on the statistic $Z = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) \sim N\left(\frac{1}{2}\ln\left(\frac{1+\rho_0}{1-\rho_0}\right), \frac{1}{n-3}\right)$, where there are n sample points, r is the measured correlation coefficient and ρ_0 is the population correlation coefficient, shows that a null hypothesis *Correlation coefficient* = ρ_0 (against *Correlation coefficient* $\neq \rho_0$) can be accepted for the approximate range $-0.7 < \rho_0 < -0.4$. This range is in the middle of the negatively correlated region, and provides no significant information. Searching for a statistical model is not of intrinsic value.

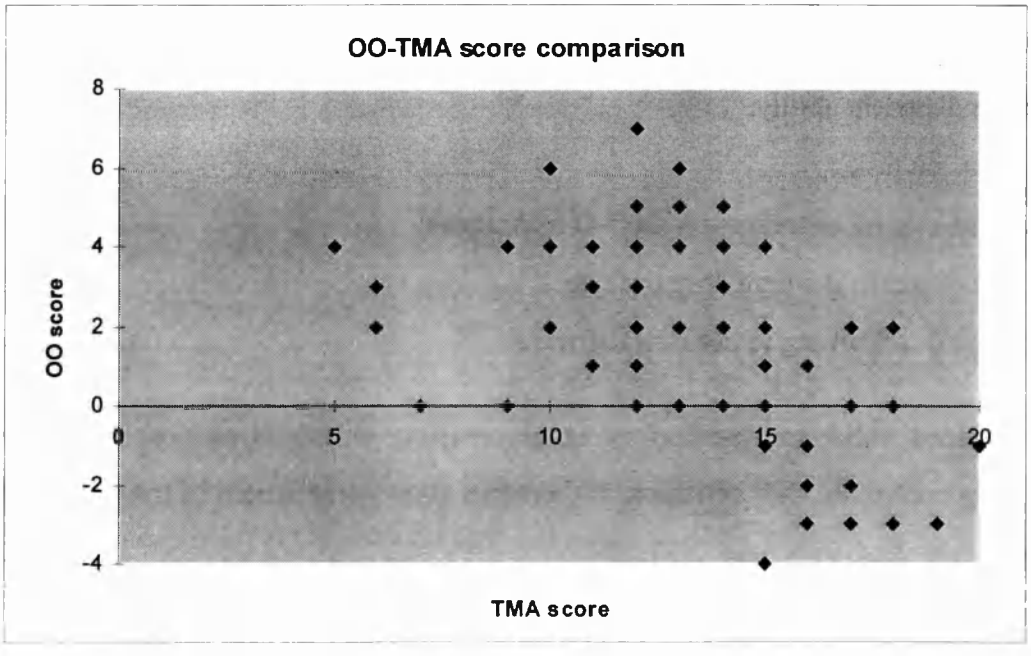


Figure 9.3

9.10 Discussion and Conclusion

In Chapter 3 I showed that the method of evaluating new software by field trials is deficient: it is not possible to measure specific factors. In this chapter I proposed a method of validation which is independent of field trials, and now show two things. First, that the 'value added' method is effective as an assessment technique. Second, that the statistical results demonstrate the effectiveness of the O-O techniques.

The O-O methodology and software developed in this thesis specifically address these components:

- principles for designing an O-O system (the *Diagram-Axiom* methodology);
- a new technique for modelling: the *Construct→Link→Invoke_Methods* cycle;
- a software implementation of the O-O methodology for mathematical modelling (AMK);
- two front-ends for the AMK software.

The 'value added' method and the associated statistical evaluation concentrate on the second and third of the above points. It eliminates the following factors, which are inherent in field trials:

- teacher and student enthusiasm;
- teacher and student ability;
- time spent 'on task';
- the need to acquire experience with O-O techniques;
- small samples due to a high drop out rate;
- the overhead of learning to use new software.

The value added validation method compares existing solutions to problems with corresponding controlled O-O solutions. It therefore allows assessment of the following items:

- feature generation;
- relationship generation;
- correct mathematical analysis;
- appropriate use of diagrams in model generation.

The statistical evaluation procedure specifically targets components of the modelling process which the O-O methodology affects. It is under the control of the assessor and can therefore be applied uniformly to raw data (the sample of 61 modelling scripts). Therefore, the 'value added' technique is an effective alternative to field trials.

Despite the stringent criteria applied when assigning scores to the test data, the significance tests applied show that the O-O technique added significant value to the modelling process. Both non-parametric tests (Section 9.9.2) and parametric tests (Section 9.9.3) produce highly significant results. The significance levels recorded are far from borderline, and stress the power of the O-O technique. In particular, the O-O technique is a significant improvement in the case where the student was unable to formulate a model (Section 9.9.4). I also used ANOVA and Kruskall-Wallis analyses (Section 9.9.5) to show that the type of modelling scenario is not an important factor. The O-O technique is equally applicable to all scenarios considered. There is also evidence that my opinion of the value added was as unbiased as possible. Section 9.9.6 showed that no value was added to my non-O-O solution, as would be expected.

I now consider the stability of the value added technique. In order to investigate the effect of a potential 'second opinion' when allocating scores to the sample scripts, I amended the summary data in the non-parametric tests of Section 9.9.2. Decreasing the number of successes and increasing the number of failures (keeping the number of 'zero' scores fixed at 9) has the effect of simulating an independent, and unfavourable, assessment of the raw data. Table 9.5 shows how non-significant results arise in the Sign Test at common significance levels. These represent perturbations from the actual data: 38 successes and 14 failures.

Significance level	Successes	Failures	No change
1%	35	17	9
5%	32	20	9

Table 9.5

The 1% significance level entry represents a reversal of the decision (overall value added for the modelling problem) on approximately 6% of the sample scripts. The 5% result represents a reversal of this decision on approximately 12% of the sample scripts. These are large deviations from the measured results and indicate that the value added

technique is well-conditioned. The result of a similar stability analysis is not as marked if all of the nine null results are counted as failures. In this circumstance there are 38 successes and 23 failures in the unperturbed data. A non-significant result can be achieved by reducing the number of successes to 37 and increasing the number of failures to 24. This represents a reversal of the decision on 2% of the sample scripts and indicates ill-conditioning in the way scores were allocated to the raw data. However, treating the results in this way is an extreme and unusual view of data.

When testing for ill-conditioning of score allocation with the *t*-test, the effects are similar. For 61 degrees of freedom the critical *t*-value at the 5% significance level is 0.71. It is 0.94 at the 1% significance level. Both are a gross reduction on the measured *t* value, 1.49. This indicates that the decisions on the value added for individual components in modelling problems is well-conditioned.

The statistical claim, that that the O-O technique 'adds value' to the modelling process, must also be justified from a modelling point of view. In particular, the precise circumstances where the O-O technique 'adds value' must be distinguished from cases where any other technique could 'add value' to existing practice. Software could improve 'traditional' modelling two main ways:

1. tie each feature with a variable name and units, thus ensuring that producing features and allocating variables are linked correctly;
2. define and use an objective function.

In addition to these, the O-O technique helps in the following ways, which cannot be provided by traditional means.

1. It stresses which elements are in the problem domain. This provides prototypes for modelling real situations and thereby proposes assumptions about these real situations.
2. A modelling strategy, implemented by constructing diagrams, explicitly addresses problems of relating elements in the problem domain.
3. Objects can easily be amended using overloaded methods, which provides a more flexible and consistent modelling tool.
4. It allows for computerisation and automation in an ordered way.

Chapter 10

Conclusion and Further Research

10.0 Abstract

This chapter first summarises the principal achievements in this thesis, which all relate to the problem of finding and relating features in a model. The *Diagram-Axiom* methodology allows O-O techniques to be applied to small-scale mathematical modelling, the *Construct→Link→Invoke_Methods* modelling cycle provides a way of relating features, and the software implementation permits automation of the process. There is a short critique of this thesis. This focuses on the extent of the "generating and relating features" problem, its solution strategy, and the 'value added' validation method as a viable alternative to field trials. Opportunities for further research arise. The main ones are to structure the *Diagram-Axiom* methodology in an algorithmic form, to find quick and easy ways to construct class hierarchies for new modelling contexts, and to improve and evaluate the icon-driven interface of Chapter 8. I suggest two further alternatives to the O-O methodology of this thesis, both using concepts stressed here. The first is a reduced version of the class hierarchy, based on a list data structure. The second is to structure the generic modelling cycle with a spreadsheet, linked to a computer algebra engine. This has advantages in reinforcing the *Diagram-Axiom* methodology, reinforcing the *Construct→Link→Invoke_Methods* cycle and in providing a semi-algorithmic way to structure a generic modelling cycle.

10.1 Achievements in this thesis

The aim of this thesis is to solve the problem of how to find features in a model and how to form relations between them. The result is an equation of state for the system, and a method for computerising the process. This work is novel in that it casts the problem into object-oriented terms in an environment that supports symbolic computation. This is a radical departure from established modelling procedures, and has not been attempted before. Only two research projects contain ideas which come anywhere near to the ideas in this thesis. Both were mentioned in Chapter 7. Dubisch (Dubisch 90) programmed a Mathematica 'toolkit' for solving problems in Newtonian mechanics. Dubisch's software contains one of the essential elements of AMK: the algebra engine. However, it relied on templates, and is therefore not as flexible as AMK. I solved the flexibility problem by using O-O principles. Viklund and Fritzson (Viklund 92) used a Mathematica O-O environment to do finite element analysis. Although the Viklund-Fritzson software has an O-O basis and a front-end, it uses symbolic computation mainly for rapid prototyping and to make coordinate transformations easier to program. A code generator produces C++ code, which performs the most significant computations. Using compiled code is necessary in computationally intensive numerical techniques such as the finite element method: interpreted code alone would be too slow. In contrast, the O-O environment and symbolic computation are fully integrated in AMK. Neither Dubisch nor Viklund and Fritzson provide a rationale for their software systems: they exist as modelling tools with applications, but do not address any particular difficulty in modelling. I have identified particular problems in modelling (and state them at the start of this section), and direct the software to solving these problems. Furthermore, I support the AMK software with a modelling methodology (*Construct→Link→Invoke_Methods*) and an appropriate O-O design methodology (*Diagram-Axiom*).

Using object-oriented techniques, it is possible to construct objects which represent elements in the problem domain. Linking them results in further objects which are needed to advance the model such that an *EquationOfState* object results, from which an *Equation* method can be called. The Newtonian particle mechanics problem domain provides a good environment for a detailed discussion of the problems that can occur in a modelling environment, particularly as it is a well-understood axiomatic system. A

computer algebra engine is central to this work, as symbolic manipulation is an essential component of mathematical modelling.

Developing the O-O modelling environment required an O-O methodology which is suitable for the contexts considered in this thesis. Established O-O design techniques are more suitable for large software projects, but not for small-scale mathematical modelling. The result is the *Diagram-Axiom* methodology, which incorporates the *Principle of Adjacency*. Part of the software developed is designed to constrain the user to follow the methodology rigorously. This ensures that the user focuses attention on two key aspects of the modelling process: identifying features and relating them.

This thesis is also significant in that it establishes that there are problems with feature identification and relationship generation in modelling, and provides solutions. Furthermore, it does so using a sample which is large enough to be representative of modelling in general. Potari's study (Potari 93) is the only work prior to this thesis that formally identifies these problems, but his study has severe limitations. His sample was small and biased towards lower ability students, his evidence for problems in relationship generation was limited, and he did not propose adequate solutions. Most references to difficulties in modelling are much more general: modelling is "hard". My sample is much larger, and I use techniques which are as objective as possible to analyse the data. In principle, a much larger sample would be preferable, but this is difficult unless more original source material can be found.[1]

A review of quantitative studies in teaching and learning using computer algebra software identified significant problems in validating controlled trials for teaching with computer algebra systems. This review is the most comprehensive account to date of the effectiveness of using computer algebra software for learning mathematics. I exposed a serious deficiency when validating the use of computer algebra software: it is not possible to isolate the effect of the software alone. This problem has not been stressed before. Only Mayes (in Mayes 97) hints at it, but even he does not discuss the implications of this problem for statistical analyses of field trials. I have developed an alternative ('value added') validation technique which can isolate precise ways in which to improve the modelling process. Choosing appropriate statistical techniques, I showed

[1] Such material is not available from the Open University.

that the O-O ideas developed in this thesis have a beneficial effect on modelling which is statistically significant, subject to constraints which have been mentioned previously.

A further aim in developing the software for this thesis was to find an application of computer algebra software which would not be possible without the algebra engine. In AMK the algebra engine provides the means to computerise symbolic modelling and to do operations in calculus. Without the symbolic aspect in a model its functional behaviour is not apparent. Few other computer algebra applications are impossible with computer algebra. (Chandler 97A) is an example of an application that is. The essential component in Chandler's paper is that the program formulates rules as it runs. The result is that outputs from evaluating symbolic expressions are not determinable from the form of the inputs. Coding using a procedural language (e.g. C++) cannot therefore account for all possible outputs.

The following short section therefore contains a minimal summary of the contribution to knowledge presented in this thesis.

Current mathematical modelling methodologies do not address the problem of how relations are formed between variables and parameters, for which there is little heuristic or theoretical research. This thesis proposes a rigorous methodology to do this, supported by computer algebra software within an object-oriented environment, with criterion-referenced validation based on empirical evidence.

10.2 Some lessons learned

In this section I indicate areas of this research which I was not able to develop further, and give reasons for this.

10.2.1 Extent of this thesis

In this thesis I include many components: computer algebra, O-O concepts, O-O design, mathematical modelling, evaluation of learning environments, statistical evaluation, and user interface software. This has caused difficulties in focussing the work to define a precise problem. The uses of computer algebra software are wide-ranging, and it was often tempting to digress. Similarly, computer algebra (and other!) programming is a significant problem in its own right, and is very time-consuming. An alternative would have been to concentrate on fewer components. I list three possibilities below.

- Later in this chapter I suggest a way of reducing the number of components in the thesis. This involves using a spreadsheet to improve the generic modelling cycle. Although computer algebra is not strictly necessary, the impact of the research would then be weaker because symbolic manipulation is a necessary component of mathematical modelling. I regarded the computer algebra component as so important for the research in this thesis that to omit it would have been a serious weakness.
- Alternatively, I could have concentrated on a simpler use for computer algebra. An example is integrating computer algebra into the model building process. A common current strategy is to construct the modelling on paper, and use a CAS to do any computations that arise. This would eliminate the automation aspect of this thesis. Since automating the modelling process is central to this thesis, to omit this aspect would have altered the aim of the research significantly.
- A third limitation would have been to restrict any discussion of teaching and learning using a CAS to modelling issues only. This omission would not have diminished the thrust of the research, but would have caused problems in validating the results. The 'value added' validation method arose from finding deficiencies in quantitative studies of the use of computer algebra in teaching and learning. With hindsight, it is likely that I would not have discovered this from a review of CASs in modelling only.

10.2.2 Non-productive investigations

In attempting to evaluate the effectiveness of a generic model as it develops, I was unable to find a suitable objective function with which to gauge the progress of the modelling process. I considered allocating the equivalent of a probability density function to each component in a generic modelling cycle. Thus, features, assumptions, data, variables and relationships would be represented by distinct 'shape functions', which I intended to be characteristic of what they represent. I did not find a way to characterise them, or to combine them in a way that provided a measure of completeness for the model. Since this investigation was not central to the work in this thesis, a lack of progress with it was not serious.

I was also unable to find ways of programming a CAS to do proofs convincingly. The example of the Gibbs phenomenon appears to show that proofs involving manipulation of infinite series cannot be automated. Work in this area is outside the scope of this thesis and is essentially an extensive programming task.

A particular problem when developing the AMK class hierarchy was to implement the *ExtensibleString* class. This was (and still is) a problem because an extensible string only behaves like a spring when its current length is greater than its unstretched length. At other times its effect is to supply initial conditions for the motion of a *Particle* instance. This time-dependent behaviour is currently absent from AMK. Exactly the same problem occurs in 'traditional' analysis, where it is often avoided. However, it is less easy to avoid the problem in automated modelling software like AMK. Time-dependent behaviour also has a theoretical O-O implication, which I discuss in section 10.2.3.

10.2.3 Further issues

In the following list I give a brief account of problems which are closely related to the work in this thesis, but which I was unable to pursue. I also give some reasons.

- It would be desirable to achieve a closer integration of the algebra engine with dedicated software (including, possibly, a database) to hold the object model. This would have the advantage of making optimal use of each package involved. Mathematica with MathLink (or the Derive DLL) could be used for symbolic computation, with C++ or Object Pascal to maintain the class hierarchy and visual environment. Not all of these tools existed when I first developed AMK. MathLink and the Derive DLL were not available, and the only full visual development environment (Visual Basic) could not maintain an O-O class hierarchy. The technical difficulties of developing a visual environment in Object Pascal or C++ would have meant an extensive and unnecessary programming overhead. Although this is an implementation issue, this issue is important because it drives the design of the class hierarchy, and the precise role of the CAS. Both the class hierarchy and the CAS are vitally important in this research.
- I would have liked to develop the icon-driven interface, IF2, further. This interface is intimately linked to a diagram, unlike IF1. IF2 is important for the modelling process because creating the diagram on screen generates the model. Mathematical modelling thereby reduces to a purely visual process involving icon manipulation. Particular aspects of IF2 which need further development are to extend its functionality to 2-D motion and other classes, to provide a reshaping facility for the icons, and to use MathLink as the communications protocol with Mathematica. If a simpler alternative exists it should be consistent with the following principle: *Diagram = set of entities with properties and actions.* Furthermore, an interface/algebra engine combination should be a maintainable system which could be handled by a domain expert with O-O expertise.

- Difficulties in implementing the *ExtensibleString* class expose two more general concepts which would be interesting to implement. The first concept is to find a way of creating a generalised aggregate class, objects of which would acquire attributes and methods from each constituent class. This would enable classes to be

grouped and used in a modular way. If such a way could be found, it would widen the scope of this research to more complex problem domains. Appendix 10M contains an outline of an O-O model which uses a modular approach. The emphasis in developing a modular approach should be to improve modelling, not to concentrate on technicalities (e.g. using aggregate classes, or multiple inheritance, or another method to combine classes) of an O-O implementation. The second concept is to create an object which can effectively belong to different classes at different times (which may solve the *ExtensibleString* problem). This would also introduce a time dependency into AMK. The *ExtensibleString* class is a small item in itself, but a time-dependent class is an unusual concept in O-O theory, and developing such a concept is outside the scope of this thesis.

I discuss alternative strategies later in this chapter, but they still use O-O ideas.

The attempt to solve one problem has created others. There is a need to learn a new methodology, new techniques, and to construct class libraries for each scenario. However, this focuses on pertinent issues and paves the way for automation by computer. The analysis is limited to given scenarios, and it can be hard to translate techniques to other scenarios, particularly when domain knowledge is involved. The necessary programming constructs in Mathematica are particularly difficult: they are restricted to a subset of available operations and are subject to the constraints of the object-oriented environment. A front-and for defining and managing classes would be advantageous.

The validation method described was necessary because field trials are inherently unsatisfactory. The "added value" method was a reasonable alternative. It is still subjective, needs domain knowledge and statistical analysis of the results can be difficult because of the inherent non-normality of the data. Attempts to make the scores appear more continuous (either by allocating an actual continuous score or by incorporating a more discriminatory discrete score) imply more subjectivity. Although there are attempts in Chapter 9 to reduce subjective elements, they could have gone further. One possibility is for an independent assessment of the O-O scores. Validation of this software must be seen in the light of its intended use, which is to demonstrate principles. As such, field validation is not necessary.

10.3 Opportunities for further work arising from this research

This section describes further work arising from this thesis. For each topic considered, I assess the volume and potential of the work which could be generated.

10.3.1 The Object Modelling technique

The Diagram-Axiom modelling technique is new to mathematical modelling. It is a significant departure from established techniques, and is capable of generating an equal volume of research to that contained in the recent ICTMA conference proceedings. The total work involved could amount to a 2 year project for a Research Fellow. The impact of the ideas behind the O-O analysis on finding features and relating them is a starting point for papers. There is no real need for an elaborate discussion of classes: the aim is to use classes as a vehicle to investigate how system components behave. Further detailed discussion of hierarchies and a full object model can follow.

10.3.2 Other aspects of the Modelling Cycle

The work in this thesis does not cover many aspects of a 'generic' modelling cycle. More research needs to be done on the value of these aspects as part of a modelling cycle, rather then merely describing what is done in practice. Some problems arising from modelling cycle development have been mentioned in this thesis, and most could amount to a follow-on paper. They include how to:

- differentiate between marginal and material features in a model;
- define and evaluate the effectiveness of an 'improved' model in terms of its complexity and validation outcome;
- improve the treatment of assumptions so that they act as essential ingredients in a model rather than peripheral elements which have no explicit impact on model formulation;
- develop iterative methods to use solutions as pointers to potential improvements in model formulation;
- incorporate heuristics about the problem domain and modelling practice within the problem domain.

Making assumptions explicit and using them, and evaluating marginal and material features, form the basis of a simple extension to the ideas in this thesis, and can probably be covered in several short papers. The other tasks in the above list are more substantial tasks, each amounting to a Ph.D. project.

10.3.3 The effect of computers on Teaching and Learning

The problem identified in Chapter 3 was that additional processes connected with teaching and learning mathematics using computer algebra software affect the teaching and learning environment to such an extent that the result of any 'before and after' comparison is rendered invalid. These elements include different ways of teaching, ways of presenting course material, training of teachers, course content, assessment, study time taken, attitudes towards computers and definition of learning outcomes. It would be highly desirable, but extremely difficult, to eliminate these error sources in any comparative trial. This might be tackled at two levels. The first is a Ph.D. study in which an attempt is made to reduce error factors as much as possible. The second is a much more extensive study in which sufficient trials are done in order for the results to be statistically significant. In principle, this could cover all Universities in the UK, and could attract a significant research grant. Trialling the AMK software of this thesis presents additional difficulties because it combines modelling with computer algebra and a new O-O modelling technique. The study could be of comparable length to the Kassel Project (Burghes 98): in the order of 3 to 5 years, and could be suitable for a Research Fellow.

More generally, the same type of analysis can be extended to trials of mathematical software in other disciplines and for other age groups. A suitable target is Mathwise, which is becoming more widespread, despite the lack of a rationale for its introduction. This could also be the subject of an extensive Ph.D. study.

10.3.4 The Impact of Computer Algebra Systems on the Mathematics Curriculum

In Chapter 2 I made the point that many CASs were designed with particular objectives in mind, and that a common objective is to present a general-purpose tool. The work in this thesis involved configuring such a general-purpose CAS to solve a well-defined

problem. The inherent capabilities of the CAS have therefore had a major impact on software development for this thesis. Chapter 2 discussed the problems associated with seeking ways of simplifying and manipulating expressions which are easy to use, and are as automated as possible. These problems are not new, but will gain increasing significance as computer algebra engines are incorporated into applications. This type of development may not be considered fundamental in research terms, but papers on the mechanics of expression manipulation and task feasibility pave the way for applications and more widespread use.

A more fundamental change is to redesign mathematics curricula around programming. This is, in effect, a solution to the problems in the last paragraph. In some countries (e.g. Austria, as cited in Chapter 3), this type of research could attract significant funding (in the order of £50000).

A third strand for computer algebra software development is to incorporate algebra engines into other software in a much more seamless way than was done in this thesis. Current examples include the Maple engine in MathCad and some Mathwise modules. The advantages and limitations of this strategy are largely unevaluated and are worthy of several research papers.

