An Intelligent Modelling Interface for Process Simulators in Process Industries

By

Graham Adrian Clark

A Doctoral Thesis

Submitted in partial fulfilment of the requirements
for the award of

Doctor of Philosophy of Loughborough University

3rd June 2001

Abstract

Over the past three decades, modelling packages for chemical processes have become more advanced and widely used. For example, equation-oriented dynamic simulators, such as gPROMS are useful for simulating plantwide processes as well as unit operations, and are widely used by process engineers. Whereas, other types of simulator (e.g. Simulink) are often used by control engineers to solve complex control problems. However, both these types of simulator rely on the user being proficient in modelling and familiar with their syntax beforehand. A useful development would be the integration of some knowledge into the formation of the process models and automatic syntax code generation. This would lead to the design engineers having a library of knowledge to check on first, much as an expert engineer uses their past experiences to help guide them through a design. If this could be incorporated into a modelling interface this would greatly help the design engineer, especially when tackling problems in areas that they have little, or no experience.

The thesis addresses this problem and describes the design of an intelligent modelling interface that incorporates a knowledge base using some form of *a priori* case library and recall facility. The interface also incorporates an automatic input file generation stage. At present, the user can: specify a single unit operation problem to search for, retrieve similar cases from the database, specify their solution in the database based on past cases and experience, and then automatically generate an input file for either gPROMS or Simulink. These features are demonstrated through four case studies.

Keywords

Process modelling, Case based reasoning, User interface, Knowledge base

To Heather

Acknowledgements

I would like to thank:

Prof. Paul Chung and Dr. Diane Rossiter, my supervisors, for their encouragement and knowledge, for aiding me through the course of my PhD and occasionally pointing me back in the right direction.

Richard Illiffe, for help with the search query implementation and CLIPS language.

All my friends and colleagues, both in the Department and outside, who have put up with me over the last three and a bit years.

To the Engineering and Physical Sciences Research Council for funding my project.

And finally to my family and to Heather, for their love and support, and for putting up with me through thick and thin.

Contents

Figures

Tables

1. Preface

1.1 Motivation

Over the past three decades, much of the approach taken towards modelling and design of chemical processes has been based upon the designers' knowledge and experience. As the modelling packages have become more advanced and more automated, many of the original simplifications have been removed from this procedure and fewer mistakes (due to simplifications) are made. The next step, therefore, would seem to be the integration of some knowledge into this semi-automated process. This would lead to the design engineers having a library of knowledge to check on first, much as an experienced engineer uses their experiences to help guide them through a design. If this could be incorporated into a modelling interface this would greatly help the design engineer, especially when tackling problems in areas that they have little, or no experience.

With increasing environmental and safety regulations, more competitive markets and tighter specifications on product quality, there is a necessity to optimise the performance of chemical processes in minimal time at minimal cost (Marquardt, 1994). Often when investigating the performance of an existing or new process it would be useful to have a dynamic model of the process for computer simulation studies. The type of dynamic model may vary depending on the type of computer simulation study to be performed. For example, a process engineer may want to perform full-scale non-linear dynamic simulations to verify that the product quality is achievable. This requires the assistance of a control engineer to provide the controllers for the process. The control engineer often only needs to use a crude linear model for the controller design and could use a package such as Simulink (MathWorks, 1999) for the modelling and controller design. On the other hand the process engineer wants to represent the complex non-linearities of the process using a detailed model to avoid constructing a pilot plant. Hence a package such as gPROMS (PSE, 1999 a & b) could be used to provide the functionality. At present if this were the case, both engineers would become involved in having to write separate simulation code for each program. This is time consuming and expensive. What is required is that the engineers firstly work on writing a common mathematical

description of the process, or model, then input that to a user interface and the interface generates the code for both simulation packages.

1.2 Overview of Project

The aim of this project is to design an intelligent modelling interface for process simulators that incorporates a knowledge base using some form of prior case library and recall facility. Included in this is an automatic modelling package input file generation stage.

Modelling is a complex task, whether using manual problem solving techniques or an automatic modelling or simulation package. When using modelling packages, the user must, in most cases, still write the conditions and equations in a form that the package can use and then manipulate in such a way so that it can solve the problem. In our search for a new approach to the problem, the thesis offers an overview of many modelling approaches in use. The advantages and disadvantages of these approaches are assessed and compared to try and find an optimum solution – the Intelligent Modelling Interface for Process Simulators (IMIPS). This interface includes the flexible, equation-oriented statement of problems, with a knowledge (or case) base the user may search for related models (or, as referred to in the thesis, cases) and help.

Knowledge bases have been used very successfully in many fields, from architecture and law, to arbitration and recipe/menu planning. Adding a knowledge base and search mechanism to the interface adds another source of knowledge so that less experienced modellers can draw on the past experience of others. Also, looking at prior cases may serve as pointers to aid in the solution of some more complicated problems. The knowledge (or case) base may be searched using Case Based Reasoning techniques. This involves the systematic problem definition, search, retrieval, and adaptation of cases from the case base. These techniques have been used in other fields to great success, e.g. CHEF (Hammond, 1986), and CYRUS (Kolodner, 1983a, 1983b). More details are given in Chapter 3.

1.3 The IMIPS related features

The interface also incorporates the automatic generation of code using translators. The code can then be solved by a simulation package. Translators have been developed for gPROMS (PSE, 2000 a & b) and Simulink (MathWorks, 1999).

Automatic code generation has been seen as a benefit in other fields (Maclay, 2000). By removing what used to be the most time consuming task from model creation the user can concentrate on the task at hand and not so greatly on its representation so that the solver can solve it. Maclay (2000) says, 'At the prototyping stage, automatic code generation can greatly accelerate the development process, ..., because there is no significant time penalty for trying alternative solutions, automatic code generation tools positively encourage innovation.' IMIPS automatically creates input code for the model simulators by translation of the selected case. The user then runs the produced file in the simulator and can then compare results of many simulator runs. More details are given in Chapters 4 and 5.