10.3.5 Modelling cycle methodology development

The menu-driven interface of Chapter 8 (IF1) allowed the user to use the *Construct→Link→Invoke_Methods* modelling sequence, but did not require it. As a result it was possible to deviate from the sequence, make errors and misunderstand the methodology. The icon-driven interface (IF2) of Chapter 8 attempted to address this problem, but was not a complete solution. It was still possible to find more than one route through the modelling cycle, and this could lead to confusion for the user as to what the next action should be. Further investigation and development of a *Construct→Link→Invoke_Methods* modelling cycle is therefore needed to provide sufficient guidance for the user. This is likely to be a complex piece of software that can track progress and suggest useful (or optimal) next steps. It could incorporate aspects of artificial intelligence. The amount of software development involved merits a Ph.D. thesis or a 3-year project for a Research Fellow. Such a project is comparable to Furse's *Maths Understander* (Furse 91).

10.3.6 *Diagram-Axiom* methodology development

The *Diagram-Axiom* methodology of Chapter 6 was not presented in algorithmic form, mainly because the contributory factors for it can be partially concurrent. More work needs to be done to order these factors, and to find a quick, convenient and efficient algorithm for deriving class hierarchies for new modelling contexts. If this project is backed up by field trials, it could amount to a Ph.D. thesis.

Closely connected with this task is the problem of developing a CASE tool to find classes, with their attributes and methods. Such a software project is likely to be complex in structure and should ideally capture domain heuristics. It is suited more to a commercial development rather than to a research project because applications in commercial contexts are feasible.

A persistent problem of the Diagram-Axiom methodology is that of what to do if there is no diagram, or if it would be unusual to draw a diagram, or if no diagram is possible. The comments at the end of Chapter 8 attempt to provide a way forward, but much more could be done to provide a generic solution. The work involved fits a short M.Sc. thesis.

10.3.7 Icon-driven modelling interface

A further study involving software development is to advance the icon-driven interface (IF2) of Chapter 8, to the extent that it constitutes workable software. This, allied with the theory behind it (the *Principle of Adjacency* and the *Diagram-Axiom* methodology) is a significant research project in its own right. This project has a clear link with the problem of what to do if drawing a diagram is not thought possible, or if it would be unusual to do so. This problem is not only theoretical: it is not clear that any theoretical solution can be implemented in practice so that it works as required. I have not considered applying such an interface to larger modelling contexts (particularly commercial ones). Research is required to establish the feasibility of large-scale development, and this could be a 2-year project for a Research Fellow. If directed towards industry, it could also attract funding commensurate with such an appointment.

Any commercial software project that arises could constitute a significant commercial venture, and would take the equivalent of at least 5 man-years to perfect.

10.3.8 Other contexts

In Appendix 7H I demonstrated that the O-O principles which underpin AMK can be transferred to another context - heat transfer. This one example does not show that the ideas and software are fully transferable to other contexts, and did not raise any particular problems of principle. It therefore remains to carry out a more extensive study by applying O-O principles to more contexts, preferably with larger class hierarchies. This task is likely to occupy a long dissertation, but may extend to a Ph.D. thesis if significant problems can be identified and solved.

10.3.9 Solving the equation of motion

In Chapter 7 I gave an example of how my O-O environment can be applied to a different context. This example included a *Solve* method for the *EquationOfMotion* class. Although, in principle, all that is required is a *Solve* method, in practice the situation is much more complex. Any automated system must decide whether or not an analytical solution exists. If it does, several equation-solving techniques may be available. For example, solving a system of linear equations may require partial pivoting as part of its Gaussian elimination process. Many numerical techniques are also available if there is no analytical solution . For example a numerical solution of a system of differential equations may require a simple Mathematica primitive such as *NDSolve*, or a more controlled technique such as Runge-Kutta, Quadratures or Splines. The problem of automating a 'most appropriate' technique and supplying heuristics in order to determine what can be done in particular cases could be the subject of an M.Sc. thesis.

A shorter project (a follow-up paper) is to amend the AMK software so that it always provides the initial conditions which are required to solve equations of motion. The minimum requirement would be to provide a default for each particle in the system if the user does not specify precise details.

10.4 Alternatives to O-O techniques

Throughout this thesis I have stressed an object-oriented solution to the problem of finding relationships between features in a model. This raises the question of whether or not there are alternative approaches to solving the same problem. I suggest two here. However, both contain elements which originate from this thesis.

The first builds on elements of the O-O analysis but does not specifically require any formal ideas of classes or a class hierarchy. It attempts to solve the problem of how elements in the problem domain interact, and how to keep track of the state of a model as it develops. It is therefore a less sophisticated analysis to that presented in this thesis, and could be covered in a follow-up paper. The AMK software in Chapter 7 and the theoretical object models in Chapter 6 reveal that many classes are essentially 'stand-alone': they are primitives that are not descended from other classes. They can be built into a class hierarchy but need not be. Some abstract classes exist for elegance rather than necessity. The importance of the class hierarchy is therefore diminished in favour of class attributes and methods. I therefore propose a modelling environment in which elements in the problem domain are considered as classes by identifying their characteristics (attributes) and the way in which they interact with other elements in the problem domain (their methods). As instances of these classes are created, they can be added to a list of objects currently in the problem domain. This list is an overall view of the state of the model. List manipulation in a computer algebra system is particularly easy, so this list can be very simple to maintain. In principle, modelling can proceed in the same way as in the AMK environment of Chapter 7. The programming overhead of a full object-oriented environment will then be minimal (i.e. simple list operations). A list structure for objects in the problem domain can be backed up by a graphical environment in which objects are displayed with links that are already established. Indicating further possible links would provide guidance to the user on modelling strategy. I had no trouble in implementing a brief study based on the principle of maintaining a list structure for objects using Mathematica.

As a second project, I suggest integrating a computer algebra component into a spreadsheet. The spreadsheet has been shown to be a valuable tool in modelling, and this thesis demonstrates the value of computer algebra in modelling. In addition, the spreadsheet has advantages in reinforcing routine algebraic concepts, spatial concepts,

graphing and in implementing numerical techniques. Some of the technical problems with this approach are partially solved using Mathematica's MathLink for Excel, which defines a communications protocol between these applications. There is therefore already a basis for algebraic manipulation in a spreadsheet.

One way in which the spreadsheet could be used is to support a generic modelling cycle by building a 3-D structure. Support for a generic modelling cycle also addresses the problem of defining how elements in the problem domain interact. It is a direct alternative to the O-O ideas in this thesis, and could be the subject of an M.Sc. thesis. Features can be listed on a *Features* sheet, with all necessary characteristics and interactions listed. A template on the *Features* sheet can compel the user to fill in all necessary cells before proceeding. Corresponding cells on an *Assumptions* sheet can state assumptions for each feature. If each assumption modifies the feature it relates to, it ensures that all stated assumptions are used. A third *Data* sheet can supply numerical data in corresponding cells. A fourth *Relations* sheet would have to address the problem of this thesis: to determine relations between features. This is the major task of this software development. It might proceed by examining cells on the *Features* sheet which contain references to other features. This task can be automated (rather laboriously) by cycling through pairs of features and requiring the user to indicate whether or not there is a link between the features in the pairs. This is not as clear cut as the O-O methodology in this thesis, but it would serve to focus attention on how features interact and order the modelling process. The possibility of referencing a library of heuristics, held on a further sheet, also exists. Any symbolic manipulation on the *Relations* sheet would have to be done by exporting string expressions via MathLink and importing the returned strings. A fifth *Solutions* sheet can then produce solutions using MathLink.

The Principle of Adjacency can also be exploited in a spreadsheet by drawing elementary diagrams to represent features. A useful way to do this is by colouring cells and adjusting their sizes. Adjacent cells are necessarily well-defined, and the result can be a reasonable representation of a physical system. Modelling can then proceed in the way suggested in the IF2 interface of Chapter 8. The logistic and theoretical issues of this type of modelling environment are not trivial, and could be the subject of a further Ph.D. thesis.

Appendices

Appendix 1B

Single sample test of proportions: Burghes 93

H(0): Number of trials in which Germany did better ~B(27,0.5)
H(alt): Number of trials in which Germany did better ~B(27,p): p>0.5

Germany did better in 18 questions
England did better in 9 questions

p = 0.6667
s = 0.0907
z = 1.8371
Not significant at the 5% level

2-sample z-test: Burghes 93

H(0): Mean test score in Germany = Mean test score in England
H(alt): Mean test score in Germany > Mean test score in England

	England	Germany
Sample size	540	302
Mean	12.73	14.05
SD	4.96	4.82

z = 3.7
Significant at 0.5%

Variance ratio test to establish that these samples can be considered to come from the same background population
H(0): Var(England) = Var(Germany)
H(alt): Var(England) > Var(Germany)

F = 1.1
F(crit) = 1 (nu1 = nu2 = infinite), 5% significance level
Since F > F(crit), reject H(0)

Conclusion: these are not samples from the same population, so we cannot apply a z-test.

Appendix 1G

The following is a quote from my first draft (with comments) for a goodness of fit question using a contingency table for UCLES paper 4861, March 1996 presentation. My second draft is still in the post…

For real data, how about this (Guardian, 29/11/94)

	D	*not D*
Q	*150*	*50*
not Q	*300*	*430*

where D = is a Director of a FTSE-100 Company which has donated to the Conservative party and Q = Has been appointed to the Board of a QUANGO.
Are attributes D and Q independent?
Is it too controversial!?

Appendix 1GWF

Modelling topics in Giordano, Weir and Fox (Giordano 97)

Case study	Technique	Model catogory	Similar in
Savings/mortgage	Recurrence relation	Discrete dynamical system	S
Biological growth	Recurrence relation	Discrete dynamical system	S, B
Cooling	Recurrence relation	Discrete dynamical system	S
Predator=prey	Recurrence relation system	Discrete dynamical system	S
Car rentals	Recurrence relation system	Discrete dynamical system	S
Voting	Recurrence relation system	Discrete dynamical system	S
Stopping distances	Fitting linear models	Empirical Linear laws	B
Car Mileage	Fitting linear models	Empirical Linear laws	
Stopping distances	Fitting non-linear models	Empirical non-linear laws	OU, EH
Harvesting	Fitting non-linear models	Empirical non-linear laws	OU, EH
Stopping distances	Polynomial splines	Empirical non-linear laws	
Area under curve	Random number sampling	Monte Carlo Simulation	EH
Inventory/Delivery	Random number sampling	Monte Carlo Simulation	EH
Queue	Random number sampling	Birth/death simulation	EH
Biological growth	Calculus/Polynomial fitting	Continuous optimisation	
Inventory/stock control	Calculus	Continuous optimisation	
Storage/holding cost	Lagrange multiplier	Continuous optimisation	
Allocation	LP/Diagram	Discrete optimisation	OU, HJ(*1)
Allocation	LP/Simplex	Discrete optimisation	HJ(*2)
Population growth	ODE	Numerical approximation	OU, B, EH, HJ(*3)
Drug dosage	Piecewise continuous/ODE	Numerical approximation	
Stopping distance	ODE	Numerical approximation	OU, EH
Compound interest	ODE	Numerical approximation	
Predator/Prey	ODE system	Numerical approximation	EH
Arms race	ODE system	Numerical approximation	EH
Voting behaviour	Stochastic/Markov chain	Probabalistic models	HJ
Tree harvest	Linear regression	Sum of squares optimisaton	OU, HJ

Not in GWF

Bulk Rainfall	Volumetric	Geometric	EH, HJ(*4)
Total rainfall	Volumetric	Calculus	EH
Tape/reel	Rate of change	Circular motion	EH, B, OU
Resisted motion	ODE/Taylor approximation	Newtonian mechanics	EH, B, OU
Spring/dashpot	2nd order ODE	Newtonian mechanics	B, OU, HJ
Washing up	Cooling	Heat tranfer	OU, B
Cooling: house/liquid	Cooling	Heat tranfer	OU, B, HJ
Disk pressing	Mensuration	Geometric	EH, HJ
Fluid accumulation	ODE	Input/Output	EH, OU, HJ
Snooker	Kinematic	Impulsive motion	EH
Kinematics	Uniformly accelerated motion	Linear/circular motion	OU, HJ, B
Harvest	Running/installation cost	Continuous optimisation	HJ, B, OU

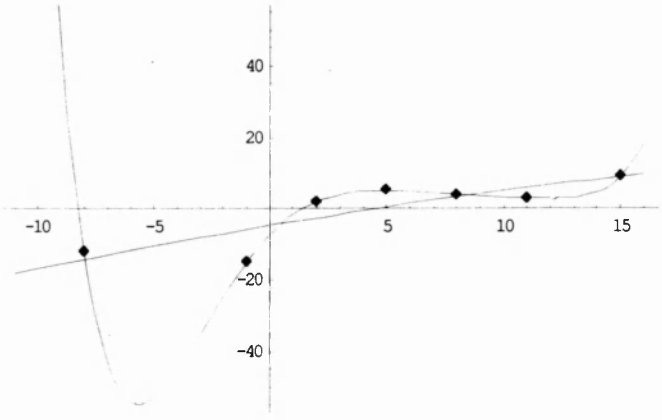
EH	(Edwards 89)
GWF	(Giordano 97)
B	(Berry 95)
S	(Sandefur 90)

Unusual or notable treatments

HJ(*1)	Route planning	Average expressed as integral to find mean distance
HJ(*2)	Apportionment	Huntington method
HJ(*3)	Population growth	BASIC program required to obtain solution
HJ(*4)	Windmill mechanics	Bulk pressure + circular motion + numerical solution

Appendix 2B

```
In[1]:=
data = {{-8,-12},{-1,-15},{2,2},{5,5},{8,4},{11,3},{15,9}};
(* Calculate and plot the interplolating polynomial,
   and least squares quadratic and least squares
   linear fit lines. *)

In[2]:=
ip = InterpolatingPolynomial[data,x]//Expand//N;
f1 = Fit[data, {1,x},x];
f4 = Fit[data, {1,x,x^2,x^3,x^4},x];
points = ListPlot[data, PlotStyle->PointSize[.02],
                        DisplayFunction->Identity]
ipplot = Plot[ip, {x,-11, 16},PlotStyle->RGBColor[0,0,1],
    DisplayFunction->Identity]
f1plot = Plot[f2, {x,-11, 16},PlotStyle->RGBColor[1,0,0],
    DisplayFunction->Identity]
f4plot = Plot[f4, {x,-11, 16},PlotStyle->RGBColor[0,1,0],
    DisplayFunction->Identity]
Show[{points,f1plot,f4plot,ipplot},
    DisplayFunction->$DisplayFunction]
```

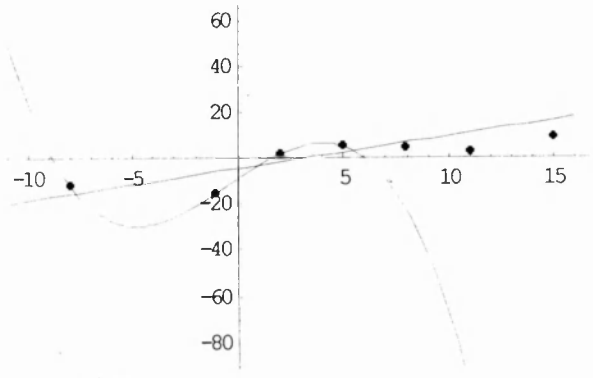



```
(* The interpolating polynomial shows a spike near x = -6,
   whereas the quadratic fit 'looks' good.  To quantify this,
   let f(x) be a piecewise continuous linear function that
   joins the points in sequence.  Calculate the sum of the
   deviations of each curve from f(x) at the mid-point of
   each adjacent data pair.  *)

In[3]:=
MidLin = Map[{(#[[1,1]]+#[[2,1]])/2,(#[[1,2]]+#[[2,2]])/2}&,
    Partition[data,2,1]];
errorIPmid = Plus @@ Map[((ip/.x->#[[1]]) - #[[2]])^2&,MidLin]
error4mid = Plus @@ Map[((f4/.x->#[[1]]) - #[[2]])^2&,MidLin]
error1mid = Plus @@ Map[((f1/.x->#[[1]]) - #[[2]])^2&,MidLin]

Out[3]=
1350.47
162.477
58.763
```

```
(* A more common measure of goodness of fit of an approximation
   to a discrete data set is the sum of squares of deviations
   at the data points.  *)

In[4]:=
errorIPords = Plus @@ Map[((ip/.x->#[[1]]) - #[[2]])^2&,data]
error4ords = Plus @@ Map[((f4/.x->#[[1]]) - #[[2]])^2&,data]
error1ords = Plus @@ Map[((f1/.x->#[[1]]) - #[[2]])^2&,data]

Out[4]=
           -27
1.23107 10
15.7285
142.856

(* Combining these two measures for each approximation by adding
   the error measures, we see that the quadratic approximation
   gives the lowest combined error measure, which is consistent
   with an intuitive view of the graphs.  The linear approximation
   is better than the interpolating polynomial because of the
   large deviation of the latter at the mid-points of the data.  *)
```

Robustness Test: fitting some of the data

```
(*  Fit the first 4 points only, and see, intuitively,
    whether or not the fits obtained project successfully
    to the other data points.  *)

In[5]:=
ipSome = InterpolatingPolynomial[Take[data,4],x]//Expand//N;
f1Some = Fit[Take[data,4], {1,x},x]
f4Some = Fit[Take[data,4], {1,x,x^2,x^3,x^4},x]
ipplotSome =
  Plot[ipSome, {x,-11, 16},PlotStyle->RGBColor[0,0,1],
    DisplayFunction->Identity]
f1plotSome =
  Plot[f1Some, {x,-11, 16},PlotStyle->RGBColor[1,0,0],
    DisplayFunction->Identity]
f4plotSome =
  Plot[f4Some, {x,-11, 16},PlotStyle->RGBColor[0,1,0],
    DisplayFunction->Identity]
Show[{points,f1plotSome,f4plotSome,ipplotSome},
  DisplayFunction->$DisplayFunction]
```




```
-Graphics-

(* Clearly, the linear fit is more robust in an intuitive sense.
   The other projections diverge markedly. *)
```

Appendix 2D

Proof of the First MVT, (Devitt 89).

```
A:= [a, f(a)]; B:= [b, f(b)];
m:= slope(A,B);
f1:= makeproc(f(a) + m*(x-a), x);


# h defines the vertical distance from the line segment AB
# to (x,f(x)). Apply Rolle's theorem to h to show  h(a)=h(b).
h:= f - f1;
[h(a), h(b)];


# Gives result [0, f(b)-f(a)-(f(a)-f(b))/(a-b)*(b-a)]
map(normal,");
# Gives result [0,0]
# but h must be continuous on (a,b)


# Now differentiate h with respect to x.
D(h)(x);
# By Rolle's theorem there exists c in (a,b) such that:
D(h)(c) = 0;
isolate(",D(f)(c));
# Gives result   D(h)(c) = (f(a)-f(b))/(a-b)
# which is the same as the slope m of f1
```

Appendix 3W

The following analyses relate to tables in (Wenger 88). Ability groups A-E are classified according to their initial algebra placement scores A,B and C. Groups 1 are heavy users of computers and groups 2 are light users of computers.

Analysis of Table 5: post-gradings by ability group A,B,C, group 1=heavy users of computers

Observed

Group	A	B	C	D	E	
A1	9	13	3	0	0	25
A2	25	16	18	5	5	69
	34	29	21	5	5	94

Expected

	A	B	C	D	E	
	9.0426	7.713	5.585	1.33	1.33	25
	24.957	21.29	15.41	3.67	3.67	69
	34	29	21	5	5	94

Grouped cells	A	B	CDE	
	9	13	3	25
	25	16	28	69
	34	29	31	94

	A	B	CDE	
	9.0426	7.713	8.245	25
	24.957	21.29	22.76	69
	34	29	31	94

Chi-Square contributions

0	3.6	3.3
0	1.3	1.2

Chi2= 9.483

On 2 DF ,critical 1% Chi2 = 9.21 SIG

Observed

Group	A	B	C	D	E	
B1	41	30	27	6	2	106
B2	31	37	27	27	19	141
	72	67	54	33	21	247

Expected

	A	B	C	D	E	
	30.899	28.75	23.17	14.16	9.012	106
	41.101	38.25	30.83	18.84	11.99	141
	72	67	54	33	21	247

Chi-Square contributions

3.3	0.1	0.6	4.7	5.5
2.48	0	0.5	3.5	4.1

Chi2= 24.78

On 4 DF ,critical 1% Chi2 = 13.28 SIG

Observed

Group	A	B	C	D	E	
C1	21	37	42	30	25	155
C2	19	48	75	56	83	281
	40	85	117	86	108	436

Expected

	A	B	C	D	E	
	14.22	30.22	41.59	30.57	38.39	155
	25.78	54.78	75.41	55.43	69.61	281
	40	85	117	86	108	436

Chi-Square contributions

3.23	1.5	0	0	4.7
1.78	0.8	0	0	2.6

Chi2= 14.65

On 4 DF ,critical 1% Chi2 = 13.28 SIG

Analysis of Table 6: possible evidence that number of diagnostic tests attempted affects subsequent gradings

Tests attempted	N	Passes obs	exp	Chi-Square	
>6	24	19	14	1.9459	
5,6	34	28	20	3.6298	
3,4	48	27	28	0.0144	
1,2	48	27	28	0.0144	
0	84	36	48	3.1559	
Totals	238	137	137	8.7603	Critical 5% (4DF) Chi-Square = 9.49 NOT SIG

Appendix 3H

Sample tests for differences in proportions:
Table 3: **Quiz** data and Table 5: **Final Exam**

p(e1) = the proportion of correct responses in Experimental group 1
p(e2) = the proportion of correct responses in Experimental group 2
p(ct) = the proportion of correct responses in the Control group
H(null): p(e1) = p(ct) and p(e2) = p(ct) H(alt): p(e1) = p(ct)

Note that experimental sample sizes are really too small to apply this test, but there is no suitable alternative.

	Expt 1	Expt 2	Control	
Question 7B (quiz)				
Sample sizes	18	17	100	
Quoted Proportions	0.84	1	0.79	
Correct responses	15.12	17	79	Accept null hypothesis at 5% significance for Expt 1
z	0.486059	2.08589		Reject null hypothesis at 5% significance for Expt 2

	Expt 1	Expt 2	Control	
Question 7C (quiz)				
Sample sizes	18	17	100	
Quoted Proportions	0.53	0.19	0.68	
Correct responses	9.54	3.23	68	
z	-1.23422	-3.82729		Reject NH against alternative p(e2)<p(ct) at 5% significance

Summary: Table 3 (Quiz data)

32 trials
Null hypothesis rejected in 10 trials, and accepted in 22 trials
Expt performed worse in 2 out of 32 cases

Summary Table 5: (Final exam data)

18 trials
Null hypothesis rejected in 0 trials, and accepted in 18 trials
Expt performed worse in 9 out of 18 cases

Tests for differences in means: Table 4

	Expt 1	Expt 2	Control	
Sample sizes	18	17	100	
Mean	105	115	117	
SD	43.6	40.9	36.7	Experimental means have clearly not improved

Single sample t-tests for individual Final Exam items: Table 5

NH: m(E1-C) = 0 and m(E2-C) = 0 AH: m(E1-C) < 0 and m(E2-C) < 0

(Note that the alternative hypotheses indicate a worse result from the experimental groups)

Item	Expt 1 %	Expt 2 %	Control %	C-E1	C-E2
1	49	53	53	4	0
2	44	59	44	0	-15
3	39	35	37	-2	2
4	17	18	16	-1	-2
5	50	71	70	20	-1
6	53	54	64	11	10
7	22	21	28	6	7
8	11	12	26	15	14
9	33	24	35	2	11
n				9	9
m				6.1111 11	2.888889
sd				7.6721	8.838049
t				2.2529 47	0.924527

For E1, reject NH at 5% significance (t-critical = 1.86, 8 DF)

For E2, Accept NH at 5% significance (t-critical = 1.86, 8 DF)

Appendix 3M

Data from (Mayes 97)

"1": computer group out-performed the control group
"-1": computer group did not out-perform the control group
"0": computer group control group were equivalent

Author	Year	Attitude to maths	Manipulation skills	Conceptual understanding	Modelling
Galinda-Morales	1995		0	0	
Padgett	1995		-1	-1	
Porzio	1995			1	
Keller	1994		-1	1	
Klein	1994	0	0		
Alexander	1993	0		1	1
Coons	1993	1			
Melin-Conejeros	1993	0	0	0	1
Park	1003	1	0	1	
Trout	1993	1			1
Crocker	1992			1	-1
Cunningham	1992		0	1	
Smith	1992	0			0
Schrock	1990	1	0	1	
Judson	1988		0	0	0

Sign tests, using:

H(null): Median number of studies with outcome 1 for experimental group = Median number of studies with outcome 1 for control group
H(alt): Median number of studies with outcome 1 for experimental group > Median number of studies with outcome 1 for control group
Let the random variable X be the number of studies in which the outcome is 1
Under H(null), X ~ B(n, 0.5) where n is the number of trials with non-zero outcomes

Attitude				
X ~ B(4, 0.5)	P(X>=4) =	0.0625	Not significant at 5%	Accept H(null)
Manipulation				
Clearly no overall improvement				
Conceptual understanding				
X ~ B(8, 0.5)	P(X>=7) =	0.035156	Significant at 5%	Reject H(null)
Modelling				
X ~ B(4, 0.5)	P(X>=3) =	0.3125	Not significant at 5%	Accept H(null)

Appendix 3H1

Hillel, Lee, Laborde and Linchevski 1992

Raw data:	Percentage pass rate, experimental group:	65%	Sample size 18
	Percentage pass rate, control group:	53%	

H(null) Mean proportionate pass rate, p = 0.53
H(alt) Mean proportionate pass rate, p > 0.53
z = (.65-p)/Sqrt(p(1-p)/n) 1.020072
Not significant at 5%, accept null hypothesis

Appendix 3K

Klinger 1994

	n	Proportion with correct results	
Expt group 1	26	0.64	DERIVE used for structural interpretation of algebraic terms
Expt group 2	31	0.82	DERIVE used for learning technique
Control	90	0.47	

For expt group 1
H(null) Mean proportionate correct result rate, p = 0.64
H(alt) Mean proportionate correct result rate, p > 0.64

z = 1.736796 Significant at 5%

For expt group 2
H(null) Mean proportionate correct result rate, p = 0.82
H(alt) Mean proportionate correct result rate, p > 0.82

z = 3.904469 Significant at 0.5%

Appendix 5A: Object-Oriented Analysis

Practical explanations of objects and classes tend to come in texts on programming, and C++ texts are the most numerous. Parsons (Parsons 94) and Pardoe (Pardoe 97) both give clear accounts of the basic concepts and C++ programming in standard texts. Lafore's account of more advanced concepts in (Lafore 95) is backed up by good programming examples. Tan (Tan 98) provides a short account of O-O concepts in the middle of an advanced text on how to program a computer algebra system in C++. It also has a clear explanation of more advanced ideas. Horstmann (Horstmann 97) links a discussion of object models with simple object-oriented design.

Basic O-O concepts

The purpose of object-oriented programming is to mimic a world of identifiable, interacting and classifiable objects. An *object* can be loosely defined as *something concrete or abstract which we can perceive in the world.* Collections of like objects constituted a *class*. A useful way to think about classes is to consider a blank form with headings. The headings characterise the class. An individual object (an *instance* of the class) is defined when a copy of the form is filled in.

Two factors must be considered when describing a class and the objects in it. The first is what the characteristics (data) of the class are. These are generally noun phrases and are called *attributes* of the class. The second is how objects in the class behave (their processes). These processes can usually be expressed in terms of verb phrases and are called *methods* of the class. Thus, a class has a name, attributes and methods. An object has identity, state (the values of the attributes) and actions (what actually happened). This unification of data and processes is known as *encapsulation*. It is usual in object models not to interact directly with the attributes of the class. The attributes are only visible to objects within the class . They are accessed by defining methods which use them. This principle is known as *information hiding*, and it partitions the attributes and methods of a class into two distinct subsets. The first is the *private* section, which hides attributes and methods so that programmers cannot access them directly. This is a protection against making fundamental changes to the basis of a system which will have significant consequences elsewhere in the system. The second is the *public* section

which contains methods, attributes and constants that are accessible to the user. Objects can only access their own attributes directly. Figure 5.A1 summarises these concepts.

Class = Data + Processes; Object = Data values + Events

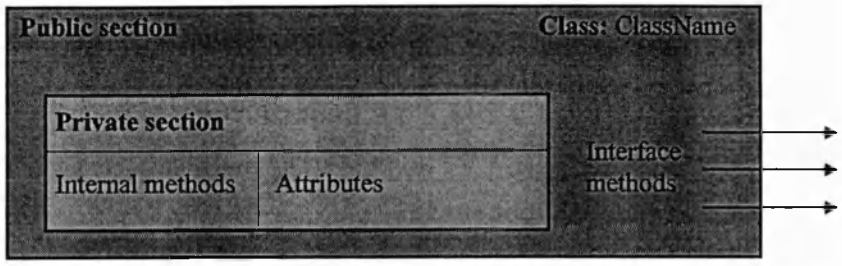


Figure 5.A1

The *public* section also contains two other important methods. The first is a *constructor*, which creates an object. The second, which is not needed in all languages, is a destructor, which destroys it. In memory management terms, these methods reserve memory for the object and release it when it is no longer needed respectively.

Objects communicate by *message passing*. A message has a name associated with a method of the receiver. This method is activated on receipt. A link between objects (usually of different classes) is called an *association*, and messages are passed according to associations. Strictly speaking, the `LinkObjects` procedure of this thesis (Chapter 7) is an explicit association.

In order to illustrate these concepts, I develop a brief object model which could be used for Boundary Element analysis. It is simply a description of the boundary of a physical object, and consists of a sequence of nodes. The characteristics of each node are a unique identity, x and y co-ordinates, and the values of two parameters: its potential and flux. It interacts with the outside world by displaying the values of these parameters in response to a mouse click within the circle that represents the node. Each node has a constructor and a destructor method. The boundary has a list of nodes as an attribute, and its own constructor and destructor methods. Here is a partial C++ implementation, in which the constructor sets x and y but the potential and flux are set by separate methods. This is not the only way of doing it: all or none could be set in the constructor.

A header file Shape2.h defines the interface for the Tnode and the TNodeList classes.

```
class TNode : public TShape
{
 private:
    double Flux, Potential;
    double X, Y;
    String ID;
    String GetCentre();
 public:
    TNode(int x, int y, String id);  // Constructor
    ~Tnode();                        // Destructor
    void MouseDown(int Xm, int Ym);
    void SetFlux(double flux);
    void SetPotential(double potential);
    double GetFlux();
    double GetPotential();
    String GetID();
};

class TNodeList: public TList
  {
   public:
     TNodeList();                    // Constructor
  };
```

The source file Shape2.cpp defines the implementation for the TNode class and the TNodeList class. Tnode objects react to mouse clicks by displaying information held in their attributes.

```
#include "Shape2.h"

TNodeList::TNodeList()
{
 TList *NodeList = new TList;  // Constructor
}

TNode::TNode(int x, int y, String id) // Constructor
{
 X = y;
 Y = x;
 ID = id;
 Flux = 0.0;
 Potential = 0.0;
 Circle(x,y,2);   // Draw a circle, centre (x,y), radius 2.
}

void TNode::SetFlux(double flux)
{
 Flux = flux;
}

void TNode::SetPotential(double potential)
{
 Potential = potential;
}
```

```
double TNode::GetPotential()
{
 return Potential;
}

double TNode::GetFlux()
{
 return Flux;
}

String TNode::GetID()
{
 return ID;
}

String TNode::GetCentre()
{
 char yStr[25];
 char xStr[25];
 int sig = 5;      // significant digits
 String s;
 gcvt(X, sig, xStr);
 gcvt(Y, sig, yStr);
 s = "(" + String(xStr) + ", " + String(yStr) + ")";
 return s;         // as a coordinate pair
}

void TNode::MouseDown(int Xm, int Ym)
{
 char yStr[25];
 char xStr[25];
 char FluxStr[25];
 char PotentialStr[25];
 int sig = 10;    // significant digits
 String s;
 if ( (Xm-X)**2 + (Ym-Y)**2 <= 4 )  // clicked in the node (radius 2)
   {
    gcvt(X, sig, xStr);
    gcvt(Y, sig, yStr);
    gcvt(GetFlux(), sig, FluxStr);
    gcvt(GetPotential(), sig, PotentialStr);
    s = "Identity = " + ID;
    s = s + "\nPotential = " + PotentialStr;
    s = s + "\nFlux = " + FluxStr;
    s = s + "\nCentre at " + GetCentre();
    ShowMessage(s);
   }
}
```

The source file ShapeMain.cpp creates some TNode instances (CreateSomeNodes()) and puts them in an instance of the TNodeList class. Procedure DeleteTheNodes() destroys them.

```
void CreateSomeNodes()
{
 Boundary = new TNodeList();
 int i;
 TNode* nn;
 for (i = 0; i<5; i++)
    {
      nn = new TNode(40+10*i, 40+30*i, "Node" + itoa(i));
      nn->SetFlux(i-42.1);
      nn->SetPotential(i+2.1);
      Boundary->Add(nn);
    }
}

void DeleteTheNodes()
{
 if (Boundary != NULL)
    {
     TNode* pNode;
     for (i = 0; i < Boundary.Count; i++)
      {
       pNode = (TNode*)( Boundary.Items[i]);
       if (!pNode) delete pNode;
      };
     delete Boundary;
    };
}
```

Further O-O concepts

One advantage of using O-O software is (said to be) reuse of existing software through *inheritance*. A new class can be derived from an existing class by adding new attributes and methods or by writing *override* methods, which supersede methods of the same name in the parent class. The compiler provides a means to decide which version of the method is to be processed at run-time. As an example, suppose that the Boundary Element object model is to contain information about the type of Boundary Element analysis which will be used: constant, linear or quadratic. Instead of redefining the classes Tnode and TNodeList, a new class, TBEMNode , descended from TNode, can be defined. It possesses all the attributes and methods of TNodeList, in addition to new ones.

```
enum MethodType {constant, linear, quadratic};

TBEMNode: TNode
{
 private:
   MethodType BEmethod;                    // new attribute
 public:
   MethodType GetBEmethod();               // new method
   void MouseDown(int Xm, int Ym);         // override method
}
```

A version of the method `MouseDown(int Xm, int Ym)` which is appropriate for this descendent class has to be coded in the implementation source file. The idea of the same method appearing in multiple guises is known as *polymorphism*. Operators can also be redefined to perform an equivalent operation on a derived class. This kind of polymorphism is known as *operator overloading*. Basing new classes on existing classes gives rise to a *class hierarchy*, which can become very complex in a large system. It is sometimes convenient to define classes that have no instances. These are called *abstract classes*, and are there because it is convenient to base *concrete classes*, which do have instances, on them. The class `TNode` of this example is really an abstract class because a node in this context must have a `MethodType` associated with it to be realistic. Polymorphism has implications for inter-object communication by *message passing*. What happens when the message is received depends on the class of the receiver: there must be a suitable method to react to the message.

Relationship with the Procedural programming paradigm

The procedural programming paradigm, as used in C, Pascal, Fortran etc., concentrates on actions and events in the problem domain. Procedures refer to elements in the problem domain through variables and data types. The correspondences with the O-O paradigm are:

Procedural	*O-O*
Variables	objects
Data types	classes
Functions/procedures	methods
Function calls	message passing

Other concepts

The concepts outlined above were all used in programming the software for this thesis. Other, more advanced concepts merit a mention. Of these, only multiple inheritance has a bearing on this thesis.

Container classes provide storage for data items or objects. They are often used to implement generic data structures such as linked lists or vectors. The aim is to make

them applicable for any elementary data type (e.g. a linked list can be used for integers, strings, Tnodes etc.)

Virtual functions are intended to provide basic services for an operation, but not to be called directly. They are overridden in a descendent class, and the descended versions are called.

Templates functions provide different functionalities, depending on their arguments. The type of argument, and the number of arguments can vary. Classic examples are the `cout` and `cin` output and input functions of C++, which provide for output and input of all the basic data types in C++.

In *multiple inheritance*, a class can be descended from more than one base class. It can always be avoided by revising the class hierarchy, and is currently out of favour.

An *aggregation* is an inclusion relation between classes in which subclasses do not inherit from the superclass. The superclass would have little meaning in the absence of the subclasses. For example, a class `MathematicalModel` might include subclasses `Features`, `Assumptions`, `Data` and others, but none of them need be descended from `MathematicalModel`.

Finally, to get out of a sticky situation, you need a *friend*. These are functions or classes that can break the encapsulation and data-hiding rules by accessing the private elements of other classes. So the GOTO statement is alive and well...

Appendix 7FS

Functional specification for AMK Mathematica software

Auxillary Functions

Function *SupplyInfo[l_List, title_, value_]*
Appends the element *title→value* to the list l: used when constructing objects.

Rewrite rule *SRrule*
Reduces functions of $\sqrt{n^2}$ to n (assumption: $n \geq 0$).

Rewrite rule *SignRule*
Returns Sign[n] for functions of symbolic n.

Rewrite rule *TrigSquare*
Reduces $a\ cos^2(x) + a\ sin^2(x)$ to a.

Rewrite rule *Mag*
Returns the magnitude $|v|$ of a vector v.

Function *GetArgument*
Returns the argument list p of a function $f[p]$

Rewrite rule *ToC[alpha_, s_]*
Converts Polar coordinates, angle α to an initial line with sense s, to Cartesian.

Rewrite rule *ToP[alpha_, s_]*
Converts Cartesian coordinates, angle α to an initial line with sense s, to Polar.

Class Definitions

Class:
CoordinateTransformations: virtual (applicable to 2D vectors)
Parent:
Base class
Attributes:
None
Methods:

Magnitude:	Returns its own magnitude.
Direction:	Returns an angular coordinate relative to the positive x-axis.
X:	Returns its own x-coordinate.
Y:	Returns its own y-coordinate.

Class:

CoordinateSystem

Parent:

CoordinateTransformations

Attributes:

c:	Coordinate list.
info:	Parameter information store - list of rewrite rules.

Methods:

new:	Constructor.
Type:	Cartesian/Polar.
Coordinates:	{x,y} or {r, theta}
Displacement:	Distance from Origin
X:	x-displacement; Override.
Y:	y-displacement; Override.
Magnitude:	Magnitude; Override.
Direction:	angular coordinate relative to the positive *x*-axis; Override.
Info:	Value of Info attribute.
Angle:	Polar Angular coordinate.
Sense:	Rotation sense of coordinate.
ToCartesian:	Constructor - this object with Cartesian coordinates.
ToPolar:	Constructor - this object with Polar coordinates.
ExpressAsCartesian:	Rotation of Cartesian principal axes.

Class:

Particle

Parent:

CoordinateSystem

Attributes:

t:	Elapsed time.
m:	Mass.

Methods:

new:	Constructor; Override.
Mass:	Value of m attribute.
T:	Value of t attribute.
Velocity:	Velocity in Polars or Cartesians.
Acceleration:	Acceleration in Polars or Cartesians.
ToCartesian:	Constructor - this object with Cartesian coordinates; Override.
ToPolar:	Constructor - this object with Polar coordinates; Override.
InitialiseDisplacement:	Sets Info InitialDisplacement
InitialiseVelocity:	Sets Info InitialVelocity
InitialVelocity:	Boolean test: true if InitialVelocity not set.
InitialDisplacement:	Boolean test: true if InitialDisplacement not set

Class:

GravitationalField

Parent:

CoordinateSystem

Attributes:

None (the symbol *g* is a reserved and protected word)

Methods:

new:	Constructor; Override.
SupplygNumeric:	Supply a numerical value for *g*.
gNumeric:	Retrieve the numerical value for *g*.
gToNumeric:	Change the numerical value for *g*.

Class:

Force

Parent:

CoordinateSystem

Attributes:

None

Methods:

new:	Constructor; Override.

Class:

EquationOfMotion

Parent:

Base class

Attributes:

p$:	Mass × Acceleration.
f$:	Sum of Forces.

Methods:

new:	Constructor.
MassAcceleration:	Returns Mass × Acceleration.
SumOfForces:	Returns Displacement of Sum of Forces.
Equation:	Returns the Equation of Motion: p$ == f$

Class:
HorizontalPlane
Parent:
CoordinateSystem
Attributes:

mu:	Coefficient of Friction.

Methods:

new:	Constructor; Override;
CoefficientOfFriction:	Returns the Coefficient of Friction.
SupplyFriction:	Sets Info parameter: symbol for Friction.
Friction:	Returns the symbol for Friction.
SupplyNormalReaction:	Sets Info parameter: symbol for Normal Reaction.
NormalReaction:	Returns the symbol for Normal Reaction.

Class:
InclinedPlane
Parent:
HorizontalPlane
Attributes:

slope:	Angle of inclination to the horizontal.

Methods:

new:	Constructor; Override.
Slope:	Returns slope.

Class:
InextensibleString
Parent:
CoordinateSystem
Attributes:

hiEnd:	Coordinates of non-inherited end.

Methods:

new:	Constructor; Override.
Length:	Returns length of string.
LoEnd:	Returns inherited coordinates
HiEnd:	Returns hiEnd coordinates.
SupplyTension:	Sets symbol for Tension.
Tension:	Returns symbol for Tension.

Class:
Spring
Parent:
CoordinateSystem
Attributes:

hiEnd:	Coordinates of non-inherited end.
Long:	Natural length.
Stiff:	Stiffness.

Methods:

new:	Constructor; Override.
NaturalLength:	Returns Natural length.
Stiffness:	Returns Stiffness.
LoEnd:	Returns Coordinates of inherited end.
HiEnd:	Returns Coordinates of non-inherited end.

Class:
CircularSurface
Parent:
HorizontalPlane
Atributes:

r$:	Radius.

Methods:

new:	Constructor; Override.
Radius:	Returns radius.

Class:
NonLinearSpring
Parent:
Spring
Attributes:

nlp$:	NonLinearParameter p in *Tension* = $(extn)^p$

Methods:

new:	Constructor; Override.
NonLinearParameter:	Returns nlp$.
ComputeTension:	*Not implemented* - defined in a LinkObjects procedure.

Class:

Dashpot1 (first definition)

Parent:

Spring

Attributes:

tm$:	time - differentiation parameter.
r:	*not implemented* - dashpot parameter r in $F = r\|\dot{x}\|$. Uses inherited *Spring.stiff*.

Methods:

new:	Constructor; Override.
T:	Returns value of tm$.

Class:

Dashpot (second definition)

Parent:

CoordinateSystem

Attributes

hiEnd:	Coordinates of non-inherited end.
long:	End-to-end Displacement.
stiff:	dashpot parameter r in $F = r\|\dot{x}\|$.
tm:	time - differentiation parameter.

Methods:

new:	Constructor; Override.
NaturalLength:	Returns End-to-end Displacement.
Stiffness:	Returns dashpot parameter.
LoEnd:	Returns Coordinates of inherited end.
HiEnd:	Returns Coordinates of non-inherited end.
T:	Returns time parameter.

Boolean class query functions

ParticleQ[x_]	True if *x* is a Particle, False otherwise
ForceQ[x_]	True if *x* is a Force, False otherwise
SpringQ[x_] :=	True if *x* is a Spring, False otherwise

Conditions:

x is not a *Dashpot1* or *NonLinearSpring*

NonLinearSpringQ[x_]	True if *x* is a NonLinearSpring, False otherwise
DashpotQ[x_]	True if *x* is a Dashpot, False otherwise
Dashpot1Q[x_]	True if *x* is a Dashpot1, False otherwise
GravitationalFieldQ[x_]	True if *x* is a GravitationalField, False otherwise
HorizontalPlaneQ[x_]	True if *x* is a HorizontalPlane, False otherwise

Conditions:

x is not a *InclinedPlane* or *CircularSurface*

InclinedPlaneQ[x_]	True if *x* is a InclinedPlane, False otherwise
EquationOfMotionQ[x_]	True if *x* is a EquationOfMotion, False otherwise
InextensibleStringQ[x_]	True if *x* is a InextensibleString, False otherwise
CircularSurfaceQ[x_]	True if *x* is a CircularSurface, False otherwise

Polymorphic Force Calculus

Overloaded binary operator + f1_?ForceQ + f2_?ForceQ

Inputs:

Force instances f1, f2

Outputs:

Constructor for new Force instance, f1 + f2 in Cartesian form

Polymorphic forms:

f1, f2 are 1D, senses aligned

f1, f2 are 1D, senses opposed

f1, f2 are 2D, senses aligned

f1, f2 are 2D, senses opposed

Overloaded unary operator – – f1_?ForceQ

Inputs:

Force instance f1

Outputs:

Constructor for new Force instance, - f1, in Cartesian form

Polymorphic forms:

f1 is 1D

f1 is 2D

Multiplier c_ f1_?ForceQ

Inputs:

Force instance *f1*

Symbolic or numerical constant *c*

Outputs:

Constructor for new Force instance, c f1, in Cartesian form

Polymorphic forms:

None

Polymorphic LinkObjects Function

Functional form: LinkObjects[x_?Class1Q, y_?Class2Q]

Inputs:

Object instances *x*, *y*

Outputs:

Constructor for new object instance, *z*, class dependent on input classes.

Polymorphic forms:

x: Particle;	y: GravitationalField;	z: Force
x: Particle;	y: Force;	z: EquationOfMotion
x: Particle;	y: HorizontalPlane;	z: Force (Friction + Normal Reaction)
x: Particle;	y: InclinedPlane;	z: Force (Friction + Normal Reaction)
x: Particle;	y: Spring;	z: Force (Tension)
x: Particle;	y: NonLinearSpring;	z: Force (Tension)
x: Particle;	y: Dashpot;	z: Force (Resistance)
x: Particle;	y: Dashpot1;	z: Force (Resistance)
x: Particle;	y: InextensibleString;	z: Force (Tension) - Cartesian or Polar
x: Particle;	y: CircularSurface;	z: Force (Friction + Normal Reaction)
x: Force;	y: Force;	z: Force *not implemented*

Appendix 7H

The following model demonstrates an application of O-O modelling techniques to the context of a simple heat transfer problem. A heat transfer problem was discussed in Chapter 6, and Chapter 9 discusses students' attempts to solve it. In the O-O implementation there are four primitive classes (Figure 7H.1)

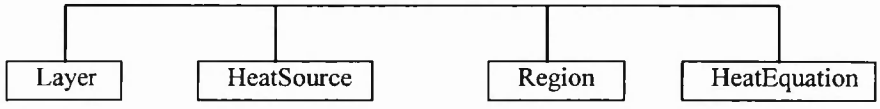


Figure 7H.1

The problem domain consists of a cuboid polystyrene box, a convective layer associated with the walls and lid, and a heat absorber (Figure 7H.2). The parameters are:

- Heat source, mass m, specific heat c
- 4 Polystyrene walls, U-value Uw, area A, thickness s
- 1 Polystyrene floor, U-value Uf, area B, thickness s
- 1 Polystyrene lid, U-value Ul, area B, thickness s1
- Convective layers, heat transfer coefficient h, not active on the floor of the box
- Outside region, temperature Tout

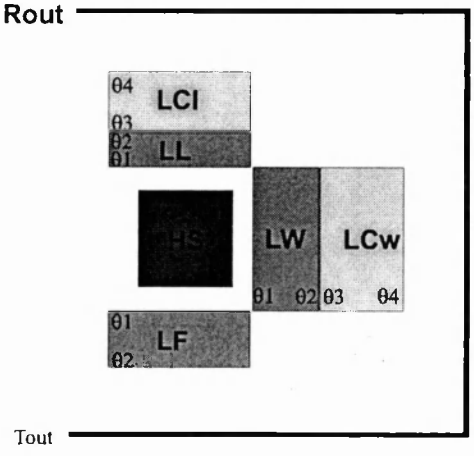


Figure 7H.2

The Mathematica implementation is:

```
LW = new[Layer, Uw,s,theta1,theta2,4A]
LCw = new[Layer, h,null,theta3,theta4,4A]
WALLS =  LinkLayer[LW, LCw]

LL = new[Layer, Ul,s1,theta1,theta2,B]
LCl = new[Layer, h,null,theta3,theta4,B]
LID =  LinkLayer[LL, LCl]

LF = new[Layer, Uf,s,theta1,theta2,B]
Rout = new[Region, Tout]
LinkRegion[WALLS,Rout,theta4]
LinkRegion[LID,Rout,theta4]
LinkRegion[LF,Rout,theta2]

HS = new[HeatSource,T[t],Tini,m,c,t]
Heq = LinkLayer[HS,{LID,LF,WALLS}]
eq = Equation[Heq]
initialCondition = InitialCondition[Heq]
SolveHeatEquation[Heq]
```

The following output shows the differential equation produced, and its solution.

```
-(c m T'[t]) ==
                        B (-Tout + T[t])    4 A (-Tout + T[t])
B Uf (-Tout + T[t]) + ---------------- + ------------------
                            1   1                1   1
                            - + --               - + --
                            h   Ul               h   Uw

{{T[t] ->
  (t (-(B Uf)-B/(1/h + 1/Ul)-(4 A)/(1/h + 1/Uw)))/(c m)
E
*(Tini - Tout) + Tout}}
```

Appendix 9P

Problem G: Gardening

The problem is to find the optimal density for planting plants in a plot, so as to maximise the total yield from the plot.