1.4 Original Contributions

The main contributions of the thesis are as follows:

- Developing a database and indexing system for cases. (Chapter 4)
- Using case based reasoning search and retrieval mechanisms for chemical engineering single unit problems. (Chapter 4)
- Combining the above contributions to create an intelligent modelling interface, IMIPS, that includes automatic code generation from case specification for gPROMS and Simulink. (Chapters 4 and 5)
- Carrying out case studies to demonstrate IMIPS functionality. (Chapter 6)

1.5 Structure of the Thesis

The thesis is split into 7 chapters. In Chapter 2, process modelling related developments are reviewed. This includes discussions on the mathematical modelling of physical-chemical systems, the different approaches used in simulation packages, and the use of interfaces and front ends both in process engineering and other fields.

Chapter 3 outlines the principles of the case-based reasoning methodology, and how this can be related to and incorporated in the intelligent interface. Again an overview of available techniques is given, including some examples of case-based reasoning use and of the wide variety of tasks to which this methodology can be applied.

Chapters 4 and 5 show an overview of our system, IMIPS, and discusses the design and implementation issues encountered. Case studies to illustrate the uses of IMIPS are explored in Chapter 6. Chapter 7 contains the conclusions from the thesis and hence the work to follow. Finally, the appendices include the syntax used in the system, the indexing schemes, flow diagrams for the two translators, the code for IMIPS and the simulator input files automatically created by the translators for the case studies.

2. Review of Process Modelling Related Developments

2.1 Introduction

This chapter describes the key stages in the development of process modelling related tools. This sets the scene for the thesis and the work on the **IMIPS**, (**I**ntelligent **M**odelling **I**nterface for **P**rocess **S**imulators) (Clark, 1998). The layout of this chapter is as follows: In § 2.2 the ideas and methods behind mathematical modelling are explained, § 2.3 explores the types of simulation package, § 2.4 gives an overview of some of the interfaces and front ends currently available in the process industries, and § 2.5 outlines the basic features of IMIPS.

2.2 Mathematical Modelling of Physical-Chemical Systems

A mathematical model consists of a variety of equations that relate to the physical variables of the system being modelled. To ensure the model is correct the modelling of physical-chemical processes requires all the basic principles of chemical engineering science: basis, assumptions, consistency, solution of the model equations, and verification (Luyben, 1990).

- Basis – the basis for each model are the fundamental physical and chemical laws that govern the behaviour of the system. (See § 2.2.1.)

- Assumptions – the engineer must use their judgement as to what assumptions can be made to create a valid model. It is also vital that the engineer also states the assumptions used so that a true representation of the model is known.

- Consistency – the model must be mathematically consistent. There must be zero degrees of freedom, i.e. the number of unknown variables must equal the number of equations. Also ensuring the units of all equations are consistent is another important part of this procedure.

- Solution of the model equations – the solution techniques available are explored in more depth in § 2.2.2.

- Verification – once the model has been solved it is also important to check that the results gained are realistic to the real world case.

The modelling of all physical-chemical systems can be undertaken if this procedure is used. The determining factor for success is that the modeller has enough knowledge to create a good model and that there are solution methods available to then solve that model. Stephanopoulos & Han, (1996) reviewed intelligent systems in process control and included a section on modelling languages, simulation and reasoning. In this section they stated that '*we should use models that capture all available knowledge*, whether it is expressed in the form of logical propositions, order-of-magnitude, or quantitative relationships'. With this in mind, it is necessary to state that it is acceptable to simplify the model to some extent, to reduce its complexity, without eliminating the important and vital details. Von Neumann and Morgenstern (1964) state that '... the definition must be precise and exhaustive in order to make mathematical treatment possible. The construct must not be unduly complicated so that the mathematical treatment can be brought ... to the point where it yields complete numerical results. Similarity to reality is needed to make the operation significant. And this similarity must usually be restricted to a few traits deemed "essential" pro tempore...'

The work involved in generating dynamic process models of individual unit operations and/or the entire process flowsheet is non-trivial. Modelling is inherently a non-intuitive process that can initially be difficult to grasp. Even if the modeller can produce sufficient equations to satisfy the problem, it may still be a difficult task turning them into a working model. To aid with this task work has been done by Barber (2000) to look at the possibility of using computers to create these models. The fundamental laws used in the modelling of deterministic systems[1] are described below.

2.2.1 Fundamental Laws and Empirical Equations used in Modelling Physical-Chemical Systems

The basis for creating models of physical-chemical systems are the fundamental physical and chemical laws. These are:

---

[1] A deterministic system is one which is completely specified as a function of time (Ogata, 1992).

- Continuity Equations – total mass balance and component mass balance.
- Energy Equations – the first law of thermodynamics is applied to all systems where there is some energy change present.
- Equations of Motion – this relates the movement within the system to the changes that are created by this movement.
- Transport Equations – these are of the form of a flux being proportional to a driving force. The proportionality constant is a physical property of the system, e.g. thermal conductivity, diffusivity, or viscosity.
- Equations of State – these are equations that relate the physical properties to each other, e.g. density is a function of pressure, temperature and molar fraction.
- Equilibrium – based on the second law of thermodynamics. There are two types of equilibrium, i.e. chemical, and phase.
- Chemical Kinetics – used to model the kinetics of chemical reactions.

Depending on the purpose of the model many of these laws are not necessary to describe the system to create a model. The greatest drawback of solely using the fundamental laws is the size and complexity of the model, e.g. to model a distillation column with ten components and fifty trays using fundamentals would lead to a model with around 500 differential equations. The other method is to create a working model of a system using empirical equations. These equations are based on physical data and responses to disturbances in the system. So, the distillation column could have an empirical equation relating the reflux flow rate to the distillate composition, without the modeller having to model each tray individually. Marlin, (1995, pp. 196-231), gives a very good overview of using empirical equations within a mathematical model and explains the procedure used to create useful empirical equations.

Once a model has been created it needs to be solved. It has become less of a worry now that the model created, assuming the degrees of freedom are correct, will be able to be solved by the package chosen. The solution of the model created is covered in the next section.

2.2.2 Solution of Mathematical Models

The solution methods used to solve mathematical models are based on work originally completed to allow mathematicians to solve large complex systems of equations by

hand. With the improvement of computing power these methods have been transferred onto computers and have been improved markedly over their predecessors.

This topic is potentially huge and so this will be a brief summary of some methods used at present. Initially mathematical solution methods can be grouped by the types of equation they can be used to solve. The three types of equations are: ordinary differential equations (ODE's), differential algebraic equations (DAE's), and partial differential equations (PDE's). The basic forms of these equations and ways to solve them are discussed at length by Schiesser (1994). Schiesser also states that 'Within the present state of knowledge, general algorithms which can reliably provide solutions to all the major classes of differential equations are not available.' It also needs to be remembered that the solution method used to solve a problem may not give the best results.

The use of an iterative technique is one method used to solve algebraic equations. The key here is finding a method that converges rapidly on the solution. Some of these methods are (from Luyben, 1990): Internal halving, Newton-Raphson, False position, Explicit convergence methods, Muller method, and Wegstein. All these methods use estimates to calculate the next 'step' to hopefully converge on a solution. However, they are all dependent on the initial guess.

Methods for solving systems of PDE's include finite difference methods and orthogonal collocation on finite elements (PSE, 2000 a & b). The usual manner of deriving finite difference methods is based on Taylor expansion of the distributed variables, but other ways have been used, e.g. basing the method on polynomial approximations. An orthogonal collocation method approximates the solution by weighted combinations of orthogonal polynomials, and demands that the describing equations be satisfied exactly at a set of finite points.

2.2.3 Simulation of Physical-Chemical Processes

Once a model has been created, a simulation can be run to see what the outcome is. The simulation of physical-chemical processes is only possible once a mathematical model for that process is available. By using fundamental laws and empirical equations, the mathematical models for many systems can be created. There are some systems, though, for which the task of creating a mathematical model is too complex

and so assumptions are included to simplify the model. It is very important, therefore, that these are stated clearly as an assumption could have a large effect on the final results of the simulation. The simulation of the process is only as accurate as the model used for the simulation, so the less assumptions used then the better the simulation will be.

As simulation packages and the methods used to solve the models have improved, so larger, more complex models may be simulated. Simulation has moved from being just a designer's tool, to being useful in training people on how to deal with dangerous situations on a plant and running start-up and shutdown procedures before the plant has even been built. This 'next step' is helping engineers design, build, and run safer, cleaner, and more user-friendly plants.

2.3 Simulation Packages

A wide variety of simulation packages have been produced over the past forty years and this section aims to look at the different approaches used.

2.3.1 Equation-Oriented and Modular Approaches

Throughout the thesis, the approaches used to model dynamic processes will be split into two different types: *equation oriented* and *modular* (Boston *et al.*, 1993, and Marquardt, 1994). The differences between these two architectures are stated in figure 2.1 with the executive performing a slightly different function by organising the equations and controlling the general purpose equation solver. (Biegler, 1989)