O-O Solution 1

The standard object model for a birth-death process, as described in Chapter 6, can be applied, and the task is to find suitable functional forms $f_B(t)$ and $f_D(t)$ for the birth and death processes.

Construct PopulationState(POP, P(t), t, P_0)

Construct Birth(B, $f_B(t)$, t)

Construct Death(B, $f_D(t)$, t)

LinkObjects(POP, B, D) → PopulationEquation(PE, Replace[f_B by P in $B(f_B(t), t)$], Replace[f_D by P in $D(f_D(t), t)$])

It is likely that there will be no births in this scenario, so that $f_B(t) = 0$. The choice for $f_D(t)$ is harder. Heuristics suggest that $f_D(t)$ should be a function of t only because the main factor influencing deaths is not the population at time t. The main factors are more likely to be water (too much or too little), pests, weather etc., all of which appear to be stochastic and time dependent. Thus, $f_D(t) = a+bt$ ($b>0$) seems sensible. With this type of problem, a standard optimisation method, which solves $dP/dt = 0$ for P or t is simple to provide. It may be hard to automate a decision whether to use P or t in software. Thus:

POP.DifferenceEquation → {P(t+dt) = P(t) - (a+bt) , P(0) = P_0}

POP.DifferentialEquation → {dP/dt) = - (a+bt) , P(0) = P_0}

POP.OptimiseCalculus → {Solve[dP/dt = 0, {t, P}]}

O-O Solution 2

An alternative approach is to treat the problem as purely stochastic on the basis that the initial population is essentially static, but subject to stochastic factors such as weather, water supply etc. The task is then to determine a suitable probability density function that models this stochastic process. This is largely a matter for data fitting and does not seem appropriate for O-O treatment.

Let the random variable P be the number of survivors at time t from an initial population P_0. Then Prob(P>Q) could be found from PDFs such as (one for each Q):

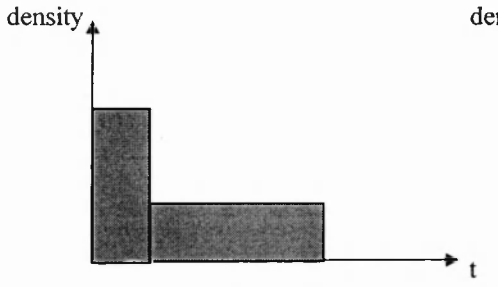


Figure 9P.G1

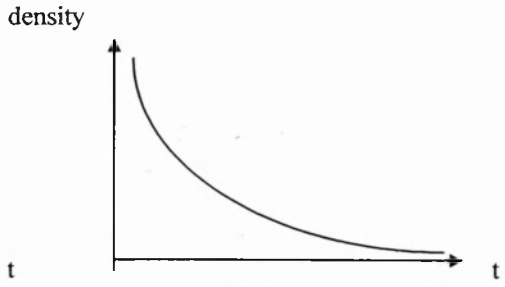


Figure 9P.G2

These model a high initial mortality rate.

Problem PB: Picnic box

A picnic box is to be constructed with an insulating layer of polystyrene. Its purpose is to keep its contents cool for a reasonable time. Determine a suitable thickness of polystyrene to do this.

The method of solution was outlined in Chapter 6, and is reproduced here with the addition of a HeatSource object and an ObjectiveFunction method of the HeatEquation object. This defines which variable (U) to express in terms of which others (initial and internal temperatures and time), and would be implemented in software by a Solve[] construct. To simplify this solution, the layers are combined into a composite layer with U-value U, which can be expressed in terms of whichever heat transfer coefficients are involved. One of these *must* involve the polystyrene thickness, which is then determinable from the value of U obtained from the model. For example, if the composite layer consists of polystyrene of thickness p and thermal conductivity u, with an outside convective layer with heat transfer coefficient h, then $U = (u/p + 1/h)^{-1}$. In Chapter 6, the corresponding U value was derived for a 3-layer composite, and the third layer is likely to be a plastic case. A HeatSource object models the water, and could also model a heat sink.

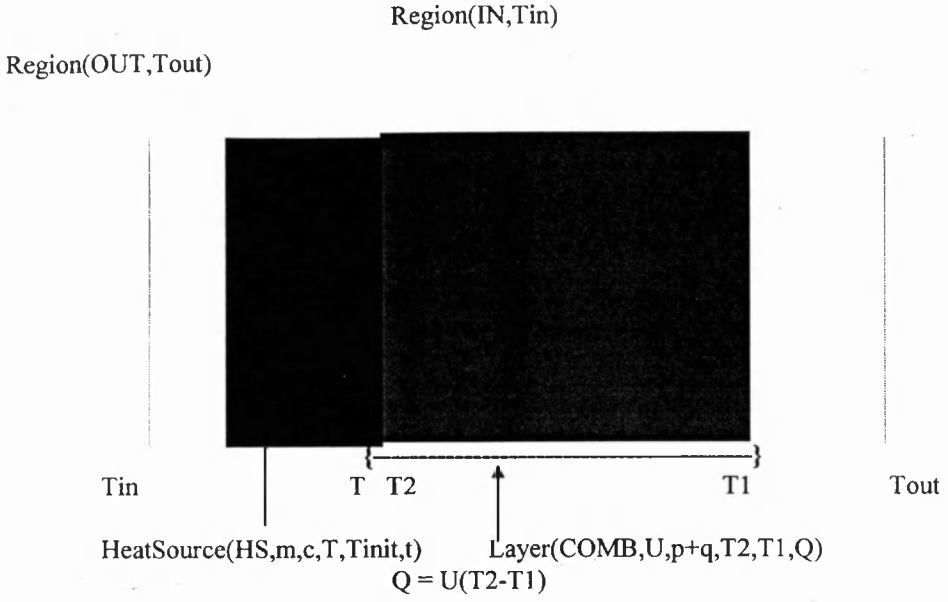


Figure 9P.PB1

Construct Region(OUT, Tout)

Construct Region(IN, Tin(t))

Construct HeatSource(HS,m,c,T(t),Tinit,t)

Derive or construct Layer(COMB,U,p,T2,T1,Q= U(T2-T1))

LinkObjects(IN, HS) → Layer(HS,m,c,Tin(t),Tinit,t)

LinkObjects(COMB,OUT) → Layer(COMB,U,p,T2,Tout,Q= U(T2-Tout))

LinkObjects(COMB,HS) → HeatEquation(HE,-mcD[Tin(t)-Tout,t], 6AU(Tin-Tout),{0,Tinit})

HE.GetHeatEquation → $-mc\frac{\mathrm{d}}{\mathrm{d}t}\left(T_{in}(t)-T_{out}\right)=6AU\left(T_{in}(t)-T_{out}\right)$

HE.ObjectiveFunction(U; Tin, Tinit, t) → Solve for U given Tin, Tinit and t, where t is the maximum time for the temperature to rise from Tinit to Tin.

Many subtle assumptions in this problem are implicit in the model. For example:

1. The heat source object occupies the entire interior of the box.
2. Contents = food = water, (easy to find numerical values for thermal properties of water).
3. There is no heat loss at the join of two or three walls of the box.
4. What constitutes a 'reasonable' time for food to stay cool is subjective.
5. The ObjectiveFunction method computes a function of the temperature profile defined by the derived heat equation. Another objective might be useful in a different context.

Problem F: Firebreak

In order to prevent excessive loss of forest through forest fires, firebreaks are to be installed in an area of forest. The task is to determine the spacing between adjacent firebreaks such that the yield from the forest is optimised.

The solution presented in Chapter 6 has a suitable formulation and objective function for this problem, and the difficulties with an O-O approach in this context were discussed there. In the following analysis, the idea of using a MultipleInstance object is pursued, and the strategy is to:

- construct 1 forest area (s by s)
- construct x vertical firebreak areas (each w by s)
- construct x(x+1) horizontal firebreak areas (each w by (s-wx)/(x+1))
- compute the total forested area: s^2 - (xws + (s-wx)xw) = $(s-wx)^2$

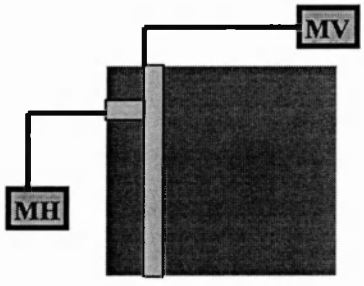


Figure 9P.F1

Construct Area(FOREST, Forested, s^2, 1)

Construct Area(FB_VERT, NotForested, ws, 1)

Construct MultipleInstance(MV, FB_VERT, x)

LinkObjects(FOREST, MV) → Area(FBV, NotForested, xws, 1)

Construct Area(FB_HORIZ, NotForested, A2, 1) where A2 = $\frac{(s-xw)w}{x+1}$

Construct MultipleInstance(MH, FB_HORIZ, x(x+1))

LinkObjects(FOREST, MH) → Area(FB_ALL, NotForested, xws + (s-wx)xw, 1)

LinkObjects(FOREST, FB_ALL) → Area(F1, Forested, s^2 - (xws + (s-wx)xw) ,1)

FOREST.DefineObjectiveFunction1(N) $\rightarrow A(x) = (s - wx)^2 - N\left(\frac{s - wx}{x+1}\right)^2$

FOREST.MaximiseAreaCalculus() → Solve A'(x) = 0,

OR

FOREST.MaximiseAreaDiscreteSearch(n) → Max[A(x), {i, 1, n}] (with a numeric *n*)

This strategy is much harder to implement than a simple computation of the total forested area in the rectangular grid represented by the MultipleInstance diagram in this section, using x horizontal firebreaks each of area ws, x vertical firebreaks each of area ws, and subtracting x^2 overlap areas each of area w^2 which have been counted twice. This gives the total forested area as $s^2 + 2wsx - w^2 x^2 = (s-wx)^2$. The DefineObjectiveFunction1 method is a heuristic which is tricky to formulate conceptually. In an O-O analysis there is the added complication of a cast into O-O terms. The result appears to be very specific and may not be more generally applicable.

Problem K: Kitchen Scales

The problem is to model the motion of a kitchen scale to ensure that when a weight is placed on the scale pan, the settling time for reading the weight is not too long.

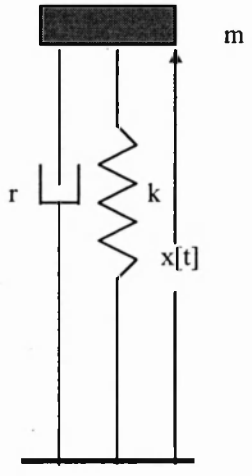


Figure 9P.K1

The model, based on Figure 9P.K1, is straightforward, and is similar to the spring-dashpot system from Chapter 6. It is difficult to propose a suitable objective function based on solving in terms of x(t). The idea in the objective function below is to find a value X such that |x(t)| < X for all t > T where T is a 'reasonable' settling time (e.g. 2 seconds). This is not as effective as the 'logarithmic decrement' for weak damping method, which many students used. It is possible to provide 'shortcut' methods for

weak damping, but these are very specific to the algebraic form of x(t) and are unlikely to be widely applicable.

Construct Particle(P, m, x[t], mg)

Construct Spring(S, k, a, x[t])

Construct Dashpot(D, r, x[t])

LinkObjects(P, S) → Force(T2, -k(x[t] - a))

LinkObjects(P, D) → Force(T2, -r x'[t])

LinkObjects(P, T1+T2) → EquationOfMotion(EoM, m x''[t] = -mg - rx'[t] - kx[t] - a))

EoM.SolveAnalytic() → x(t) = …(solve previous result)

EoM.ObjectiveFunction(k; r, x=X, t=T) → Express k in terms of other parameters

OR

EoM.ObjectiveFunction(r; k, x=X, t=T) → Express r in terms of other parameters

Problem S: Speed Bumps

The problem is to determine the spacing of speed bumps in a road such that the passage of a car over them is 'comfortable'.

The starting point for this problem is the spring-dashpot system from Chapter 6. There is a variable base line, a distance y(t) above the nominal 'road level'. y(t) is modelled as a sine function, and the analysis will be limited to determining a 'comfortable' approach speed, u. This ought to be slow and make it not worth exceeding the speed limit (otherwise excessive braking is involved). The actual spacing can then come from consideration of the kinematics of motion between bumps, but many assumptions about this motion must be made to derive a result. Two 'comfort' criteria seems reasonable, as below. Possibly, a requirement for critical damping may constitute a third.

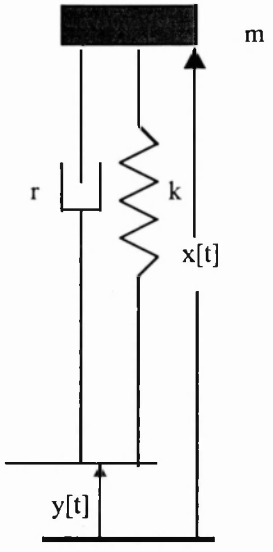


Figure 9P.S1

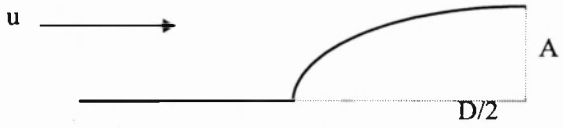


Figure 9P.S2

Construct Particle(P, m, x[t], mg)

Construct Spring(S, k, a, x[t] - y[t])

Construct Dashpot(D, r, x[t] - y[t])

LinkObjects(P, S) → Force(T2, -k(x[t] -y[t] - a))

LinkObjects(P, D) → Force(T2, -r (x'[t] - y'[t]))

LinkObjects(P, T1+T2) → EquationOfMotion(EoM,

m x''[t] = -mg - r(x'[t] - y'[t]) - k(x[t] - y[t] - a))

EoM.SetParameter(y[t], Asin(wt)) → Sets the functional form of y to Asin(wt)

(D = uπ/w).

EoM.SetParameter(w, uπ/D) → Expresses w in terms of u and D.

'Comfort' Criterion 1: limit the acceleration

EoM.SolveAnalytic() → x(t) = … (solve exactly for x)

EoM.ObjectiveFunction(u; x''(t)=g/10) → Limit |x''(t)|,

solve for u in terms of other parameters

'Comfort' Criterion 2: limit the spring compression to prevent 'bottoming out':

'Comfort' Criterion 2: limit the spring compression to prevent 'bottoming out':

$|x(t) - y(t)| > Z$ for all t

EoM.SolveAnalytic() → x(t) = … (solve exactly for x)

EoM.ObjectiveFunction(u; x=Asin(wt) + Z) → Limit |x(t)-y(t)|,
solve for u in terms of other parameters

Problem S: Speed Bumps: alternative solution

This solution is based on a speed/time diagram, and is purely kinematic. There is no model of the car suspension. In this model, the car accelerates from a minumum speed at which the bump has to be negotiated, to the maximum allowable speed. It reaches this speed when it gets to the next bump. The known parameters are U, Vmax and the car's acceleration, A. The distance s is unknown.

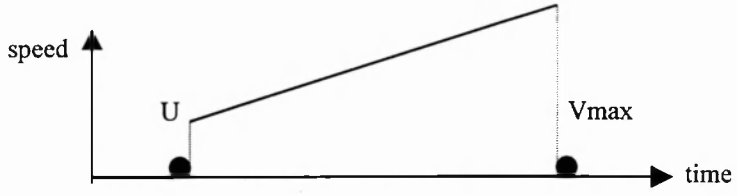


Figure 9P.S3

Construct ConstantAccelerationProfile(VP, {x,x',x'',t}, {U,Vmax,A})

VP. ConstantAccelerationFunction(s; U,Vmax,A) → $s = (Vmax^2-U^2)/(2A)$ (A constant)

Both the O-O and the 'traditional' models are very simple, and the former does not advance the latter significantly, except to allow a way of computerisation. A more fundamental approach is to use the kinematic methods of the Particle object. This solves the problem, but asks for a diagram which is unlikely to appear in practice. The analysis requires a constant force (of magnitude m A, where m is the mass of the particle) so that constant acceleration, A, results.

Construct Particle(P, m, x[t],t)

Construct Force(F, m A)

LinkObjects(P,F) → EquationOfMotion(EoM, m vdv/dx, m A))

EoM.EquationOfMotion() → x'' = A

EoM.ObjectiveFunction(s; U, Vmax, A) → Solve x'' = A,
obtain s in terms of U, Vmax and A.

Problem W: Washing up

The problem is to determine the maximum of plates which can be washed up in a sink before the water temperature drops too low to be useful.

To model this situation, a diagram of a sink is drawn, containing water (a heat source) and n plates. The sink is assumed to be cubic with five faces of area A, and there is a convective layer where the water is in contact with the air. Considerable care must be taken when drawing the diagram to stress what is in contact with what. The sink, convective layer and the plate (initially) are all 'outside'. The OUT Region wraps around them to show this. In addition, the Region objects contain an additional attribute: area.

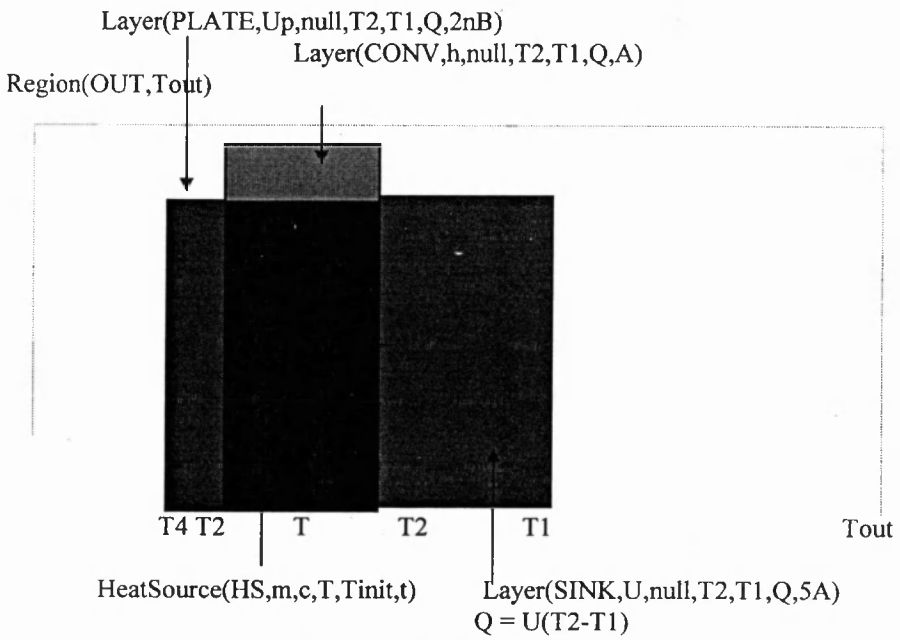


Figure 9P.W1

Construct	Region(OUT, Tout)
Construct	Region(IN, Tin(t))
Construct	HeatSource(HS,m,c,T(t),Tinit,t)
Construct	Layer(SINK,U,null,T2,T1,Q= U(T2-T1), 5A)
Construct	Layer(CONV,h,null,T2,T1,Q= k(T2-T1), A)
Construct	Layer(PLATE,Up,null,T2,Tp,Q= U(T2-T4), 2nB)

LinkObjects(SINK,OUT) → Layer(SINK,U,null,T2,Tout,Q= U(T2-Tout), 5A)

LinkObjects(CONV,OUT) → Layer(CONV,h,null,T2,Tout,Q= h(T2-Tout), A)

LinkObjects(PLATE,OUT) → Layer(PLATE,Up,null,T2,Tout,Q= Up(T2-Tout), 2nB)

LinkObjects(CONV+SINK+PLATE,HS)

→ HeatEquation(HE,-mcD[T(t)-Tout,t],

5AU(T-Tout)+hA(T-Tout)+2nBUp(T-Tout),{0,Tinit})

HE.GetHeatEquation →

$$-mc\frac{\mathrm{d}}{\mathrm{d}t}\left(T(t)-T_{out}\right)=6AU\left(T(t)-T_{out}\right)+Ah\left(T(t)-T_{out}\right)+2nBU_p\left(T(t)-T_{out}\right)$$

HE.ObjectiveFunction(n; T, Tinit, t_{max}) → Solve for n given T, Tinit and t_{max}, where t_{max} is the maximum time for the temperature to fall from Tinit to T.

There are two main problems with this approach. The first is to link the appropriate objects, as shown on the diagram. The second is to provide the appropriate parameters for the objects. The model assumes that *n* plates in the water at once is equivalent to replacing each by the next in turn.

In an alternative formulation, the plate acts as a heat sink (modelled by a HeatSource object). In the diagram above, the object HeatSource(HSPLATE,mp,cp,T,Tout,t) replaces the Layer object. There is an instantaneous energy exchange when it is placed in the water which results in a sudden drop in the water temperature. This is modelled by linking the two HeatSource objects, HS (the water) and HSPLATE, and the result modifies the initial temperature of HS (Tinit to T#). There after, Newtonian cooling applies for the time, t_p, that the plate is in the water. The objective function then computes the water temperature, $T(t_p)$. This deals with one plate. Subsequent iterations then model the addition of more plates, each with the replacement Tinit ▯T#, until the water temperature reaches a lower bound T_L. The solution to the problem is then the total number of iterations.

Construct	HeatSource(HSPLATE,mp,cp,T,Tout,t)
Construct	Region(OUT, Tout)
Construct	Region(IN, Tin(t))
Construct	HeatSource(HS,m,c,T(t),Tinit,t)
Construct	Layer(SINK,U,null,T2,T1,Q= U(T2-T1), 5A)
Construct	Layer(CONV,h,null,T2,T1,Q= k(T2-T1), A)

LinkObjects(HSPLATE,HS) → HeatSource(HS,m,c,T(t),T#,t)

where T# is given by mc (T-T#) = mp mc (T#-Tout)

LinkObjects(SINK,OUT) → Layer(SINK,U,null,T2,Tout,Q= U(T2-Tout), 5A)

LinkObjects(CONV,OUT) → Layer(CONV,h,null,T2,Tout,Q= h(T2-Tout), A)

LinkObjects(CONV+SINK, HS)

→ HeatEquation(HE,-mcD[T(t)-Tout,t],

5AU(T-Tout)+hA(T-Tout),{0,T#})

HE.GetHeatEquation →

$$-mc\frac{\mathrm{d}}{\mathrm{d}t}\left(T(t)-T_{out}\right)=6AU\left(T(t)-T_{out}\right)+Ah\left(T(t)-T_{out}\right);\quad T(0)=T_{\#}$$

HE.ObjectiveFunction(T; T#, t_p) → Solve for T in terms of t=T# and t_p,

where t_p is the time a plate is in the water.