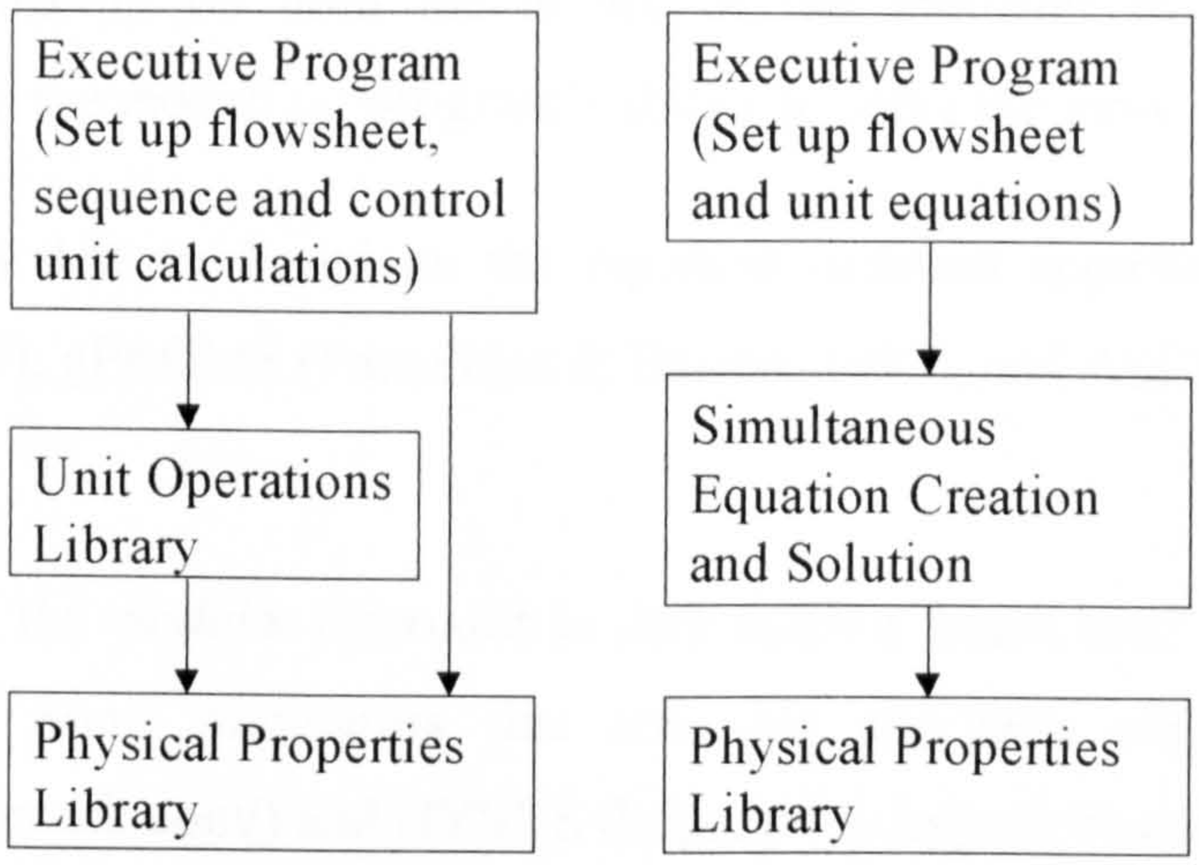


Figure 2.1. Structure of process simulators. Left, sequential modular architecture; right, equation-oriented architecture. (Biegler, 1989).

The different architectures describe the way the model is written and thus how the modelling program deals with it. The largest difference between these two approaches is the amount of information the user sees. The modular approach is very much a 'black box' approach. When picking units from a library the user is unable to see what goes on inside the unit, but just sees it as a black box with information going to and the results coming from it. This approach is often applied for steady state modelling. The equation-oriented approach shows the equations describing the process so the user can adapt the models for each unit to ensure correct fitting to the problem in hand. This approach makes modelling dynamic systems more feasible and accessible.

The *equation-oriented approach* is based around the assumption that the complete model of the plant can be expressed as one large system of equations. The iteration is simultaneously done on all unknowns. The ideas behind this approach are not recent, as they were described by Sargent (1967) and Shacham, *et al.* (1982). But it is only over the past decade that general-purpose process modelling tools based on this approach have reached the stage of being used widely in industry. The main reasons for this delay were the lack of computing power, reliable algorithms, software for the solution of the underlying mathematical problems, and the common practise in industry was to only model steady state behaviour. Due to the nature of the equation-oriented approach, as long as the user can write suitable equations to describe the process, then a dynamic (or steady state) model of some sort can be made. This does,

however, rely heavily on both the ability of the modeller to supply complete equations, and on the modelling program's ability to solve the more complex models.

Some simulators that are based on the equation oriented approach are: SpeedUp[2] (Pantelides, 1988), gPROMS (Pantelides & Barton, 1993), and ASCEND (Piela *et al.*, 1991).

As stated earlier, the *modular approach* is very much a 'black box' approach and can be prescriptive. Some simulators that use this approach are: ASPEN PLUS (http://www.aspentech.com/) and HYSYS (http://www.hyprotech.com).

There are three types of problem that these approaches are used for: simulation, design, and optimisation problems. The modular structure can easily be applied to simulation problems, but difficulties can arise with design and optimisation problems. For the design and optimisation problems, the assumptions used to simplify the process model for the simulation case no longer apply and the use of the modular approach becomes cumbersome.

These two approaches are in no way mutually exclusive, as there are some equation-oriented packages that have a library of units with them. These units can be seen as 'black boxes', although the user can, usually, still see what is going on inside. For example, DIVA (Holl, P. *et al.*, 1988) is an equation oriented system with many of the equations used hidden from the user.

Other ways of grouping these approaches have also been presented by Svrcek, *et al.* (2000) and Marquardt (1994). Svrcek, *et al.* (2000), states that there are six different types of approach (see table 2.1) which have evolved over time as computing power and solution techniques improved.

[2] SpeedUp is now sold by AspenTech as part of Custom Modeller V10.1 (AspenTech, 1999)

Approach	Era	Example System
Analog	(1950 – 1980)	
Hybrid	(1960 – 1980)	
Digital Analog	(1955 – 1980)	
CSMP/CSSL (Continuous System Simulation Language)	(1965 – present)	Simulink (MathWorks, 1999)
Equation Oriented	(1980 – present)	SpeedUp (Pantelides, 1988)
Procedural	(1975 – present)	
Object Oriented	(1970 – present)	
Modular	(1990 – present)	HYSYS (http://www.hyprotech.com)

Table 2.1. Modelling Approaches (Svrcek, *et al.*, 2000).

Marquardt (1994) also groups recent packages into four styles (table 2.2).

Grouping	Example System
General Modelling Language	ASCEND (Piela *et al.*, 1991)
Process Modelling Language	MODEL.LA (Stephanopoulos, *et al.*, 1990 a & b)
Modelling Expert System	PROFIT (Telnes, 1992)
Interactive (Knowledge-Based) Modelling Environment	DIVA (Bär & Zeitz, 1990)

Table 2.2. Classification Groups (Marquardt, 1994).

This method of grouping shows the different ways the packages are moving forward with different methods for solving similar modelling problems.

Another review of the current state of simulation was written by Boston *et al.*, (1993) and was based on where the simulation packages would be in 2001. Again, they split the packages into two approaches: modular and equation-oriented. By 2001, it was felt that these two approaches would be combined to form an object-oriented modelling environment, useable by both experts and novices alike. The other main advance would be in the solution techniques used inside the simulators. With improving models useable for physical properties and increasing data sources the models simulated will be closer to the real world than ever before. It can be seen in current packages that some of these prophecies are coming to fruition, with many advances being behind the scenes (solution techniques and data sources, see § 2.2.2).

2.3.2 Summary

A brief summary of the benefits and drawbacks of each approach is shown in table 2.3. The models available in some of these simulators (within a form of knowledge library) should be used carefully and may need amending, or replacing, to be suitable for use with the users requirements.

Modelling Approach	Modelling Features	Code Generation	Example Systems
Equation-Oriented Approach	Flexible equation based, but little or no guidance or help.	Manual: Time consuming, easy to make errors.	gPROMS, SpeedUp
Modular Approach	Fixed black box models from library. Choice of models and some guidance.	Automatic.	HYSYS, ASPEN PLUS

Table 2.3. Brief review of some current approaches.

Two types of approach have been discussed in this section and the following conclusions may be drawn from this discussion. The greatest difficulty with the textual, equation-oriented approach is that the modeller still has to write all the equations down and then turn them into a program that the package can use. For complex dynamic models, this can lead to many errors and much time can be spent de-bugging the model. The second approach, the modular approach, solves this problem by removing the model writing stage from the modeller. The modeller simply joins units (taken from a library) and then fills in the relevant data for each of the units. The package contains all the equations for each unit and decides which are necessary for solving the problem. This speeds up the modelling process and reduces the amount of time spent de-bugging the model. It is, however, generally harder to model complex (e.g. dynamic) or novel units and plant through a modular system. Hence, as long as the modeller can write the equations for the system it can be modelled by a textual system, but with the modular system, incorporating these equations can be a very complex task, for which the user may need assistance.

For the inexperienced modeller, the library of units available in the modular approach also act as a pointer should they become stuck or unsure how to continue. There is

also better provision for on-line help as the package knows what information it needs to solve a particular problem and so can prompt the modeller for this information.

The main advantage to the textual approach, though, is that the modeller can see exactly what they are modelling and will know that the model input is exactly what they wanted. This is important for non-standard pieces of equipment and units. The more experienced a modeller is, the more likely they are to use a modelling package with a textual interface as this allows them more control over the total process. They can see exactly what equations are used and the way the problem has been tackled.

These two approaches are at either end of the modelling package spectrum (see table 2.3), i.e. the equation-oriented approach requires considerable expertise, whereas the modular approach is led far more by an expert system. What is required is an approach that draws on the positive aspects of these two approaches and integrates them to provide a user-friendly modelling interface that can be used for both steady state and dynamic modelling. In the next section some user interfaces are described showing how they aid the modeller in their task.

2.4 Interfaces, Front Ends and Translators

In order to reduce the complexity of creating a model in a simulation package or solver, many are being produced with their own front end packages. The aim of the interface is to simplify the modelling task by aiding the user with the creation of the input file that the simulator or solver uses. These interfaces are usually graphical in nature and aid the user by including much information (such as the equations needed to solve a particular unit) behind the scenes of the graphical unit. The user is then prompted to input certain data to solve that unit.

The use of a front end to simplify the modelling task has also been used in ecological modelling (Ushold *et al.*, 1984).

2.4.1 Aspen Plus and HYSYS

Two of the most well known packages that include a user interface are Aspen Plus (http://www.aspentech.com/) and HYSYS (http://www.hyprotech.com). Their interfaces are shown in figures 2.2 and 2.3 below.

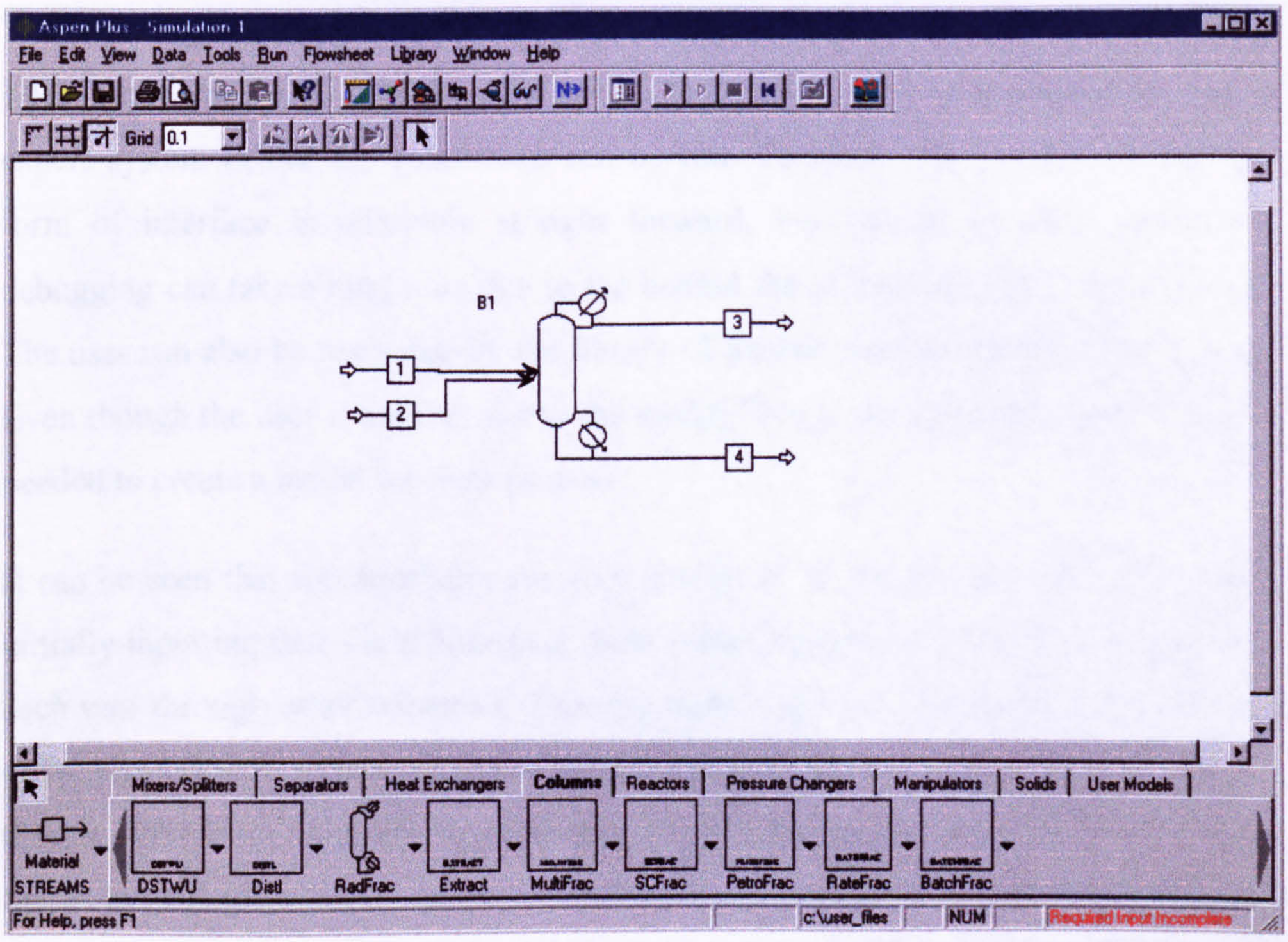


Figure 2.2. Aspen User Interface.