This O-O formulation corresponds more to what might actually happen, but accounts for one iteration only. To do further iterations requires a formal destruction and reconstruction of the HS and HSPLATE objects. Although this is tedious, it is not much more tedious than solving the ODE for each iteration. The objective function for an O-O or a non-O-O formulation suffers from the problem that the solution cannot be obtained without actually doing the iterations. There might be many iterations, especially if certain brands of washing-up liquid are used!

Problem CP: Car purchase

A decision has to be made whether to buy a new car or to continue to maintain an old one. The task is to find the optimal age of an old car when it is replaced. It is common, in this type of problem, to model the depreciation on the old car and the cost of maintaining it. The total cost is the sum of these two terms. It is possible to use a standard cash flow construction (n = 12 to 72 months typically), with an added Costs method.

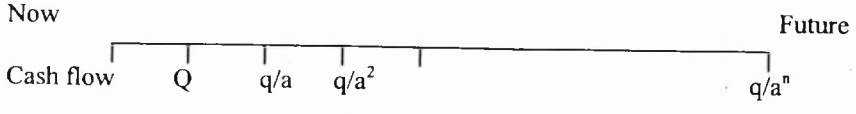


Figure 9P.CP1

Create CashFlow(CF, Q, q, a=1+m/100, n)

CF. NPV(r,Q,q, a) → $qa^{-r} - Q$ (r <= n)

CF.Costs(c, r) → c(r+1) where c is a cost function, probably increasing with time, as in Figure 9P.CP1.

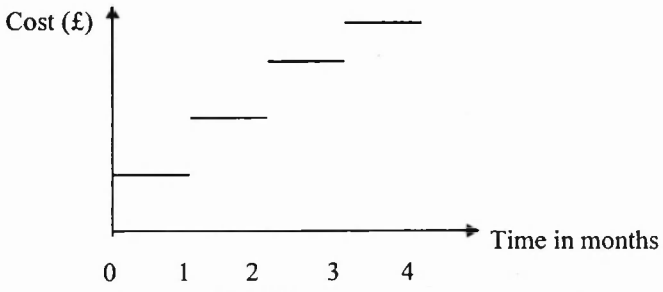


Figure 9P.CP1

Then calculate Min[CF. NPV(r,Q,q, a) + CF.Costs(c, r) , {r, 1, n}]

An alternative to this discrete formulation is to use continuous versions of the depreciation and cost methods:

NPV(t, q, a) = $q e^{-at}$

Costs(t) = ct + b (c, b > 0)

In this case a minimisation by calculus would replace the discrete minimisation above.

A further alternative is to model "Costs" (other than depreciation) as a separate object.

```
Cost{c(t), t
        public:
         GetCost(t)   // returns c(t)
    }
```

Construct Cost(C, c+bt)

LinkObjects(C, CF) → EquationOfState(EoS, C.GetCost(t), CF.NPV(t,q,a))

EoS. EquationOfState → $q e^{-at} + (ct + b)$

EoS.Minimise → Solve[$q e^{-at} + (ct + b) = 0$, t]

Appendix 9PJ

A parachutist jumps from an aircraft, freefalls for a time t_f, and then opens the parachute. Calculate the minimum altitude of the aircraft in order for the parachutist to land safely.

This is an example of a problem in which it is essential to supply initial conditions.

Freefall part

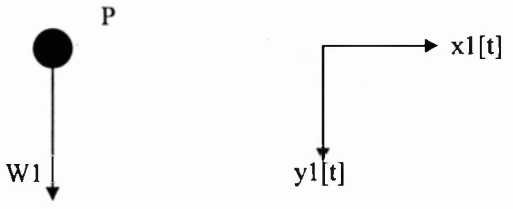


Figure 9P.PJ1

Construct Particle(P, m, {x1[t], y1[t]}, mg)

P.MakeWeight() → Force(W1, {0, mg})

LinkObjects(P, W1) → EquationOfMotion(EoM1,

mx1''[t] = 0,

my1''[t] = mg},

{x1(0) = 0,

x1'(0) = U,

y1(0) = 0,

y1'(0) = 0 })

EoM.SolveAnalytic() → {x1[t] = …, y1[t] = …, x'1[t] = …, y1'[t] = …}

EoM.ObjectiveFunction(Vy, Hy,Vx,Hx;

Hy = $y1(t_f)$,

Vy = $y1'(t_f)$,

Hx = $x1(t_f)$,

Vx = $x1'(t_f)$)

→ Find vertical & horizontal distance and speed at freefall time, t_f

Parachute Open part

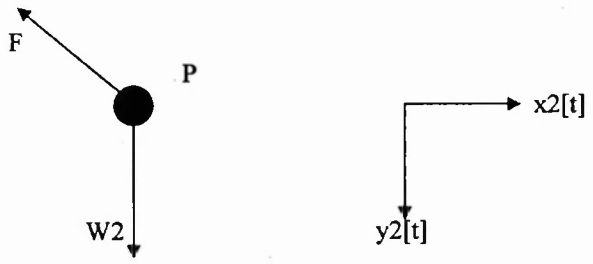


Figure 9P.PJ2

Construct Particle(P, m, {x2[t], y2[t]}, mg)

Construct Force(F, {$-kU^2$, $-h(y2'[t])^2$})

P.MakeWeight() → Force(W2, {0, mg})

LinkObjects(F, W2) → Force(AllForces, {$-kU^2$, mg - $h(y2'[t])^2$})

LinkObjects(P, AllForces) → EquationOfMotion(EoM2,

m x2''[t] = $-kU^2$,

m y2''[t] = mg - $h(y2'[t])^2$},

{x2(0) = Hx,

x2'(0) = Ux,

y2(0) = Hy,

y2'(0) = Vy })

EoM.SolveAnalytic() →

{x2[t] = …, y2[t] = …, x'2[t] = …, y2'[t] = …}

EoM.ObjectiveFunction(y2'[H-Hy]) →

Find vertical speed at ground level, y2 = H-Hy

Appendix 9S

Detailed validation results

Problem/ Total/ TMA	Script	Category	Score	Reason
PB	1	R1	1	O-O model would relate heat flows through walls
1		M1	1	O-O would have focussed on objective function
12		M3	-1	Good solution: O-O would not have added
PB	2	F3	1	Excessive irrelevant features would not appear
2		R1	-1	Well-defined model, as good as O-O
15		M2	1	Unclear strategy: O-O would provide dependent and independent variables
		O1	1	Well-defined diagram would have defined objects for O-O analysis automatically
PB	3	F1	1	O-O would have captured thermal features
4		R2	1	O-O model would not assume linear temperature gradient, which contradicts assumptions
15		A2	1	Newtonian heating would be explicit in O-O analysis
		O1	1	Diagram was drawn with heat flows: would have provided links to advance model
PB	4	F1	1	O-O would have captured time and energy features
7		F3	1	O-O model would not permit repeated features
12		R1	1	Cannot assume result in O-O analysis ('thickness proportional to time')
		R2	1	O-O analysis would not produce linear model with features in script
		M2	1	O-O provides optimisation/solution strategies
		A2	1	rate of heating' would be explicit: ill-defined in script
		O1	1	Diagram was drawn with heat flows in/out: would have defined Region and Layer objects
PB	5	F1	1	Missed time, temp in/out, thermal properties: these are supplied by O-O
6		F4	1	Variables would not be confused with constants
10		R1	1	Result cannot be assumed with O-O, and features are necessarily related
		R2	1	O-O temperatures are well-defined
		M1	1	O-O would have focussed on objective function
		A2	1	O-O necessarily relates heat flows in all parts of the system
F	6	F1	-1	Stochastic features not covered by O-O
-4		F2	-1	Scripts has richer features list than O-O can supply
15		F3	-1	Multiple instances not covered by O-O
		R4	-1	Simple formulation superior to that of O-O
		M2	1	Would have looked at search solutions supplied by O-O methods
		O1	-1	O-O does not cope well in calculating non-overlap areas

Problem/ Total/ TMA	Script	Category	Score	Reason
PB	7	F1	-1	More complex model than O-O template allows
-3		F4	0	?? Adjustments made for undefined symbols would be automatic with O-O
16		R2	-1	Well-defined matrix formulation: not covered by O-O template
		R4	1	Over-complication by 'transfer coeffs' would not happen with O-O
		M2	-1	Well-defined alternative strategy: O-O need not compete
		O1	-1	Good use of alternative diagrams: O-O need not compete
				Overall: well-defined formulation, not aided by O-O
F	8	F1	1	O-O would have supplied 'area' concept
2		F2	1	O-O variables necessarily correspond to features
6		R1	-1	Well-defined model: O-O model less understandable
		M1	1	O-O would have focussed on objective function
PB	9	F1	1	O-O would have captured thermal features
4		R1	1	O-O model would relate heat flows through walls
9		M1	1	O-O would have focussed on objective function
		A1	1	Assumptions implicit in O-O model
				Sound script, unable to start with 'relations' stage
F	10	F3	1	O-O would identify material features
0		R1	-1	Well-defined model: O-O model less understandable
14		M2	1	O-O would provide methods for objective
		O1	-1	Analysis based partly on diagrams, but still unable to develop. Hard with O-O: unlikely to improve formulation
				High TMA score rewarded peripheral activities
F	11	F1	-1	Full list: O-O could not add
0		R2	1	Same relations as O-O but unable to form equation
12		M1	1	Model expressed in words only. O-O necessarily forms equations
		O1	-1	Very hard O-O formulation unlikely to help
F	12	F2	1	O-O variables necessarily correspond to features
0		F3	1	O-O would concentrate on essential features only
7		R1	-1	Useful diagrams provide easy non-O-O model
		M1	1	O-O would suggest that an objective is necessary
		O1	-1	Completely wrong geometrical concepts in places: O-O could not help
		O1	-1	Script demonstrates negative attitude to modelling: no method with subjective elements likely to help
PB	13	F1	-1	Good features list: O-O would not have enhanced
0		R1	-1	Derived same type of equation as O-O: features related well
9		R2	1	O-O would not make error in heat transfer equation
		A1	1	Used internal dimensions = external dimensions: not possible with O-O analysis

Problem/ Total/ TMA	Script	Category	Score	Reason
K	14	F1	-1	Same features list as O-O
-1		R2	1	O-O would not confuse displacements from equilibrium and another fixed point
15		R5	1	Used all damping cases incorrectly: O-O does not consider these
		M1	-1	Clear strategy based on damping ratio: not in O-O formulation
		M3	0	Easy solution but did not consider initial conditions: O-O requires input of these
		O1	-1	Hard to formulate objective in O-O terms: weak damping is too specific
K	15	F1	-1	Fuller features than O-O, including circular motion of dial
0		R2	1	O-O would not confuse displacements from equilibrium and another fixed point
13		M1	1	O-O would provide objective
		O1	-1	Hard to formulate objective in O-O terms: weak damping is too specific
K	16	F1	0	O-O could not add significantly
1		R2	0	O-O would not confuse displacements from equilibrium and another fixed point
11		M2	-1	Hard for O-O to improve on strategy
		M3	1	Initial conditions would be used by O-O
		O1	1	Diagram would result in model
S	17	F1	-1	Full list, with kinematic quantities
-1		R1	-1	Good formulation of undamped model: O-O could not improve
16		R3	1	O-O cannot mismatch features with variables
		M1	1	O-O attempts to relate shape of bump with approach speed by 'Substitute' method
		O1	-1	Hard to link O-O model with kinematic model
S	18	F1	-1	Fuller list than O-O
-3		R1	-1	Good formulation: O-O could not improve
19		R2	1	O-O would not ignore initial conditions (needed for objective)
		M1	-1	Reasonable attempt to consider Reaction as comfort criterion
		O1	-1	Hard to link O-O model with kinematic model
S	19	F1	1	O-O would express in terms of parameters
3		R2	1	O-O would not assume wrong equation of motion
14		M1	1	O-O would provide strategy (not based on resonance)
		A1	1	O-O would not model rough road
		O1	-1	Hard to link O-O model with kinematic model

Problem/ Total/ TMA	Script	Category	Score	Reason
S	20	F2	1	Confusion over spring/dashpot properties
2		F3	1	Many irrelevant features: not captured by O-O
17		R1	-1	Good formulation: O-O could not improve
		M2	1	Wrong strategy to find w: O-O stresses need to substitute for w
		A1	1	Confused atempts to introduce other features: O-O concentrates on important ones
		O1	-1	Hard to link O-O model with kinematic model
CP	21	F3	1	Many unquantifiable features
5		R2	1	O-O gives correct discounted values
14		R2	1	O-O would not subtract costs
		M3	1	O-O would provide a method for minimisation
		A2	1	Assumptions did not relate to features modelled: some O-O assumptions implicit in features
CP	22	F1	1	O-O provides functional forms for depreciation and costs
6		F3	1	Many features not quantified
13		R1	1	Unable to relate any features
		R2	1	O-O would clarify concept of depreciation
		M1	1	Unable to start: O-O would provide objective
		A2	1	Assumptions did not relate to features modelled: some O-O assumptions implicit in features
CP	23	F3	1	O-O would isolate important features
6		R1	1	Unable to relate any features
10		R3	1	Many irrelevant unused features
		M1	1	Unable to start: O-O would provide objective
		A2	1	O-O assumes forms for cost and depreciation functions
		O1	1	Diagrams of cost and depreciation v. time would have provided clues for suitable functional forms
CP	24	F1	1	O-O would capture NPV features
4		R2	1	Wrong symbols in depreciation term
14		R2	1	O-O provides Total cost = Costs + Depreciation relation
		M2	1	O-O would provide a method for minimisation
CP	25	F1	1	O-O would capture depreciation features
5		R1	1	O-O would relate depreciation and costs
12		M1	1	O-O would provide objective
		A2	1	O-O assumes form for cost function
		O1	1	Diagrams would provide model for depreciation
CP	26	F1	-1	O-O would not improve
-2		R1	-1	O-O would not improve
17		R2	1	some dubious formulations: hard to assess
		M1	-1	clearly stated objective, same as O-O

Problem/ Total/ TMA	Script	Category	Score	Reason
CP	27	R2	1	Method unclear: O-O would clarify
2		M2	1	Math. formulation of objective contradicts text: one only with O-O
15				
CP	28	R1	-1	Good formulation without O-O
0		R2	1	Discrete formulation unclear: O-O would clarify
18		M1	-1	Clear objective stated
		M3	1	O-O would not have continuous solution method with discrete formulation
CP	29	F3	1	Too many features, few used: O-O captures material ones
4		R2	1	O-O requires initial value for iteration
13		R2	1	O-O can incorporate depreciation term correctly in discrete formulation
		M2	1	O-O has clear objective
CP	30	R2	1	O-O requires explicit functional forms
4		R2	1	O-O incorporates depreciation term
13		R3	1	Not all features used: O-O would prompt need for additional methods
		M2	1	doubt about how to analyse equation of state: O-O can clarify
CP	31	F2	1	O-O would provide sufficient variables
4		F3	1	Many irrelevant features: O-O concentrates on important ones
13		R2	1	O-O provides workable depreciation term
		M2	1	O-O would provide a method for minimisation
CP	32	F1	-1	O-O would not improve
2		R2	1	O-O would provide a correct formulation for depreciation
12		R2	1	O-O would provide a correct (linear) formulation for maintenance
		M3	1	Unclear how to proceed: O-O would provide suitable methods
CP	33	F1	-1	O-O would not improve
-2		R1	-1	O-O would not improve
16		R2	1	O-O would provide correct term for d(Depr)/dt
		M1	-1	O-O would provide strategy
G	34	F3	1	O-O would not present excessive features
2		F3	-1	Pruned features list OK: O-O would not have helped
10		R2	1	O-O would have prevented wrong formulation of input/output principle
		M2	1	O-O would have provided solution strategy through an optimisation method
		D1	1	O-O would focus on dependent and independent variables for data fitting
		D1	-1	O-O would not contribute to the actual fitting process, which was done well

Problem/ Total/ TMA	Script	Category	Score	Reason
G	35	F1	1	Missed time and Population(t): supplied by O-O
4		R2	1	Dubious Logistic model: no births
13		M1	1	O-O provides optimisation variables and methods
		A1	1	O-O cannot provide contradictory assumptions: e.g. logistic model and Population proportional to t
PB	36	F1	-1	Same features list as O-O
2		R2	1	Temperatures would be explicit in O-O formulation: harder to confuse
15		M2	1	Unclear objective with confused symbols:O-O needs parameters for Solve, less open to confusion
		M2	1	O-O uses explicit Solve, not an undefined search
F	37	F2	1	O-O would concentrate on geometric features: too many marginal features here
2		R2	1	Dimensional error in fire outbreak frequency: O-O requires number of damaged regions
18		M2	-1	Discrete search same as O-O
		O1	1	Diagram would result in model
F	38	F1	-1	Has non-O-O features which could lead to a more complex model (but don't)
-3		M1	-1	Same objective as O-O
18		M2	-1	Analysis of number of areas lost cannot be done by O-O
				Significant tutor input with main ideas in this model
F	39	F1	-1	Same features list as O-O
0		R1	-1	Same analysis as O-O
18		M2	1	Incomplete strategy: O-O could complete
		D1	0	?? Partly a non-algebraic model: substituted data too early. O-O requires algebraic parameters
		O1	1	Diagram would result in model
F	40	F1	-1	Has useful but undeveloped features: not in O-O formulation
2		F2	1	O-O would isolate key features
14		R1	-1	Same as O-O formulation
		R2	1	Dimensional error: O-O uses areas and is necessarily dimensionally right
		M2	1	O-O would suggest clear strategy
		O1	1	Diagram would result in model
PB	41	F2	1	O-O requires variable Temperature
3		R2	1	O-O would relate heat flows to contents and through walls
12		O1	1	2 Regions + 1 Layer on diagram would have produced an O-O model
PB	42	F1	-1	Good features list: O-O would not have enhanced
-3		R1	-1	Derived same equation as O-O
18		R1	-1	Same as O-O objective

Problem/ Total/ TMA	Script	Category	Score	Reason
PB	43	R1	-1	Derived same equation as O-O
1		M2	1	Treats thickness as a 'variable constant'. O-O clarifies objective by requiring arguments to Solve
15		A1	1	Assumed thickness, wants to calculate thermal conductivity: O-O requires variable thickness
PB	44	F1	1	O-O requires variable Temperature
4		F3	1	O-O would concentrate on material features
12		R2	1	O-O would relate heat flows to contents and through walls
		M1	1	O-O would provide method + strategy for objective
PB	45	F1	-1	Same features list as O-O
0		R1	-1	Derived same equation as O-O
15		M2	1	O-O would clarify inputs for objective function
		M2	1	O-O could not confuse non-linear model with linear objective
PB	46	R1	-1	Derived same equation as O-O
1		M1	1	O-O would provide method for formulating objective
16		O1	1	2 Regions + 1 Layer on diagram would have produced an O-O model
K	47	F3	-1	Only used important features
-2		F3	1	Many non-material features listed
16		R1	-1	Same analysis as O-O
		M1	-1	Objective based on ratio of successive displacements: more efficient than O-O
		M3	-1	Useful short cuts used: O-O could not do this
		O1	1	Diagram would result in model
K	48	F1	-1	Same features list as O-O
-3		R1	-1	Same analysis as O-O
17		M2	-1	Clear strategy based on damping ratio: not in O-O formulation
		M3	-1	Useful short cuts used: O-O could not do this
		O1	1	Diagram would result in model
W	49	F1	-1	O-O could not add
0		R1	-1	Good formulation: different features to O-O
17		R2	1	Doubtful treatment of plate/water heat exchange
		M2	1	O-O would account for plate/water heat exchange with heat loss elsewhere
		O1	1	Diagram would have produced an O-O model
		O2	-1	O-O formulation as difficult conceptually as non-O-O, but aided by diagram
W	50	F1	-1	O-O could not add
-1		R2	1	O-O would account for plate/water heat exchange with heat loss elsewhere
20		O1	-1	O-O formulation as difficult conceptually as non-O-O, but aided by diagram

Problem/ Total/ TMA	Script	Category	Score	Reason
F	51	F2	1	O-O models principal features: main features not modelled in script
3		R2	1	O-O would relate areas correctly
11		M1	1	O-O would provide objective
		D1	1	Data used as 'assumptions': O-O requires symbolic parameters
		O1	-1	Difficult O-O formulation
F	52	R2	1	O-O would provide dimensionally correct relations
3		R2	1	O-O would relate other features correctly
6		M1	1	O-O would provide objective
		A2	1	Written discription contradicts math. description
		O1	-1	Difficult O-O formulation
				Considerable tutor input required
F	53	F2	1	O-O would concentrate on main features
4		R2	1	O-O would relate forested and unforested areas
11		R3	1	Must justify incorporaton of features in O-O model
		M1	1	O-O would provide objective
		O1	1	Diagram would result in O-O model
		O1	-1	Difficult O-O formulation
S	54	F2	1	Mix of features based on kinematic and spring/dashpot: O-O requires all to be modelled
4		R1	1	no model: O-O would relate features
5		M1	1	O-O would provide objective
		A1	1	Unrealistic assumptions used as 'solutions'
				Considerable tutor input required
PB	55	F1	1	O-O has time, rates of heat flow
5		F2	1	O-O must associate features with variables
13		R1	1	O-O would relate heat flows and use symbols
		R1	1	Contradiction between linear model and non-linear ODE not possible with O-O
		M1	1	unable to start: O-O would provide objective
K	56	F2	1	O-O would associate dashpot with 'friction'
2		R1	-1	same model as O-O
12		R2	1	O-O requires initial conditions
		M1	1	unable to start: O-O would provide objective
PB	57	F1	1	O-O has thermal quantities, time, temperature
4		R1	1	O-O cannot mix linear and non-linear formulations
10		R1	1	O-O requires justification for link object procedures: attributes must be used
		M1	1	unable to start: O-O would provide objective
CP	58	F1	-1	O-O cannot add
-3		R1	-1	as O-O model but with alternative depreciation and maintenance models
18		M1	-1	O-O cannot add

Problem/ Total/ TMA	Script	Category	Score	Reason
PB	59	F2	1	O-O has time, temperature
2		R2	1	O-O relates heat flows in system components
13		A1	0	?? states that rate of temperature rise is assumed constant, but this is not used
PB	60	F2	1	O-O has thermal quantities, time, temperature
3		R2	1	O-O relates heat flows in system components
14		R2	1	O-O requires variables in heat flow equations: not only constants
F	61	F1	-1	O-O cannot add
-3		R1	-1	clear non-O-O model
19		M1	-1	O-O cannot add
		M3	-1	O-O cannot add
		O1	1	Diagram would have produced O-O model
				Very competent non-O-O model

Appendix 9C

I supplied Control Scripts (C) as outline solutions for TMA 4 problems.

Problem/ Total	Script	Category	Score	Reason
SB/Spring-dashpot	C19	F1	-1	Same features as O-O
1		R1	-1	Same diagram as O-O model, giving the same equation of motion
		R1	1	Easier to define lengths correctly in non-O-O formulation
		R2	1	Easier to use O-O 'toolkit' and avoid error
		M1	1	objective function insufficiently well defined: must do this in O-O analysis
		M2	-1	Hard to define O-O objective, particularly if multi-stage model used
		M2	1	O-O forces consideration of inputs/outputs to objective function
CR/continuous	C24	F1	-1	Hard for O-O to capture unusual/varied features in a standard O-O model
-3		R1	-1	Same formulation as O-O model
		M1	-1	Same calculus solution/objective as O-O analysis
CR/discrete	C32	F1	-1	Hard for O-O to capture unusual/varied features in a standard O-O model
-1		R1	-1	Same formulation as O-O model
		M1	1	O-O focusses on discrete/continuous objectives since both are present
G	C35	F1	1	O-O is more careful to consider the functional form of birth/death processes
-1		R1	-1	Same (logistic) formulation as O-O
		M1	1	Same calculus solution/objective as O-O analysis
PB	C44	F1	-1	Same features as O-O
-2		R1	-1	Same formulation as O-O model
		M1	-1	Same type of objective as O-O analysis
		M2	1	Objective strategy less efficient than O-O: found the time to rise to a given temperature for various thicknesses
F	C52	F1	-1	Hard for O-O to capture unusual/varied features in a standard O-O model
-4		R1	-1	Same geometric formulation as O-O
		M1	-1	Same calculus/discrete search methods as O-O analysis
		O1	-1	Very difficult to formulate O-O model
F	C54	F1	-1	Same features as O-O
-2		R1	-1	Same formulation as O-O model
		M1	-1	Same type of objective as O-O analysis
		M2	1	Precise inputs/outputs to objective function not considered

Problem C62 was a solution supplied at a tutorial, not a control script.