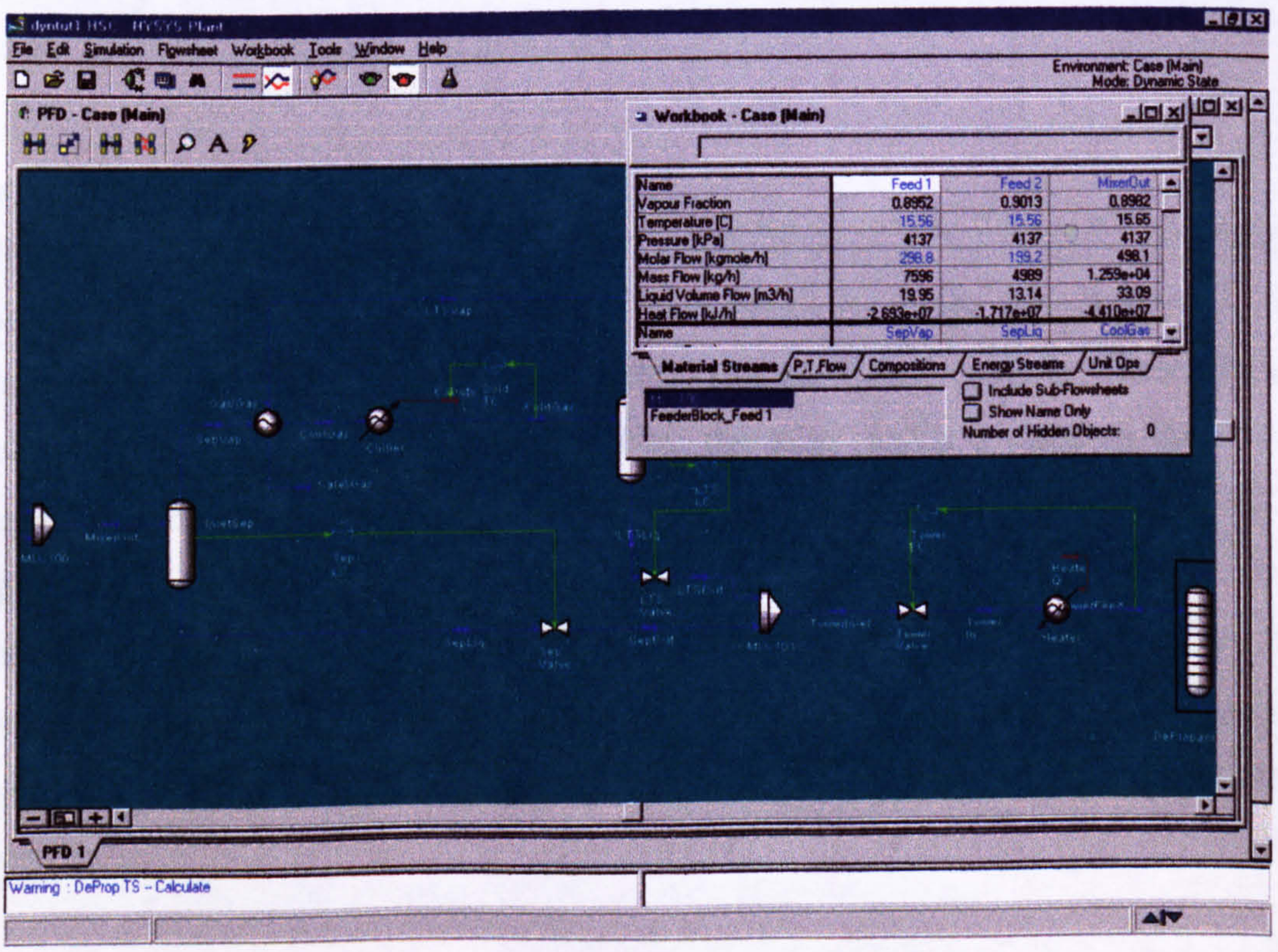


Figure 2.3. HYSYS User Interface.

These packages use their interfaces as a tool to manipulate the 'black box' units. These units are linked and the information the solver needs is prompted for via an expert system before the simulation can be run. To create the models through this form of interface is relatively straight forward, but, should an error occur, the debugging can take a long time due to the behind the scenes nature of the modelling. The user can also be restricted by the library of models available within the package. Even though the user is able to add to the model library, they may not have the skills needed to create a model for their process.

It can be seen that the interfaces are very similar in layout and design with the user initially inputting data via a flowsheet, then being prompted for more information for each unit through other windows. The two main ways that Aspen Plus and HYSYS differ are: HYSYS interactively deals with the users input; the user does not need to run the simulation manually (by pressing a 'run' button) and, secondly, HYSYS has the ability to allow information to propagate in both the forwards and backwards directions. This allows for less iterative steps and shorter solution times. One property the majority of these simulators have is the ability to run small parts of the whole system fairly easily through unit selection and grouping.

2.4.2 Ascend

Another type of interface is used by ASCEND (Piela *et al.*, 1991). This interface guides the user through the creation of their model by using pre-set objects, for example, relation, distance, volume, factor, and cost_per_volume. These objects have certain properties that are inherited by the item that is a child to them. For example, in figure 2.4 D IS_A distance means that D inherits all the properties of type distance: dimension, lower and upper bound, nominal and default value. This front end incorporates the benefits of the modular system with many pointers for the user to follow, it does, however, place an extra level of complexity on the modelling process as the user must learn how to use this interface to benefit from its full potential.

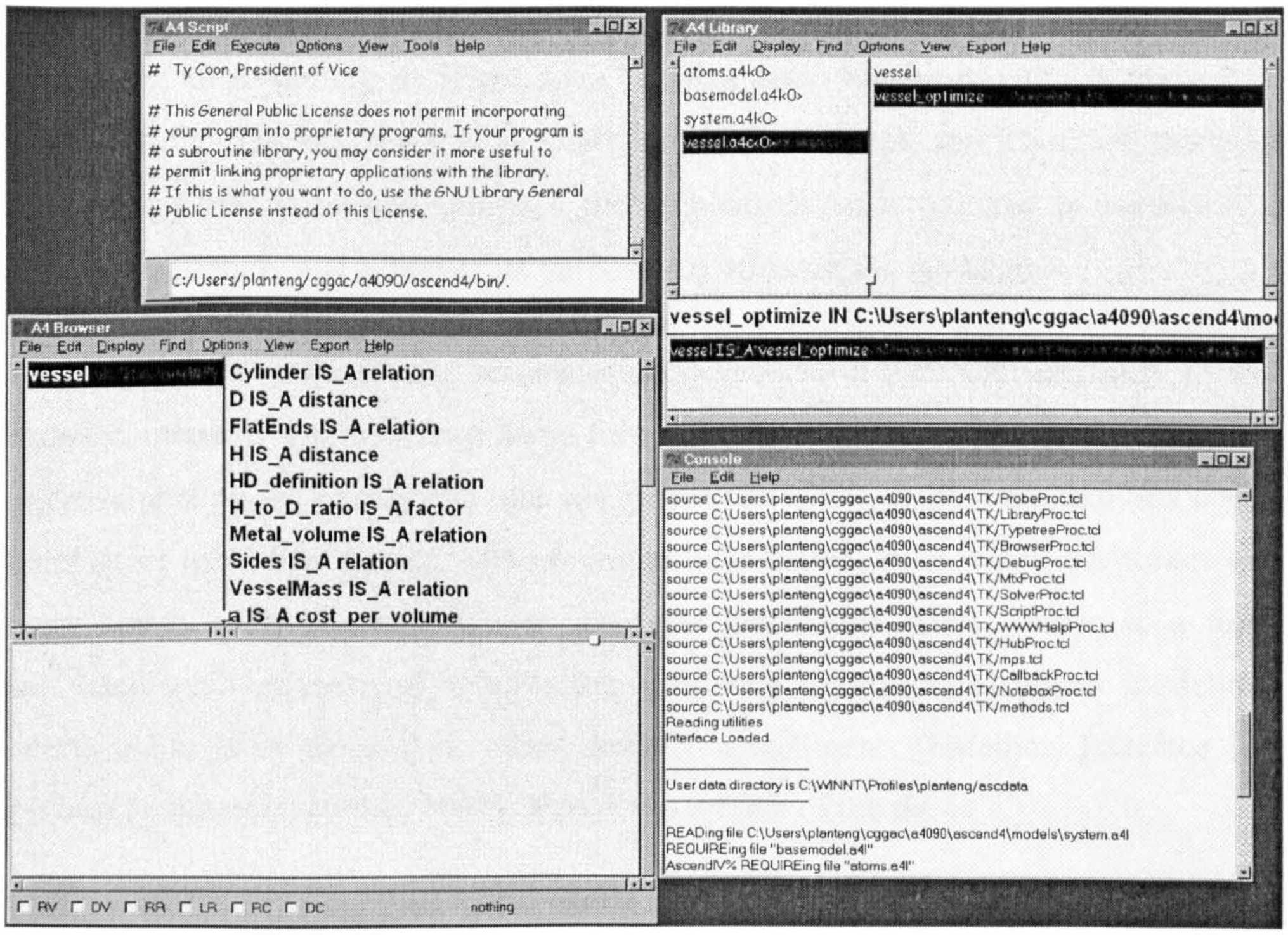


Figure 2.4. ASCEND User Interface.

2.4.3 Translators

The use of translators to aid the modeller have been available for a number of years. In some respects, the front ends and interfaces are translators that turn one representation of information into a model for the solver. Other examples of situations where translators have been used are:

- A translator to turn Simulink models into VisSim models (http://www.adeptscience.co.uk/ products/mathsim/vissim/). VisSim is a program for the modelling and simulation of complex dynamic systems. The simulation engine within VisSim provides fast and accurate solutions for linear, nonlinear, continuous time, discrete time, time varying and hybrid system designs.

- The automatic translation of a Simulink model into executable code to be used in engine management systems (Maclay, 2000).

2.5 Conclusions

Ultimately, all modelling packages have to solve many equations and use numerical methods to do this. It is merely the interface that is different, and from that, the way the package (and modeller) approach the problem. What is required is an interface that is user friendly and achieves its purpose, i.e. to assist the modeller.

An interface that incorporates automatic code generation with the flexibility of the equation-oriented approach, and some form of help or knowledge base would be an improvement on the packages in use today (see table 2.4). A package like this could be of great use when dealing with corporate information, as it would encourage the reuse and possibly re-evaluation of corporate knowledge. Thus leading to a fully annotated evolving base of information with time. Such an intelligent modelling interface has been developed, called **IMIPS**, (**I**ntelligent **M**odelling **I**nterface for **P**rocess **S**imulators) (Clark, 1998). This is the subject of the thesis.