PJ	C62	F1	-1	Same features as O-O
0		R1	-1	Same formulation as O-O model
		R2	1	Easier to use O-O 'toolkit': avoids error
		M1	1	Objective not considered in detail: O-O forces this
		O1	-1	Harder to link stages of a 2-stage model
		O1	1	Forces consideration of 2-D motion

Results summary

Problem	Script	Score		
SB/Spring-dashpot	C19	1	m	-1.5
CR/continuous	C24	-3	sd	1.603567
CR/discrete	C32	-1	n	8
G	C35	-1		
PB	C44	-2		
F	C52	-4		
F	C54	-2		
PJ	C62	0		

2-sample t-test based on experimental and control data

	Expt	Control
n	61	8
m	1.491803	-1.5
sd	2.766604	1.603567

Preliminary variance ratio test to establish equivalent variance of control and expt. populations
NH: var(expt) = var(control)
AH: var(expt) != var(control)

F = 1.725281
5% F(crit, n1=7,n2=60) = 2.17
F < F(crit), so accept NH

NH: expt pop. mean = control pop mean
AH: expt pop. mean > control pop mean

Unbiased estimators	mean	1.144928
of population parameters	var	7.123073

t = 2.981156
DF = 67
1% t(crit)~ 2.66
t > t(crit), so reject NH

Appendix 9S1

Blank score sheet

Script number Problem

TMA score

Reason

		Score
Features		
F1	Material features missed	
F2	Features list/variables mismatch	
F3	Excessive or insufficient features	
F4	Units/dimensions mismatch	
Relations		
R1	Cannot start/model pre-solved	
R2	Material error in formulation	
R3	Additional features/not all features used	
R4	Excessive complication	
R5	Non-relevant formulation	
Objective/Method of solution		
M1	No objective found	
M2	Confused strategy/wrong strategy	
M3	Inaccurate solution	
Assumptions		
A1	Contradict model	
A2	Present, not used/absent but used	
Data		
D1	Inappropriately used	
Other		
O1	.....	
	Total	

Appendix 9S2

Summary of scores per script, which forms the basis for subsequent statistical analysis.

Problem	Script	O-O Score	TMA Score	Sign(O-O Score)
PB	1	1	12	1
PB	2	2	15	1
PB	3	4	15	1
PB	4	7	12	1
PB	5	6	10	1
F	6	-4	15	-1
PB	7	-3	16	-1
F	8	2	6	1
PB	9	4	9	1
F	10	0	14	0
F	11	0	12	0
F	12	0	7	0
PB	13	0	9	0
K	14	-1	15	-1
K	15	0	13	0
K	16	1	11	1
S	17	-1	16	-1
S	18	-3	19	-1
S	19	3	14	1
S	20	2	17	1
CP	21	5	14	1
CP	22	6	13	1
CP	23	6	10	1
CP	24	4	14	1
CP	25	5	12	1
CP	26	-2	17	-1
CP	27	2	15	1
CP	28	0	18	0
CP	29	4	13	1
CP	30	4	13	1
CP	31	4	13	1
CP	32	2	12	1
CP	33	-2	16	-1
G	34	2	10	1
G	35	4	13	1

Problem	Script	O-O Score	TMA Score	Sign(O-O Score)
PB	36	2	15	1
F	37	2	18	1
F	38	-3	18	-1
F	39	0	18	0
F	40	2	14	1
PB	41	3	12	1
PB	42	-3	18	-1
PB	43	1	15	1
PB	44	4	12	1
PB	45	0	15	0
PB	46	1	16	1
K	47	-2	16	-1
K	48	-3	17	-1
W	49	0	17	0
W	50	-1	20	-1
F	51	3	11	1
F	52	3	6	1
F	53	4	11	1
S	54	4	5	1
PB	55	5	13	1
K	56	2	12	1
PB	57	4	10	1
CP	58	-3	18	-1
PB	59	2	13	1
PB	60	3	14	1
F	61	-3	19	-1

Appendix 9S3

z-test

M =	1.4918033
SD =	2.7666041 (unbiased estimator)
n =	61
z =	4.2114287

t-test

mean and SD as for z-test

t =	4.211429
Critical t, 60 DF	3.23 0.1% level

Appendix 9S4

Sign test, discounting zero scores

"=1	38 mu	26
"-1	14 sd	3.605551
	z	3.189526

Sign test, combining zero and negative scores

"= 1
"= -1
"= 0

Normal approximation

38 mu =	26
14 sd =	3.605551
9 z =	3.189526

Appendix 9S5

Analysis of Fault R1: unable to formulate model.
The following scripts had an R1 score.

Problem	Script	Score for category R1	Overall O-O Score
PB	1	1	1
PB	2	-1	2
PB	4	1	7
PB	5	1	6
F	8	-1	2
PB	9	1	4
F	10	-1	0
F	12	-1	0
PB	13	-1	0
S	17	-1	-1
S	18	-1	-3
S	20	-1	2
CP	22	1	6
CP	23	1	6
CP	25	1	5
CP	26	-1	-2
CP	28	-1	0
CP	33	-1	-2
F	39	-1	0
F	40	-1	2
PB	42	-1	-3
PB	43	-1	1
PB	45	-1	0
PB	46	-1	1
K	47	-1	-2
K	48	-1	-3
W	49	-1	0
S	54	1	4
PB	55	1	5
K	56	-1	2
PB	57	1	4
CP	58	-1	-3
F	61	-1	-3

Test of proportions

Based on a Bin(61,p) distribution with p = 33/61
NH: p = 0.540984 AH: p < 0.540984

Score			
+1	10	m	33
-1	23	sd	3.891984
0	0	z	2.440914

Appendix 9S6

ANOVA analysis of scores per problem

NH: mu(PB) = mu(CP) = ... AH: Not all means equal

Problem	n	sum(y)	sum(y^2)	var(y)
PB	19	43	225	127.6842
CP	14	35	211	123.5
W	2	-1	1	0.5
G	2	6	20	2
S	5	5	39	34
F	13	6	80	77.23077
K	6	-3	19	17.5
Totals	61	91	595	382.415

ANOVA Table

Source of variation	DF	Sum sq.	Mean sq.	F
Between samples	6	76.83092	12.80515	1.808188
Within samples	54	382.415	7.081759	
Total	60	459.2459		

Critical 5% F(6,60) = 2.25
Test statistic does not exceed critical value
Hence accept NH: All means equal

Appendix 9KW

Kruskall-Wallis test

Problem	PB	RankPB)	CP	Rank(CP)	WGS	Rank(WGS)	F	Rank(F)	K	Rank(K)
	1	25.5	5	56	0	19	2	33	-1	13
	2	33	6	59	-1	13	0	19	0	19
	4	49	6	59	2	33	0	19	1	25.5
	7	61	4	49	4	49	0	19	-2	10
	6	59	5	56	-1	13	-4	1	-3	5
	-3	5	-2	10	-3	5	2	33	2	22
	4	49	2	33	3	41	-3	5		
	0	19	0	19	2	33	0	19		
	2	33	4	49	4	49	2	33		
	3	41	4	49			3	41		
	-3	5	4	49			3	41		
	1	25.5	2	33			4	49		
	4	49	-2	10			-3	5		
	0	19	-3	5						
	1	25.5								
	5	56								
	4	49								
	2	33								
	3	41								

Rank Sums	677.5	536	255	317	94.5
Sample sizes	19	14	9	13	6
$Rank^2/n$	24158.22	20521.14	7225	7729.9	1488.4

H = 7.937593

Critical Chi2, 4 DF, 5% = 9.49

NH Means of all categories are equal
AH Means of categories are not all equal
Hence accept NH: All means equal

Appendix 10M

Townend (Townend 95) models a car suspension system which consists of: a wheel and axle, ¼ of the car's mass, and two spring-dashpot combinations (Figure 10M.1). The *SpringDashpot* objects produce *Force* objects when linked with *Particles*. Care needs to be taken to produce correct signs for the *Force* objects in a software implementation.

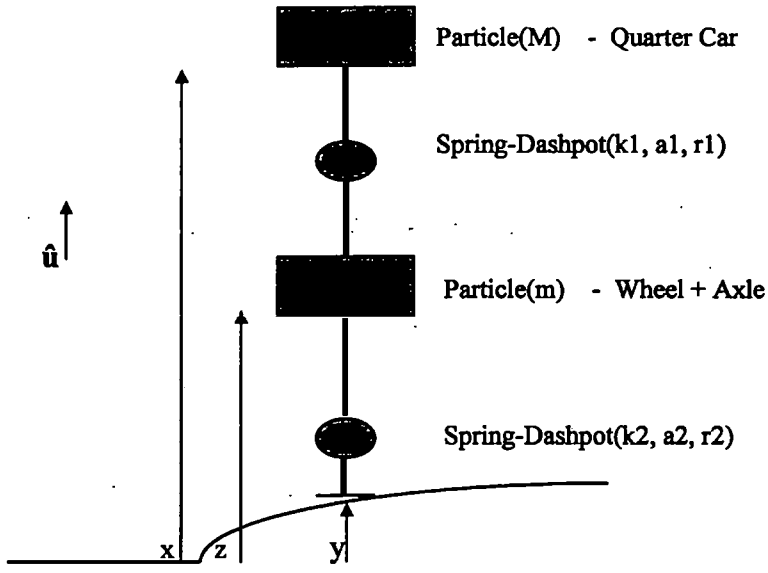


Figure 10M.1

Construct Particle(Car4, M, x)

Construct Particle(Wheel, m, z)

Construct SpringDashpot(SD1, k1, a1, r1, x, z)

Construct SpringDashpot(SD2, k2, a2, r2, z, y)

Construct GravitationalField(GF, g)

LinkObjects(Car4, GF) → Force(WtCar4, Mg, -û)

LinkObjects(Wheel, GF) → Force(WtWheel, mg, - û)

LinkObjects(Car4, SD1) → Force(F2, k1 (x - z - a) + r1 (x' - z'), - û)

LinkObjects(Wheel, SD1) → Force(FW1, -k1 (x - z - a) + r1 (x' - z'), - û)

LinkObjects(Wheel, SD2) → Force(FW2, k2 (z - y - a) + r2 (z' - y'), - û)

LinkObjects(Car4, F1+WtCar4) → EquationOfMotion(EoMCar, M x'' =
k1 (x - z - a) + r1 (x' - z') + Mg, - û)

LinkObjects(Wheel, F1+WtWheel) → EquationOfMotion(EoMWheel, m z'' =
-k1 (x - z - a) + r1 (x' - z') +
k2 (z - y - a) + r2 (z' - y') + Mg, - û)

References

(Abbott 83) Abbott, R. Program Design by Informal English Descriptions. Comm. ACM vol 26, 11. Nov 1983

(Abdali 86) Abdali,S., Cherry,G. and Soiffer,N. An object-oriented approach to Algebra System Design, in Proc. SYMSAC '86, Waterloo. ACM 1986

(Abelson 85) Abelson, H. and Sussman, G. Structure and Implementation of Computer Programs (p126). MIT Press, Cambridge MA. 1985

(Alabi 89) Alabi, B. A Model for the problem of Ground Vibration induced by the Wheels of a moving Train, in Applied Mathematical Modelling, Vol 13, Butterworth Publishers, Dec 1989.

(Andrews 76) Andrews J.G. and McLone R.R. Mathematical Modelling. Butterworths 1976.

(Aris 78) Aris,R. Mathematical Modelling Techniques. Dover 1978.

(Arney 91) Arney,D.C. Derive Laboratory manual for Differential Equations. Addison-Wesley 1991.

(Aspetsberger 84) Aspetsberger,K and Funk,G. Experiments with MuMath in Austrian High Schools, in SIGSAM Bulletin, 18,4 pp 4-7 (and ICME 5, Australia 1984). 1984

(Aspetsberger 89) Aspetsberger,K and Kutzler,B. Using a Computer Algebra System at an Austrian High School, in Proc. 6th Int Conf on Technology and Education, Orlando USA, (eds Collins, J and Estes,N.) March 1989

(Atkinson 81) Atkinson,A. Two Graphical displays for outlying and influential Observationsin Regression, in Biometrika 68 pp13-20. 1981

(Battye 97) Battye, A. and Challis, M. Deriving Learning Outcomes for Mathematical Modelling, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) (pp135-45) Albion Publishing, January 1997.

(Beare 96) Beare,R. Mathematical Modelling using a new spreadsheet-based system, in Teaching Mathematics and its Applications, volume 15, No. 3. IMA 1996

(Beck 89) Beck,K. and Cunningham,W. A Laboratory for teaching object-oriented thinking, in SIGPLAN Notices,vol 24, #10. ACM October 1989

(Beilby 93) Beilby, M. MathWise: a computer-based learning environment, in Maths and Stats vol. 4, #4, pp2-5. CTI Centre, Birmingham. Nov 1993

(Bellin 97) Bellin,D. and Simone,S. The CRC Card Book. Addison-Wesley 1997

(Bender 89) Bender,P. The Internal Rate of return of an Investment, in Applications and Modelling in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Berry 93), Berry,J., Graham, E. and Watkins, A. Learning mathematics through Derive. Ellis Horwood, 1993.

(Berry 95) Berry,J. and Houston,K. Mathematical Modelling. Edward Arnold 1995

(Berry 97) Berry,J. Investigating mathematics with the TI-92, in Int. J of Computer algebra in mathematics teaching (Berry,J. ed.) vol 4, #1. Research Information Ltd 1997

(Brown 91) Brown, D., Porta, H. and Uhl,J. Calculus and Mathematica. 1991 Addison-Wesley

(Blum 91) Blum, W. Application and Modelling in Mathematics teaching - A review of arguments and Instructional aspects, in Teaching of Mathematical Modelling and Applications (proc ICTMA-4, Denmark July 1989) (Niss,M, Blum,W. and Huntley,I. Eds) 1991 Ellis Horwood.

(Blum 91A) Blum, W. and Niss,M. Applied Mathematical Problem Solving, Modelling, Applications, and links to other subjects - State, Trends and Issues in Mathematics Instruction, in Educational Studies in Mathematics vol 22 pp 37-68. 1991

(Blum 95) Blum,W. Applications and Modelling in Mathematics teaching and Mathematics education - Some important aspects of Practice and Research, in Advances and Perspective in the teaching of Mathematical Modelling and Applications (Proc. ICTMA-6, Delaware, August 1993) Water Street Mathematics 1995

(Boekaerts95) Boekaerts, M., Seegers, G. and Vermeer, H. Solving Mathematical Problems: where and why does the solution process go wrong?, in Educational Studies in Mathematics (ed. Dörfler, W.) Vol 28, No. 3 (pp 241-262), April 1995, Kluwer, Boston.

(Bonadio 89) Bonadio,A. and Warren,E. Theorist Reference Manual. Prescience Corporation, 814 Castro St. San Francisco, CA. 1989

(Booch 86) Booch,G. Object-Oriented Development, in IEEE Trans. On Software Engineering, vol SE-12,#2. February 1986

(Booch 94) Object Oriented Analysis and Design. Addison Wesley 1994

(Booss-Bavnbek 91) Booss-Bavnbek,B. Against ill-founded irresponsible modelling, in Teaching of Mathematical Modelling and Applications (proc ICTMA-4, Denmark July 1989) (Niss,M, Blum,W. and Huntley,I. Eds) 1991 Ellis Horwood.

(Bostock 96) Bostock,L. and Chandler,S. Mechanics for A-level. Stanley Thornes 1996

(Bowtell 89) Bowtell, G., Haines, C. and Tanner, D. The teaching of mathematics at tertiary level through imperfect microcomputer simulations, in Applications and Modelling in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Brebbia 78) Brebbia, C.A. The Boundary Element Method for Engineers. Pentech Press 1978

(Brebbia 84) Brebbia, C.A., Telles, J. and Wrobel, L. Boundary Element Techniques. Springer Verlag 1984

(Brebbia 89) Brebbia, C.A. and Dominguez, J. Boundary Elements: an Introductory Course. Computational Mechanics 1989

(Bromilow 97) Bromilow,M. A new Multimedia Open University Course: Mathematical Methods, Models and Modelling, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Bronstein 92) Bronstein,M. Linear Ordinary Differential equations: breaking through the order 2 barrier, in Proc ISSAC 92, Berkeley,CA. (Wang,P ed.) ACM 1992

(Brown 91) Brown, D., Porta, H. and Uhl,J. Calculus and Mathematica. 1991 Addison-Wesley

(Brown 97) Brown, R. Mathematical Modelling and Current Events using Hand Held Graphing Technology, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Buchberger 65) Buchberger,B. An Algorithm for finding a basis for the residue class ring in a Zero-dimensional polynomial ideal. PhD Thesis, Innsbruck University Maths. Inst 1965

(Buchberger 85) Buchberger,B. Groebner bases: an Algorithmic method inpolynomial ideal theory, in Multidimensional Systems Theory: Progress, Directions and Open problems in Multidimensional Systems (ed. Bose,N.) Reidel Publishing, Dordrecht 1985

(Burghes 89) Burghes,D. Introducing decision mathematics into the school syllabus, in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Burghes 92) Blum,W., Burghes,D., Green,N. and Kaiser-Messmer,G. Teaching and Learning of mathematics and its applications: first results from a comparative empirical study in England and Germany, in Teaching Mathematics and its Applications, volume 11, No. 3. IMA 1992

(Burghes 93) Blum,W., Burghes,D., Green,N. and Kaiser-Messmer,G. British/German comparative project: some preliminary results, in Teaching Mathematics and its Applications, volume 12, No. 1. IMA 1993

(Burghes 98) Burghes,D. The Kassel project: an international longitudinal comparative project in secondary mathematics. The Centre for innovation in mathematics teaching, University of Exeter. January 1998

(Calmet 84) Calmet,J. Introducing Computer Algebra to users and to students, in SIGSAM Bulletin, 18,4. ACM November 1984

(Calmet 91) Calmet,J. and Tjandra,I. An Artificial Intelligence Environment for Computer Algebra, in Int. Conf. on Artificial Intelligence in Mathematics (eds. Johnson,J., McKee,S. and Vella,A.), University of Strathclyde, April 1991. IMA 1994

(Cannon 82) Cannon,J. A Language for Group Theory. Dept of Pure Math, U. Sydney. Dec 1982

(Capildeo 68) Capildeo,R. Vector Algebra and Mechanics. Addison-Wesley 1968

(Carslaw 50) Carslaw, H. An introduction to the theory of Fourier's series and Integrals. Dover 1950

(Chandler 97) Chandler,S. a comparison of the public expectation of achievement in mathematics in England and Wales and Bavaria - a personal view, in Teaching Mathematics and its Applications, volume 16, No. 1. IMA 1997

(Chandler 97A) Chandler,S. Modeling law: using cellular automata to study legal regulations of conflicting land use, in *Innovation in Mathematics: Proc. 2nd. International Mathematica Symposium*, Rovaniemi, Finland (Eds. V. Keränen and P. Mitic). July 1997. Computational Mechanics Publications

(Char 84) Char, B., Geddes, K., Gonnet,G., Marshman,B. and Ponzo,P. Computer algebra in the Undergraduate Mathematics Curriculum, in SIGSAM Bulletin, vol 18 #4. November 1984. ACM New York.

(Chatfield 88) Chatfield,C. Problem Solving: a statistician's guide. Chapman and Hall 1988

(Chen 77) Chen,P. The Entity-Relationship approach to Logical Database Design. QED Information Sciences, Wellesley MA. 1977

(Clements 91) Clements, R. The integrated micro computer software environment for the support of mathematical modelling teaching, in Teaching of Mathematical Modelling and Applications (proc ICTMA-4, Denmark July 1989) (Niss,M, Blum,W. and Huntley,I. Eds) 1991 Ellis Horwood.

(Clements 95A) Clements,R.R. Teaching the Numerical solution of ODEs using spreadsheets, in Maths and Stats vol. 6, #2 pp 10-14. CTI Birmingham 1995

(Clements 95B) Clements,R.R. Spreadsheets as a tool for teaching elementary Numerical Analysis, in Maths and Stats vol. 6, #1 pp 2-7. CTI Birmingham 1995

(Clements 95C) Clements,R.R. Illustrating Finite difference solutions of PDEs using, in Maths and Stats vol. 6, #3 pp 8-12. CTI Birmingham 1995

(Coad 90) Coad,P. and Yourdon,E. Object-Oriented Analysis. Prentice Hall 1990

(Cocking 88) Cocking, R.R.and Chipman, S. Conceptual issues related to Mathematics achievement of language minority children, in Linguistic and cultural influences on learning Mathematics, pp. 17-46. (Cocking, R.R. and Mestre, J., eds.) Lawrence Erlbaum Associates, Hillsdale, NJ. 1988

(Collett 91) Collett,D. Modelling Binary Data, Chapman Hall 1991

(Cox 86) Cox,B. Object-oriented programming: an evolutionary approach. Addison-Wesley, p29. 1986

(Crighton 95) Crighton, D. IMA Presidential Lecture, Royal Society of, 23 March 1995

(Cunningham 86) Cunningham,W. and Beck,K. A Diagram for Object-oriented Programs, in Proc. OOPSLA '86, in SIGPLAN Notices vol 21, #11. ACM November 1986

(Daly 95) Daly,F., Hand,D., Jones,M., Lunn,A. and McConway,K. Elements of Statistics. Addison Wesley 1995

(Davenport 86) Davenport,J and Singer,F. Elementary and Liouvillian Solitions of Linear Differential equations, in J. Symbolic Computation, vol 2 pp237-260 1986

(Davenport 88) Davenport,J., Siret,Y and Tournier,E. Computer Algebra: systems and algorithms for algebraic computation. Academic Press (2nd ed. 1993) 1988

(Davenport 91) Davenport,J. The role of Intelligence in Computer Algebra, in Int. Conf. on Artificial Intelligence in Mathematics (eds. Johnson,J., McKee,S. and Vella,A.), University of Strathclyde, April 1991. IMA 1994

(Davies 97) Davies, A. Finite elements on a spreadsheet, in Int. J. of Mathematics Education in Science and Technology, vol 28, #2 pp249-67. Taylor and Francis Ltd 1997

(Davis 85) Davis, R.B. The role of representations in problem solving: case studies, in J. Mathematical Behaviour (ed. Davis, B.) vol 4, no. 1. April 1985. Ablex Press, Greenwich CT.

(Dettori 95) Dettori,G., Garuti,R., Lemut,E. and Netchitailova,L. An analysis of the Relationship between Spreadsheet and Algebra, in Technology in Mathematics Teaching - a bridge between Teaching and Learning (Proc. Technology in Mathematics Teaching Conference, Birmingham, Sept. 93) (Burton,L. and Jaworski,,B. eds) Chartwell-Bratt 1995

(Derive 97) DERIVE version 4.03 Soft Warehouse, Inc., 3660 Waialae Avenue, Suite 304, Honolulu, Hawaii 96816-3236. 1997

(Dershowitz 85) Dershowitz, N. Computing with Rewrite Systems, in Information and Control vol 65 (pp122-157) Academic Press 1985

(Dettman 62) Dettman,J. Mathematical methods in Physics and Engineering. MacGraw-Hill 1962

(Devitt 89) Devitt,J. Unleashing Computer Algebra on the Mathematics Curriculum, in Proc ISSAC 89, Portland, Oregon (Gonnet,G., ed.) ACM July 1989

(Dewer 92) Dewer,M. Using Computer Algebra to Select Numerical Algorithms, in Proc ISSAC 92, Berkeley,CA. (Wang,P ed.) ACM 1992

(Dijkstra 68) Dijkstra, E. The Structure of the "THE" Multi-programming System. Comm ACM vol 11, 5. May 1968

(Draper 96) Draper, W., Brown, M., Henderson, F. and McAteer, E. Integrative Evaluation: an emerging role for classroom studies of CAL, in Computers and Education 26, 1-3, pp17-32 (eds Kibby,M. and Heller, R.) November 1996. Pergamon.

(Dreger 89) Dreger, B. Function Point Analysis (p4). Prentice Hall 1989

(Dubinsky 92) Dubinsky, E. A learning approach to Calculus, in Symbolic Computation in Undergraduate Mathematical Education (Karian, Z, A., ed) MAA note #24 . Mathematical Association of America 1992

(Dubisch 90) Dubisch,R. The Tool Kit: a Notebook Subclass, in Mathematica Journal vol 1 #2. Fall 1990

(Dudewicz 88) Dudewicz,E. and Mishra,S. Modern Mathematical Statistics. Wiley 1988

(Dyke 92) Dyke, P. and Whitworth, R. Guide to Mechanics. MacMillan 1992

(Easthope 64) Easthope,C. Dynamics: a Vectorial Treatment. Butterworth 1964

(Edwards 89) Edwards, D and Hamson, M. Guide to Mathematical Modelling. MacMillan 1989.

(Edwards 92) Edwards,L. A LOGO Microworld for Transformational Geometry, in Learning Mathematics and Logo (Hoyles,C and Noss,R eds.) MIT Press 1992

(Engebretson 97) Engebretson ,A. A Mathematical look at a Free Throw using Technology, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Erikksson 98) Erikksson, H-E and Penker,M. UML Toolkit. Wiley 1998

(Fateman 90) Fateman,R. Advances and trends in the Design and Construction of Algebraic Manipulation Systems, in Proc ISSAC 90, Tokyo, (ed. Watanabe,S and Nagata,M.) ACM Press 1990

(Fateman 96) Fateman,R. Why Computer Algebra Systems sometimes can't solve simple equations, in SIGSAM Bulletin 30,2, issue 116 June 1996

(Fitch 77). Fitch,J. and Norman,A. Implementing LISP in a high-level Language, in Software Practice and Experience vol 7 pp713-725. 1977

(Fitch 85) Fitch,J. Solving algebraic problems with Reduce, in J. Symbolic Computation vol 1 pp211-227. 1985

(Fitch 98) Fitch, J. on http://www.bath.ac.uk/~masjpf/home.html#comp February 1998

(Fitzharris 96) Fitzharris, A. Modelling with ARENA, in Maths and Stats vol. 7, #1, pp 40-44. CTI Centre, Birmingham. Feb 1996

(Furse 91) Furse,E. The Mathematics Understander, in Int. Conf. on Artificial Intelligence in Mathematics (eds. Johnson,J., McKee,S. and Vella,A.), University of Strathclyde, April 1991. IMA 1994

(Geddes 89) Geddes,K. On the Risch-Norman Integration Method and its implementation in Maple, in Proc ISSAC 89, Portland, Oregon (Gonnet,G., ed.) ACM July 1989

(Geiger 97) Geiger,V. Secondary School Applications and Graphing Calculators - What should we be doing now?, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Giordano 97) Giordano,F., Weir,M. and Fox,W. A First Course in Mathematical Modelling. Brookes-Cole 1997

(Glynn 90) Glynn, J. Exploring Math from algebra to calculus with Derive. MathWare 1990.

(Gonzalez 98) Gonzalez,C., Hoy,M. and Barr,S. A Georgia Initiative:Calculus, the TI-92, Distance learning and support via the Internet, in Proc. 3rd. Int. Derive and TI-92 Conference, Gettysburg, July 1998 (Leinbach,C., ed.). Mathware. July 1998

(Goos 97) Goos,M. Technology as a Tool for Transforming Mathematical Tasks, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Gray 94) Gray,J. Mastering Mathematica: Programming Methods and Aplications. Academic Press 1994

(Greenhalgh 90) Greenhalgh, D. An Epidemic Model with a Density-Dependent Death rate, in IMA Journal of Applied Mathematics in Medicine and Biology, Vol 7 (pp1-26), Oxford University Press 1990.

(Greenman 94) Greenman, J. Maps, Diagrams and Equations-the Stella approach to Mathematical Modelling, in Maths and Stats vol 5 # 3. CTI Centre, Birmingham. August 1994

(GSO 98) Schulordnung für die Gymnasien in Bayern. Ref. VI/7-S5503-8/91244. Bayeisches StaatsMinisterium für Unterricht, Kultus, Wissenschaft und Kunst, 80327 München. 19 July 1998.

(Guardian 94) From an article in The Guardian 29/11/94

(Gunn 96) Gunn, C. CAL Evaluation: What questions are being answered? A response to the article "Integrative Evaluation" by Draper et al., in Computers and Education 27, 3-4, pp157-160 (ed Heller, R.) August 1996. Pergamon.

(Guthery 89) Guthery,S. Are the Emperor's new clothes Object-Oriented?, in Dr. Dobb's Journal vol 14, #12. (pp80-86) December 1989

(Haie 93) Haie, N., Martins, J., Veloso de Vega, E. Application of the dual reciprocity boundary element method (DRM) to ground water pollution, in Boundary Element Technology VIII (pp 37-46) (eds. Pina, H. and Brebbia, C.A.) Computational Mechanics Publications 1993.

(Haines 95) Haines, C. and Izard, J. Assessment in Context for Mathematical Modelling, in Advances and Perspectives in the teaching of Mathematical Modelling and Applications (Proc ICTMA-6, August 1993, Delaware. ed. Sloyer, C., Blum, W. and Huntley, I.) pp 163-74. Water Street Mathematics, 1995

(Hair 84) Hair,J., Anderson,R., Tatham,R. and Black,W. Multivariate data analysis with readings. Prentice Hall 1984

(Hamson 97) Hamson.M and Lynch M. Experiences with System Modelling in a social and business context, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(Hamson 97A) Hamson,M. Using the SB Modelmaker Package, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Hamson 97B) Hamson,M. Developing simple random number models using Microsoft Excel, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Hanna 89) Hanna, G. Teaching the appropriate use of a mathematical model, in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Harper 91) Harper,D., Wooff,C. and Hodgkinson,D. A guide to Computer Algebra Systems. Wiley 1991

(Harper 96) Harper, D. Review of MathWise Astronomy Module, in Maths and Stats vol. 7, #3, pp 4-7. CTI Centre, Birmingham. Aug 1996

(Hayes 91) Hayes,F. and Coleman,D. Coherent Models for Object-Oriented Analysis, in Proc. OOPSLA '91, Phoenix,AZ. (ed. Paepcke,A.) SIGPLAN Notices 26, #11. ACM November 1991

(Hearn 67) Hearn, A. REDUCE User Manual. Institute of Physics, Stanford University ITP-247 1967

(Hearn 71) Hearn,A. Application of Symbol Manipulation in Theoretical Physics, in Proc. SYMSAC pp17-21 1971

(Hearn 83) Hearn, A. REDUCE-3 User Manual. The Rand Corporation. Santa Monica, CA. 1983

(Heid 88) Heid, M. K. Resequencing Skills and Concepts in applied Calculus using the computer as a tool, in Journal for Research in Mathematics Education vol. 19, #1, pp 3-25. 1988

(Henn97) Henn, H-W. Mathematics as Orientation in a Complex World, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(Henn 97A) Henn,H-W. The Impact of Technology on Modelling Activities, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Herod 98) Herod,J. Creating Models for Undergraduate Education, in CASE Newsletter, Dept. of Math. Sciences, US Military Academy West Point NY 10996 (Small, D, ed.) March 1998

(Herring 97) Herring,M. Motoring - Modelling in the fast lane, inTeaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(Heubach 98) Heubach,S. An Innovative Modelling Approach at the Freshman/Sophomore level, in Proc. 3rd. Asian Technology Conference in Mathematics (ATCM98, ed. Yang,W.), Tsukuba, Japan (http://www.runet.edu/~atcm/atcm98/) August 1998

(Heugl 94) Heugl,H. the Austrian research project: Symbolic computation systems in the classroom, in DERIVE in education: opportunities and strategies (eds Heugl, H. and Kutzler, B.) Proceeding of the 1993 DERIVE conference, Krems Austria. Chartwell-Bratt 1994

(Heugl 98) Heugl,H. Modular Thinking and Learning - a new quality or a problem for Mathematics Education, in Proc. 10th Int. T^3 Conference, Nashville. March 1998

(Hillel 92) Hillel,J., Lee,L., Laborde,C. and Linchevski, L. Basic Functions through the lens of a Computer Algebra System, in Journal of Mathematical Behaviour, vol 11, pp 119-158. 1992.

(Hirstein 91) Hirstein, J. Applications in secondary school mathematics, in Proceedings of the Korea/US seminar on comparative analysis of mathematical education in Korea and the United States. (J. H. Woo(ed.)) Seoul Korea 1991.

(Hirstein 95) Hirstein, J. Assessment of Mathematical Modelling, in Advances and Perspectives in the teaching of Mathematical Modelling and Applications (Proc ICTMA-6, August 1993, Delaware. ed. Sloyer, C., Blum, W. and Huntley, I.) pp 163-74. Water Street Mathematics, 1995

(Hodgson 95) Hodgson,T. and Amend,J. Using the Laboratory interface in the mathematics classroom - What, why and how, in Advances and Perspective in the teaching of Mathematical Modelling and Applications (Proc. ICTMA-6, Delaware, August 1993) Water Street Mathematics 1995

(Hodgson 97) Hodgson,T. On the use of Open-ended, real-world problems, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(Horstmann 97) Horstmann,C. Practical Object-Oriented Development in C++ and Java. Wiley 1997

(Hosack 84) Hosack,J, Lane,K. and Small,D. Report on the use of a Symbolic Mathematics System in Undergraduate Instruction, in SIGSAM Bulletin, 18,4. ACM November 1984

(Houston 97) Houston, K. Evaluating Ratings scales for the Assessment of Posters, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) (pp146-56) Albion Publishing, January 1997.

(Hoydalsvik 93) Hoydalsvik,G. and Sindre,G. On the purpose of Object-Oriented Analysis. Proc OOPSLA '93. 1993

(Hoyles 87) Hoyles,C. and Noss,R. Synthesising Mathematical concepts and their formulation through the construction of a Logo-based school mathematics curriculum, in Int. J. Mathematics Education in Science and Technology, vol 18 #4 (p581-95) 1987

(Huetinck 97) Huetinck,L. Modelling in Secondary level schools using Computer Visualisation, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Hunter 95) Hunter,M., Marshall,P., Managhan,J. and Roper,T. Using a CAS with 14-15 year-old Students, in Technology in Mathematics Teaching - a bridge between Teaching and Learning (Proc. Technology in Mathematics Teaching Conference, Birmingham, Sept. 93) (Burton,L. and Jaworski,,B. eds) Chartwell-Bratt 1995

(Huntley 90) Huntley, I.D. and James, D.J.G. Mathematical Modelling: a Source Book of Case Studies. Oxford 1990

(Huntley 91) Huntley, I. Computing mathematics-a new direction for applied mathematics? in Teaching of Mathematical Modelling and Applications (proc ICTMA-4, Denmark July 1989) (Niss,M, Blum,W. and Huntley,I. Eds) 1991 Ellis Horwood.

(Hurley 91) Hurley, J. A Computer Laboratory for Calculus, in The Laboratory approach to teaching Calculus (Leinbach, L., ed.) MAA notes 20, Mathematical Association of America 1991

(ICTMA3 89) in Learning and teaching Mathematics. Proc ICTMA-3, Kassel, Germany Sept 1987. Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. (eds.) Ellis Horwood 1989

(ICTMA4 91) Teaching of Mathematical Modelling and Applications. Proc. ICTMA-4, Denmark July 1989. Niss,M., Blum,W. and Huntley,I. (eds) Ellis Horwood 1991

(ICTMA5 93) Innovation in Maths Education by Modelling and Applications. Proc. ICTMA-5, 1991. De Lange,J., Keitel,C., Huntley,I.D. and Niss,M. (eds.) Ellis Horwood 1993

(ICTMA6 95) Advances and Perspective in the teaching of Mathematical Modelling and Applications Proc. ICTMA-6, Delaware, August 1993. Sloyer,C., Blum,W. and Huntley,I. (eds.) Water Street Mathematics 1995.

(ICTMA7 97) Teaching and Learning Mathematical Modelling. Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T. (eds.) Albion Publishing, January 1997.

(ICTMA8 98) Mathematical Modelling - Teaching and Assessment in a Technology-rich world. Proc ICTMA-8, Brisbane, Australia, August 1997. Galbraith,P., Blum,W., Booker,G. and Huntley, I.D. (eds.) Ellis Horwood 1998.

(Iglesias 97) Iglesias,A. and Power,H. Boundary Elements with Mathematica, in *Innovation in Mathematics: Proc. 2nd. International Mathematica Symposium*, Rovaniemi, Finland (Eds. V. Keränen and P. Mitic). July 1997. Computational Mechanics Publications

(Ikeda 97) Ikede, T. A case study of instruction and Assessment in mathematical modelling-"the delivering problem", in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(IMS 95) Mathematics with Vision: Proc. 1st. International Mathematica Symposium, Southampton (Editor with V. Keränen). July 1995. CMP

(IMS 97) Innovation in Mathematics: Proc. 2nd. International Mathematica Symposium, Rovaniemi, Finland (Editor with V. Keränen and A. Hietamäki). June 1997. CMP

(Jablonka 97) Jablonka,E. What makes a model effective and useful (or not) ? in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(Jacob 97) Jacob,C. Simulating Evolution with Mathematica, in *Innovation in Mathematics: Proc. 2nd. International Mathematica Symposium*, Rovaniemi, Finland (Eds. V. Keränen and P. Mitic). July 1997. Computational Mechanics Publications

(Jacobson 87) Jacobson,I. Object-oriented development in an Industrial Environment, in Proc. OOPSLA ,87. SIGPLAN vol 22, #12. ACM December 87

(Jacobson 92) Jacobson,I., Christerson,M., Jonsson,P. and Overgaard,G. Object-oriented Software Engineering. Addison-Wesley 1992

(Jenks 84) Jenks,R. A Primer: 11 keys to new Scratchpad, in Proc. EUROSAM 84 pp123-147. (SV Notes in Computer Science 174). Springer Verlag, 1984

(Jenks 92) Jenks,R. and Sutor,R. AXIOM: the Scientific Computation System. Springer Verlag 1992

(Jungerson 97) Jungerson,P., Nissen,K. and Vagner,S. Application of IT-92 and other Computer Algebra Systems in the Teaching of Mathematics in Denmark, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Kahrimanian 53) Kahrimanian,H. Analytical differentiation by a digital computer. M.A. Thesis, Temple Univ, Philadelphia. May 1953

(Kaiser 95) Kaiser-Messmer,G. Results from a comparative empirical study in England and Germany on the learning of mathematics in context, in Advances and Perspective in the teaching of Mathematical Modelling and Applications (Proc. ICTMA-6, Delaware, August 1993) Water Street Mathematics 1995

(Kaiser-Messmer 91) Kaiser-Messmer,G. Application-oriented mathematics teaching: a survey of the theoretical debate, in Teaching of Mathematical Modelling and Applications (proc ICTMA-4, Denmark July 1989) (Niss,M, Blum,W. and Huntley,I. Eds) 1991 Ellis Horwood.

(Kajler 90) Kajler, N. Building graphical user interfaces for Computer Algebra Systems. (Ed. A. Miola) *Proc. of DISCO 1990*, pages 235-244, Capri, Italy. Springer Verlag 1990

(Kajler 92) Kajler, N. CAS/PI: a Portable and Extensible Interface for Computer Algebra Systems. (Ed.Wang, P.) *Proc. ISSAC 1992*, pages 376-386, Berkeley, USA. ACM Press 1992

(Kajler 94) Kajler, N. and Soiffer, N. Some Human Interaction Issues in Computer Algebra. *SIGSAM Bulletin 107*, vol 28, 1. ACM Press March 1994

(Kawski 97) Kawski, M. How CAS and visualization lead to a complete rethinking of an introduction to vector calculus, in Proc ICTMT3, Koblenz (ed. Fraunholz,W.) Inst für Mediandidaktik der Universität Koblenz. 1997

(Kawski 98) Kawski,M. Stokes Theorem and control: Visualisation and applications. 11th. ICTCM Conference, New Orleans, (http://hepg.awl.com/ICTCM/) November 1998

(Keady 96) Keady,G. University of Western Australia. Private Communication. 1996

(Keiren 92) Keiren,T. Mathematics in a LOGO environment: a recursive look at a complex phenomenon, in Learning Mathematics and Logo (Hoyles,C and Noss,R eds.) MIT Press 1992

(Keitel 93) Keitel,C. Implicit Mathematical models in Social Practice and Explicit Mathematics Teaching by Applications, in Innocation in Mathematics Teaching by Modelling and Applications, Proc. ICTMA-5, Utrecht, 1991 (De Lange,J., Keitel,C., Huntley,I. and Niss,M., eds). Ellis Horwood 1993

(Keller 97) Keller, B. And Russell, C. Effects of the TI-92 on Calculus Students Solving Symbolic Problems, in International Journal of Computer Algebra in Mathematics Education 4, #1 (Berry, J. ed.) Research Information Limited 1997

(King 89) King,R. My cat is object-oriented, in Object-oriented concepts, databases and applications (Kim,N. and Lochevsky,F., eds.) pp23-30 1989

(Klinger 94) Klinger, W. Using DERIVE in the third and fourth form of grammar schools in Austria, in DERIVE in education: opportunities and strategies (eds Heugl, H. and Kutzler, B.) Proceeding of the 1993 DERIVE conference, Krems Austria. Chartwell-Bratt 1994

(Koshafian 86) Koshafian, S. and Copeland, G. Object Identity, SIGPLAN Notices vol 21, 11 (p406) Nov. 1986

(Kovacic 77) Kovacic,J. An algorithm for solving 2nd order linear homogeneous differential equations, Preprint, Brooklyn College, City University of NY. 1977

(Kovacic 86) Kovacic,J. An algorithm for solving 2nd order linear homogeneous differential equations, in J. Symbolic Computation, vol 2, 1986

(Knuth 68) Knuth,D. Fundamental Algorithms. Addison-Wesley 1968.

(Knuth 70) Knuth,D. and Bendix, P. Simple word problems in Universal Algebras, in Computational Problems in Abstract Algebra, pp263-297. Pergamon, Oxford 1970

(Kruketskii 76) Kruketskii, V.A. The Psychology of Mathematical abilities in Schoolchildren (Kilpatrick, J. And Wirszup, I., eds.) University of Chicago Press 1976

(Kruczynski 98) Kruczynski,B. The TI-83 does a Loop-to-loop. 10th T^3 Conference, Nashville, (http://ti.com/calc/docs/) March 1998

(Krzanowski 98) Krzanowski,W. Introduction to statistical Modelling, Arnold 1998

(Kutzler 97) Kutzler,B. With the TI-92 towards computer age mathematics teaching, in Int. J of Computer algebra in mathematics teaching (Berry,J. ed.) vol 4, #1. Research Information Ltd 1997

(Lafore 95) Lafore, R. Object-Oriented Programming in C++. Waite Group Press 1995

(Lambert 89) Lambert,P., Steward,A., Mankelow,K. and Robson,E. A Cogitive Psychology Approach to Model Formulation in Mathematical Modelling, in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Lamon 97) Lamon,S. Mathematical Modelling and the way the Mind Works, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(de Lange 93) De Lange,J. Innovation in Mathematics Education using Applications: Progress and Problems, in Innocation in Mathematics Teaching by Modelling and Applications, Proc. ICTMA-5, Utrecht, 1991 (De Lange,J., Keitel,C., Huntley,I. and Niss,M., eds). Ellis Horwood 1993

(Lawson 97) Lawson, D and Tabor, J. Computer-based experiments in mechanics, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion Publishing, January 1997.

(Lee 97) Lee,R. and Tepfenhart,W. UML and C++: a Practical Guide to Object-Oriented Development. Prentice Hall 1997

(Leinbach 91) Leinbach, L.C. Calculus laboratories using Derive. Wadsworth 1991

(Leong 86) Leong,B.L. Iris: Design of a User Interface Program for Symbolic Algebra, in Proc SIGSAM 86, pp1-6. ACM 1986

(Li 96) Li, Y., Borne,I. and O'Shea, T. A Scenario Design tool for helping students learn Mechanics, in Computers and Education 26, 1-3, pp17-32 (eds Kibby,M. and Heller, R.). November 1996. Pergamon.

(Lichtenberger 84) Lichtenberger,F. Self-explanatory Symbolic Computation for Math Education, in SIGSAM Bulletin, 18,4. 1984

(Liouville 35) Liouville,J. Memoire sur l'integration d'une classe de fonctions trancendantes, in Crelle's J, vol 13 pp93-118. 1835

(Loe 85) Loe, K.F., Ohsawa, N and Goto, E. Circuit Simulation Code Generation by Computer Algebra, in 2nd Int. Symp. on Symbolic and Algebraic Computation by Computer (Aug 1984) eds. Inada, N and Soma, T. World Scientific Publishing 1985.

(Longhawk 98) Longhawk,K. Teaching Trigonometry with the TI-92. 10th T^3 Conference, Nashville, (http://ti.com/calc/docs/) March 1998

(Loomis 87) Loomis,M., Shah,A. and Rumbaugh,J. An object-modelling technique for conceptual design, in Proc. European Conference on Object-Oriented programming (ECOOPS 87), Paris, June '87. Lecture Notes in Computer Science, vol 276, Springer Verlag 1987

(Mackie 92) Mackie,D. An evaluation of Computer-Assisted learning in Mathematics, in Int. J. Maths Edu. In Science and Technology, 23, #5 pp331-7

(Mackie 97) Mackie,D. Strategies for embedding computer-based learning into teaching, in Proc ICTMT3, Koblenz (ed. Fraunholz,W.) Inst für Mediandidaktik der Universität Koblenz. 1997

(Maeder 87) Maeder,R. Solving Boundary-value problems with Perturbations, in A Collection of Projects for the Mathematical Laboratory. SIGSAM Bulletin, 21, 3 August 1987

(Maeder 92) Maeder, R.E. Polymorphism and Message Passing, Mathematica Journal 2,4, Miller Freeman. 1992

(Maeder 93) Maeder, R.E. Oriented Programming, Mathematica Journal 3,1, Miller Freeman 1993.

(Maeder 94) Maeder, R. Mathematica's Programming Language, in Tutorial Notes, Developers Conference, Champaign IL and Oxford, UK 1994

(Maeder 94A) Maeder, R.E. The Mathematica Programmer, Academic Press, 1994.

(Maple 89) Maple: A sample interactive session. Symbolic Computation Group, Waterloo Maple Software, University of Waterloo, Canada. December 1989

(Marti 83) The Bath Concurrent LISP Machine, in Proc EuroCal, Lecture Notes in Computer Science #162, Springer Verlag 1983

(MathLink 91) MathLink Reference Guide: WRI Technical Report. Wolfram Research 1991

(Mathskills 98) http://www.hull.ac.uk/mathskills/

(Mayes 94) Mayes, R. The application of a computer algebra system as a tool in college algebra, in DERIVE in education: opportunities and strategies (Proceedings of the 1993 Krems DERIVE conference, eds. Heugl, H.and Kutzler, B.) Chartwell-Bratt 1994

(Mayes 95) Mayes, R. An Application of a Computer Algebra system as a tool in College algebra, in School Science and Mathematics, vol 95, #2. Feb 1995

(Mayes 97) Mayes, R. Current state of research into CAS in Mathematics Education, in The State of Computer Algebra in Mathematics Education (ed. Berry, J., Monaghan, J., Kronfellner, M., Kutzler, B.) Chartwell-Bratt. 1997

(Mayes 98) Mayes, R. ACT in Algebra: Student Attitude and Belief, in Proc. 3rd. Int. Derive and TI-92 Conference, Gettysburg, July 1998 (Leinbach,C., ed.). Mathware. July 1998. (also to appear in Int. J. Computer Algebra in Mathematics Education)

(Mayes 98A) Mayes, R. and Lesser,L. ACT in Algebra. McGraw-Hill 1998

(McCrae 97) McCrae,B. Modelling using Dynamic Geometry Software in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(McLean 94) McLean, S. and Scotney, B. Computer-Based Tutorials for introductory Statistics , in Innovation in Mathematics Teaching (Houston, K. ed.). Staff and Educational Development Association (SEDA) #87. December 1994

(Melin 92) Melin-Conejeros,J. The effects of using a computer algebra system in a mathematics laboratory on the achievement and attitude of calculus students. Dissertation Abstracts vol 53, #7 (Order # DA9235882) p2283. 1993

(Mendenhall 96) Mendenhall,W., Schaeffer,R. and Wackerly,D. Mathematical Statistics with Applications. PWS Boston 1986

(Meyer 96) Meyer , J. and Parsons, P. An exploration of student learning in Mathematics, in International Journal of Mathematics Education in Science and Technology 27, #5 ,pp 741-751 (ed. Walker, D.) Taylor and Francis Limited 1996

(Milne 48) Milne, E. Vectorial Mechanics. Methuen 1948

(Mitic 91) Mitic, P. Unpublished results on limits of rational functions involving exponentials, reported to Soft Warehouse and Wolfram Research, 1991

(Mitic 92) Mitic,P. and Thomas,P. Pitfalls and limitations of computer algebra, in Computers and Education vol. 22, 4. Elsevier Science 1994

(Mitic 94) Mitic,P. and Thomas,P. A Mathematical Model of a Firebreak using Derive, in DERIVE in education: opportunities and strategies (Proceedings of the 1993 Krems DERIVE conference, eds. Heugl, H.and Kutzler, B.) Chartwell-Bratt 1994

(Mitic 94A) Mitic,P and Thomas,P. Modelling the Effects of a Firebreak with DERIVE. In *International DERIVE Journal* 1, 2.(Berry,J., ed.) Ellis Horwood 1994

(Mitic 94B) Mitic,P and Thomas,P. Pitfalls and Limitations of Computer Algebra. In *Computers and Education* 22, 4. Elsevier Science 1994

(Mitic 95A) An Object-Oriented Environment for Newtonian Particle Mechanics. In *Proc. 1st. International Mathematica Symposium*, Southampton (P. Mitic, V. Keränen (editors)). Computational Mechanics Publications July 1995.

(Mitic 95B) An Event-Driven Interface for the *AMK* Object-Oriented Newtonian Particle Mechanics system. In *Proc. 2nd. Conference on Software Engineering in Higher Education (SEHE)*, Alicante, Spain (L. Sucharov, P.Mitic and J-L. Uzo (editors)). Computational Mechanics Publications November 1995

(Mitic 97) Mitic, P. Symbolic-Numeric Option Valuation, in *Innovation in Mathematics: Proc. 2nd. International Mathematica Symposium*, Rovaniemi, Finland (Eds. V. Keränen and P. Mitic). July 1997. Computational Mechanics Publications

(Mitic 98) Mitic,P. A Computer Proof of the Central Limit Theorem, in Proc. 3rd. Int. Derive and TI-92 Conference, Gettysburg, July 1998 (Leinbach,C., ed.). Mathware. July 1998

(Moses 67) Moses,. Symbolic Integration. PhD Thesis MAC TR-47, MIT 1967

(Moses 71) Moses,J. Symbolic Integration: the Stormy Decade, in Comm. ACM, 14, 2. August 1971

(MST204 89) Mathematical Models and Methods, 2nd Level course. Open University Press, 1989

(MST204 91) Mathematical Methods and Modelling, Tutor-marked Assignment 3, Q2 Open University 1991

(MST204 92) Mathematical Methods and Modelling, modelling project in Tutor-marked Assignments 4 and 8. Open University 1992

(MST204Project) MST204 Mathematical Models and Methods Project Guide (pp 25 and 29) Open University Press 1989

(Neill 94) Neill, N. and Curran, D. Computer-Based Testing and Assessment, in Innovation in Mathematics Teaching (Houston, K. ed.). Staff and Educational Development Association (SEDA) #87. December 1994

(Neill 97) Neill,N. Maple - Methodology, Modelling and Matrices, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Nocker 98) Nocker,R. The Austrian TI-92 Project: a preliminary report, in Proc. 3rd. Int. Derive and TI-92 Conference, Gettysburg, July 1998 (Leinbach,C., ed.). Mathware. July 1998

(Nolan 53) Nolan, J. Analytical differentiation on a digital computer. M.A. Thesis MIT. May 1953

(Norman 74) Norman,A. and Moore,P. Implementing the new Risch Integration Algorithm, in Proc. 4th Internation Colloquium on Advanced Computing methods in Theoretical Physics, St Maximin, pp99-110. 1974

(Norman 94) Norman, F. and Pritchard, M. Cognitive obstacles to the learning of Calculus: a Kruketskiian perspective, in Research Issues in Undergraduate Mathematics Learning (Kaput, J.J. and Dubinsky, E., eds.) MAA Notes #33. MAA 1994

(Noss 97) Noss, R., Healy, L. and Hoyles, C. The construction of mathematical meaning: connecting the visual with the symbolic (pp 203 - 33) in Educational Studies in Mathematics vol. 33, #2. (ed. Ruthven,K.) Kluwer. July 1997

(Ohtsuka 98) Ohtsuka,H. Higher order programming using Mathematica in a Teaching Programme, in Proc. 3rd. Asian Technology Conference in Mathematics (ATCM98, ed. Yang,W.), Tsukuba, Japan (http://www.runet.edu/~atcm/atcm98/) August 1998

(Orman 95) Orman,B. An attempt to integrate traditional applied mathematics and modern mathematical modelling activities, in Advances and Perspective in the teaching of Mathematical Modelling and Applications (Proc. ICTMA-6, Delaware, August 1993) Water Street Mathematics 1995

(Padget 85) Padget,J. Current Developments in LISP, in Proc. SYMSAC 85, 1985

(Page 89) Page,T., Berson,S., Cheng,W. and Muntz,R. An Object-Oriented Modelling Environment, in Proc. OOPSLA '89, New Orleans (SIGPLAN Notices vol 24, #10) ACM Press. October 1989

(Palmiter 91) Palmiter, J. R. Effects of Computer Algebra Systems on Concept and Skill Acquisition in Calculus, in Journal for Research in Mathematics Education, 22, #2, pp. 151-156. 1991

(Pardoe 97) Pardoe,J. and King,M. Object-Oriented Programming using C++. Macmillan 1997

(Parnas 79) Parnas, D. On the Criteria to be used in Decomposing Systems into Modules, in Classics in Software Engineering. Yourdon Press 1979

(Parsons 94) Parsons,D. Object-oriented Programming with C++. Letts 1994

(Pavelle 85) Pavelle,R. and Wang,S. MACSYMA from F to G, in Journal. of Symbolic Computation, vol 1 #1 North Holland, 1985

(Pavelle 85A) Pavelle,R. MACSYMA: capabilities and applications to problems in engineering and the sciences. Symbolics Inc. MACSYMA Group, Cambridge MA 17 March 1985

(Pavelle 85B) Ravelle,R. (ed.) Applications of Computer Algebra. Kluwer 1985

(Penrose78) Penrose, O. J. Math Mod. For Teachers, Vol 1, (pp 31-42), 1978

(Pemberton 97) Pemberton,M. The Implications of using Symbolic Manipulators in Teaching Undergraduate Mathematics, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Piaget 78) Piaget, J. The equilibration of cognitive structures. Harvard University Press, Cambridge MA. 1978

(Polya 45) Polya,G. How to Solve it. Princeton University Press. 1945

(Potari 93) Potari,D. Mathematisation in a real life investigation, in Innocation in Mathematics Teaching by Modelling and Applications, Proc. ICTMA-5, Utrecht, 1991 (De Lange,J., Keitel,C., Huntley,I. and Niss,M., eds). Ellis Horwood 1993

(Pröpper 98) Pröpper,W. Introducing the concept of Integration with the TI-92, in Proc. 3rd. Int. Derive and TI-92 Conference, Gettysburg, July 1998 (Leinbach,C., ed.). Mathware. July 1998

(Quadling 57) Quadling,D. and Ramsay,A. Elementary Mechanics. Bell 1957

(Ross 87) Ross, R. Entity Modeling: Techniques and Application. Boston MA: Database Research Group. 1987

(Quatrani 98) Quatrani,T. Visual Modelling with Rational Rose and UML. Addison Wesley 1998

(Quigley 97) Quigley,M. Integrating Computer Algebra into mainstream Mathematics Teaching, in Proc. 2nd. Asian Technology Conference in Mathematics (ATCM97, ed. Yang,W.), Penang, Malaysia (http://www.runet.edu/~atcm/atcm97/) June 1997

(Rambaugh 91) Rambaugh,J., Blaha, M., Premerlani,W., Ey,F. and Lorensen, W. Object-Oriented Modelling and Design. (p459) Prentice Hall 1991

(Ramsden 95) Kent,P, Ramsden,P and Wood,J. Mathematica for valuable and viable computer-based learning, in Mathematics with Vision: Proc. 1st. Int Mathematica Symposium, Southampton (Eds V. Keränen and P. Mitic). July 1995. CMP

(Rational 98) Rational Rose 4.0 CASE tool, Rational Software Corporation, 18880 Homestead Rd. Cupertino CA 95014 (http://www.rational.com) 1998

(Rayna 68) Rayna, G. Reduce - Software for algebraic computing. Springer Verlag 1968

(Repo 94) Repo, S. Understanding and Reflective Abstraction: Learning the Concept of the Derivative in the Computer Environment, in International DERIVE Journal (ed. Berry, J.) 1, #1. Ellis Horwood. April 1994

(Rickhuss 93) Rickhuss, M. Computer Algebra Systems and Secondary Education. M. Phil. Thesis, Open University June 1993

(Riddle 91) Riddle,A. Tricks of the Trade: Dealing with Equations. Mathematica Journal 1,3. Miller Freeman, Winter 1991

(Risch 69) Risch,R.H. The Problem of Integration in finite terms, Trans A.M.S 139 pp 167-189, MR 38 #5759, 1969

(Roach 92) Roach,K. Symbolic Integration, in Advanced Tutorial Notes, 1992 Mathematica Conferences, Boston and Rotterdam. Wolfram Research 1992.

(Rodin 89) Rodin, E.Y. and Egan, K. Mathematical Modelling of the rate of Chemical Reactions in Mathematical Computing Modelling Vol 12, No. 12 (pp 1707-1713). Pergamon Press 1989

(Rogers 93) Rogers,G. and Thomlinson,M. Spreadsheet Applications in Modelling problems based on partial differential equations, in Innovation in Mathematics Teaching by Modelling and Applications, Proc. ICTMA-5, Utrecht, 1991 (De Lange,J., Keitel,C., Huntley,I. and Niss,M., eds). Ellis Horwood 1993

(Rosamond 94) Rosamond, F.A. The role of Emotion: Expert and Novice Mathematical Problem-solving, in Research Issues in Undergraduate Mathematics Learning (Kaput, J.J. and Dubinsky, E., eds.) MAA Notes #33. Mathematical Association of America 1994

(Rubin 92) Rubin,K. and Goldberg,A. Object Behaviour Analysis, in Comm. ACM vol 35, #9 September 1992

(Sacristan 97) Sacristan,I. Windows on the Infinite: Constructing meanings in a Logo-based Micro-world: Ph.D. Thesis, University of London 1997

(Saeki 89) Saeki M, Horai Hand Enomoto H, Software Development Process from Natural Language Specification, *in Proc 11th International Conference on Software Engineering, New York.* IEEE Press 1989

(Sakakibara 97) Sakakibara,S. Design of a Wavelet Package, in *Innovation in Mathematics: Proc. 2nd. International Mathematica Symposium*, Rovaniemi, Finland (Eds. V. Keränen and P. Mitic). July 1997. Computational Mechanics Publications

(Salzano 83) Salzano,L. The Teaching of Skills in Mathematical Modelling. M.Phil Thesis, U. Nottingham. May 1983

(Sandefur 90) Sandefur, J.T. Discrete Dynamical Systems. Oxford University Press 1990.

(SCAA 93) CGE Advanced and Advanced Supplementary Examinations - Subject Core for Mathematics. Schools Curriculum and Assessment Authority, London 1993

(Schoenfeld 88) Schoenfeld,A.H. Uses of computers in mathematical instruction, in Computers and Mathematics: the use of computers in undergraduate instruction. Committee on Computers in Mathematical Education, MAA 1988

(Schupp 89) Schupp, H. Applied Maths instruction in the Lower secondary level - between traditional and new approaches, in Applications and Modelling in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Seidewitz 86) Seidewitz, E. and Stark, M. Towards a General Object-Oriented Software Development Methodology, in Proc. 1st Int Conf. on Ada Programming Language Applications for the NASA Space Station. (pD.4.6.4) NASA: Lyndon B Johnson Space Center 1986

(Selwyn 97) Selwyn, N. Students' attitudes towards Computers: Validation of a computer attitude scale for 16-19 education, in Computers and Education 28, #1 (ed. Heller, R.) pp35-41 1997. Pergamon.

(Seppanen 91) Seppanen,V., Heikkinen,M. and Lintulampi,R. SPADE - Towards CASE tools that can Guide Design, in Advanced Information Systems Engineering (Proc CAiSE 91), Lecture Notes in CS 498 (eds. Anderson,R., Bebenko,J. and Solvberg,A.) Springer 1991

(Sharke 98) Sharke,P. and Koenig,E. New perspectives in Teaching Mathematics due to the use of the TI-92, in Proc. 3rd. Int. Derive and TI-92 Conference, Gettysburg, July 1998 (Leinbach,C., ed.). Mathware. July 1998

(Shay 97) Shay,K. An excellent companion for differential equations reform, in Int. J of Computer algebra in mathematics teaching (Berry,J. ed.) vol 4, #1. Research Information Ltd 1997

(Shlaer 88) Shlaer, S. and Mellor, S. Object-Oriented Systems Analysis: Modeling the World in Data. Yourdon Press, Englewood Cliffs NJ. 1988

(Shlaer 89) Shlaer,S. and Mellor,S. An Object-oriented approach to Domain Analysis, in SIGSOFT Software Engineering notes, vol 14, #5. ACM July 1989

(Shlaer 92) Shlaer, S. and Mellor, S. Object Lifecycles: Modeling the World in States. Prentice Hall. 1992

(Shutler 97) Shutler,P. and Springham,S. How to stay dry in Singapore, in Teaching Mathematics and its Applications, volume 16, No. 1. IMA 1997

(Siew 97) Siew,P. On the differentiation between the magnetic anomalies of cylinders and disks, in IMA Jnl of Applied Maths (ed. Ogden,R.) vol59, #2 OUP October 1997.

(Singer 81) Singer,M. Liouvillian solutions of nth order linear differential equations, in Am. J. Math, vol 103 1981

(Singer 92) Singer,M.and Ulmer,F. Liouvillian solutions of 3rd order linear differential equations: New bounds and necessary conditions, in Proc ISSAC 92, Berkeley,CA. (Wang,P ed.) ACM 1992

(Slagle 61) Slagle,J. A Heuristic Program that solves Symbolic Integration problems in Freshman Calculus. Ph.D. Dissertation, Harvard U. Cambridge MA May 1961

(Sloyer 89) Sloyer,C. Contemporary ideas in Applied Mathematics, in Applications and Modelling in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Smith 86) Smith,C.J. and Soiffer,N. MathScribe: A User Interface for Computer Algebra Systems, in Proc SYMSAC 86, pp7-11. ACM July 1986

(Smith 88) Smith,M. and Tockey, S. An integrated approach to software requirements definition using Objects. Boeing Commercial Airplane Support (p132) Boeing Corporation, Seattle, 1988

(Smith 94) Smith,B. Studying different methods of Technology Integration for teaching problem solving with systems of equations and inequalities and linear programming, in the Journal of Computers in Mathematics and Science Teaching 13(4) pp 465-479 1994

(Spode 82) The Spode Group. Solving real problems with mathematics (vols. 1 and 2) Cranfield Press 1984

(Stella 94) Stella Simulation Manual, High Performance Systems, Hanover NH 1994

(Stephenson 61) Stephenson,G. Mathematical Methods for Science Students. Longman 1961

(Stiff 92) Stiff,K., McCollum,M and Johnson,J. Using symbolic calculators in a constructivist approach to teaching mathematics of finance, in the Journal of Computers in Mathematics and Science Teaching vol. 11 pp 75-89 1992

(Stoutemyer 77) Stoutemyer,D. sin(x)**2 + cos(x)**2 = 1, in Proc. 1977 Macsyma users conference (NASA pub. CP-2012, National Technical Information Service, Springfield VA), pp425-433. 1977

(Stoutemyer 84) Stoutemyer,D. A Radical Proposal for Computer Algebra in Education, in SIGSAM Bulletin, 18,4. ACM November 1984

(Stoutemyer 91) Stoutemyer,D. Supplementary notes for an advanced DERIVE mini-course. Soft Warehouse, Hawaii 1991

(Stoutemyer 91A) Stoutemyer, D. Private communication 1991

(Stoutemyer 92A) Stoutemyer,D. Addendum to Supplementary notes for an advanced DERIVE mini-course. Derive User Group lecture at Nottingham University April 1992.

(Tall 92) Tall, D. Mathematical Processes and Symbols in the Mind, in Symbolic Computation in Undergraduate Mathematical Education (Karian, Z, A., ed) MAA note #24 . Mathematical Association of America 1992

(Tall 93) Tall, D. and Razali M. R., Diagnosing students' difficulties in learning Mathematics, in Int. J. of Mathematical Education in Science and Technology, vol 24 #2, pp209-222 Taylor & Francis 1993

(Tan 98) Tan,K.S. and Steeb, W-H. SymbolicC++: An Introduction to Computer Algebra using Object-Oriented Programming. Springer 1998

(Taylor86) Taylor, A.B. Mathematical Models in Applied Mechanics. Oxford University Press 1986.

(Thomas 88) Thomas,P and Leadbetter,P. The Exploration of certain Operator Equations using Reduce. Technical Report 88/14. Open University 13/12/1988

(Thomlinson 89) Thomlinson , M and Norcliffe,A. Modelling case studies at the Maths-IT Interface: Mathematical Modelling of all oil Reservoirs, in Applications and Modelling in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Townend 95) Townend,S. and Pountney,D. Learning Modelling with Derive. Chartwell-Yorke 1995

(Treilibs 79) Treilibs,V. M.Phil Thesis: Formulation Processes in Mathematical Modelling. University of Nottingham. 1979

(Treilibs 80) Treilibs,V., Burkhardt,H and Lowe,B. Formulation Processes in Mathematical Modelling. Shell Centre, Nottingham. 1980

(Tsai 98) Tsai,S. and Sheu,T. Some Physcal insights into a 2-row finned-tube heat transfer, in Cmputers and Fluids (ed. Rubin,S.) vol 27, #1 pp29-46. Elsevier Science January 1998

(Tully 96) Tully, C. J. Informal Education by Computer-Ways to computer knowledge, in Computers and Education 27, 1 (eds. Kibby, M. and Heller, R.) Pergamon. Aug 1996

(Turner 73) Turner,L. and Knighton,D. Advanced Mathematics vols 1, 2. Longman 1973

(UCLES 94) University of Cambridge Local Examinations Syndicate, A and AS Modular Mathematics Syllabus Aims and Objectives (p4). UCLES Press, 1994

(UCLES 95) University of Cambridge Local Examinations Syndicate, Paper 4681 (Mechanics 1), question 8. UCLES Press, March 1995

(Usher 97) Usher, J.R. and Henderson, D. Modelling Cancer Chemotherapy, in Teaching and Learning Mathematical Modelling:Proc ICTMA-7, Jordanstown, N. Ireland, July 1995 (eds. Houston, S.K., Blum,W., Huntley, I.D. and Neill, N.T.) Albion , January 1997.

(Viklund 92) Viklund,L. and Fritzon,P. An Object-oriented Language for Symbolic Computation - Applied to Machine Element Analysis, in Proc ISSAC 92, Berkeley,CA. (Wang,P ed.) ACM 1992

(Virdefors 97) Virdefors,B. Some Excel Models in Astronomy in Senior High School, in Abstracts of Proc ICTMA-8, Brisbane, Australia, (Galbraith,P. ed.). August 1997

(Waits 95) Waits,B. and Demana,F. TI-92, The Hand-held Revolution in Computer-enhanced Maths Teaching and Learning, in Maths and Stats 6,2 (CTI) May 1995

(Wang 71) Wang, P. Evaluation of Definite Integrals by Symbolic Integration. Ph.D. Thesis and Project MAC TR-92, MIT 1971

(Warzel 89) Warzel, A. General Theory of modelling and theory of Action - A solution for the educational situation at school? in Applications and Modelling in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Watkins 93) Watkins, A.J. DERIVE-based investigations. Chartwell-Bratt 1993

(Watkins 94) Watkins,A. and Gadd,K. DERIVE-centered research at the University of Plymouth, in DERIVE in education: opportunities and strategies (eds Heugl, H. and Kutzler, B.) Proceeding of the 1993 DERIVE conference, Krems Austria. Chartwell-Bratt 1994

(Watkins 96) Watkins, A. M. Phil. Thesis. University of Plymouth. 1996

(Wenger 88) Wenger, R. H. Computers in "Transitional" Mathematics courses: Pragmatic experience and future Perspectives, in Computers and Mathematics: the use of computers in undergraduate instruction (Smith, D. A., Porter, G., Leinbach, L.C., and Wenger, R. eds.) MAA Notes # 9. MAA 1988

(Wetherill 86) Wetherill, G. Regression Analysis with Applications, Chapman Hall 1986

(Wilensky 97) Wilensky, U. Modelling Rugby: kick first, generalise later? in International Journal of Computers for Mathematical Learning, vol 1, #1. (also http:// www.tufts.edu:80/~uwilensk/papers/rugby/rugby.htm) 1997

(Wilenski 97A) Wilensky,U. Connected Mathematics - Building Concrete Relationships with Mathematical Knowledge: Ph.D. Thesis MIT 1997

(Wilson 89) Wilson,M. and James,D. Incorporating Financial Features into a Mathematical model, in Applications and Modelling in Learning and teaching Mathematics (Proc ICTMA-3, Kassel, Germany Sept 1987) (Blum,W., Berry,J., Biehler,R., Huntley,D. Kaiser-Messmer,D. and Profke,L. eds.) Ellis Horwood 1989

(Wirfs-Brock 89) Wirfs-Brock ,R. Object-oriented Design: a Responsibility-Driven approach, in Proc. OOPSLA '89, New Orleans (SIGPLAN Notices vol 24, #10) ACM Press. October 1989

(Wirfs-Brock 90) Wirfs-Brock,R., Wilkerson,B. and Wiener,L. Designing object-oriented software. Prentice Hall 1990

(Wolfram 96) Wolfram, S. The Mathematica Book (3rd edition) CUP 1996

(Young 87) Young, D.A. and Wang, P.S.GI/S: A Graphical User Interfaces for Symbolic Computation Systems. *J. Symbolic Computation*, vol 4, pages 365-380, 1987

(Yourdon 79) Yourdon,E. and Constantine,L. Structured Design. Yourdon Press, Englewood Cliffs, NY. 1979

(Yourdon 89) Yourdon,E. Modern Structured Analysis. Yourdon Press, Englewood Cliffs, NY. 1989

(Zimmerman 95) Zimmerman,L. and Olnes,F. Mathematica for Physics. Addison Wesley 1995