The knowledge base and knowledge retrieval involves case-based reasoning and so in the next chapter, case-based reasoning is reviewed and it is shown how its methodology could be applied within an interface such as IMIPS.

Modelling Approach	Modelling Features	Code Generation	Example Systems
Equation-Oriented Approach	Flexible equation based, but little or no guidance or help.	Manual: Time consuming, easy to make errors.	gPROMS, SpeedUp
Modular Approach	Fixed black box models from library. Choice of models and some guidance.	Automatic.	HYSYS, ASPEN PLUS
NEW: Case-Based Reasoning Approach	*Flexible equation based. Provides guidance for model.*	*Automatic.*	*IMIPS*

Table 2.4. Brief review of some current approaches and proposed approach.

3. Case-Based Reasoning

3.1 Introduction

To incorporate the benefits of the two approaches into one, we have incorporated case based reasoning ideas to allow us to intelligently search through a database of past cases. A case is an example of a chemical engineering problem, with the information to index and solve that problem. This is to allow the less experienced user to have a knowledge base to start their work from, much as an experienced engineer bases some of their decisions on their past experiences.

The layout of this chapter is as follows: In §3.2 a brief explanation of the background behind case-based reasoning is given. § 3.3 outlines the advantages and disadvantages of case-based reasoning. §3.4, §3.5, and §3.6 outline the indexing, retrieval and adaptation methods used at present. The chapter finishes with some conclusions that can be drawn about case-based reasoning.

3.2 Background

If a person bases any decisions they make on past experiences or examples that they have seen, then they are using case-based reasoning. Case-based reasoning is a type of reasoning that incorporates problem solving, understanding, and learning, and integrates all with memory processes. (For the general case-based problem solving process, see figure 3.1.)

A case to be included in the case memory must be relevant to the system and so be able to be indexed using the scheme present. Features of the case are identified and these are used to base the indexing of the case. If a case is to be retrieved, the features that are relevant are found and a search carried out on those parameters.

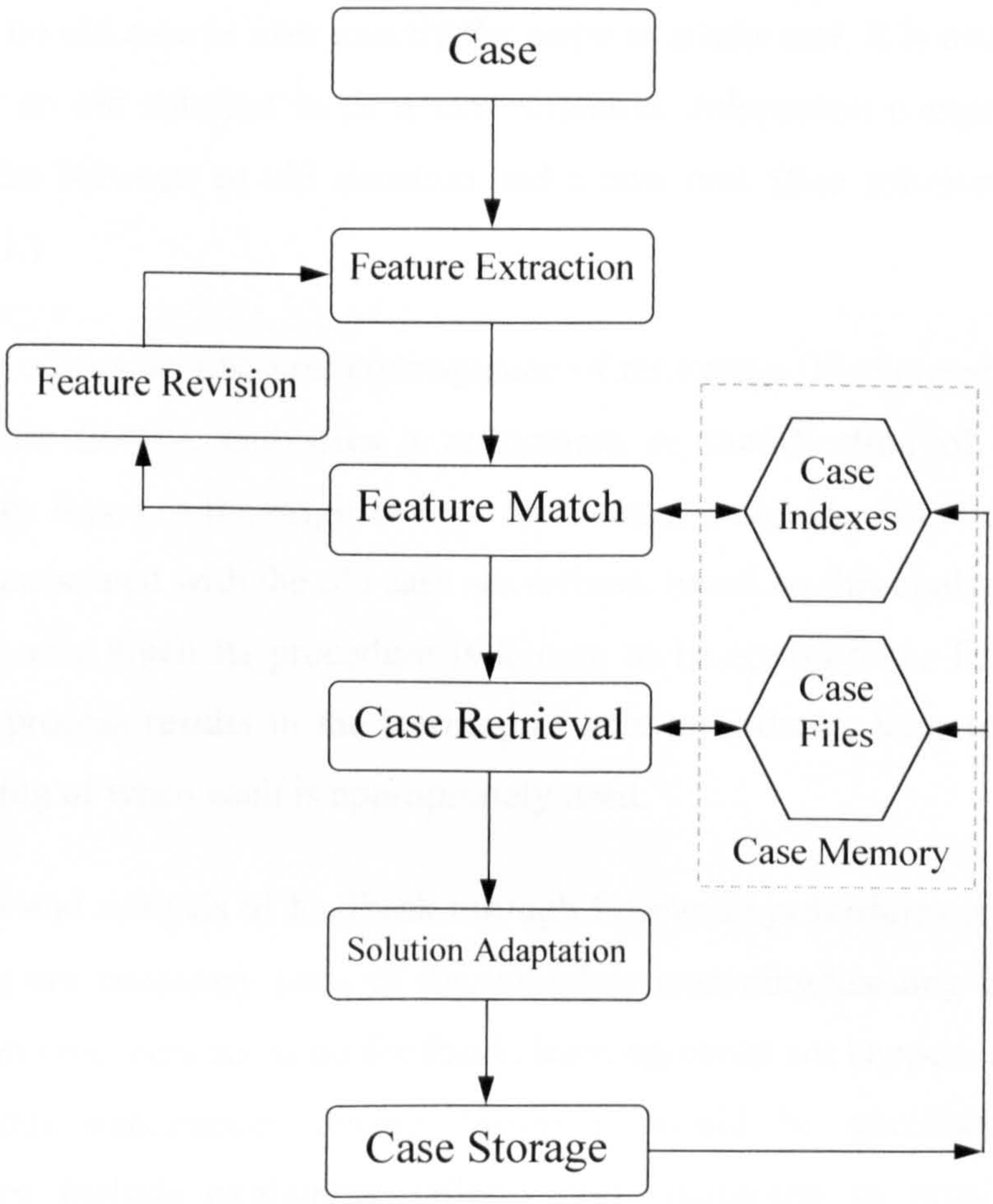


Figure 3.1. Process of case-based problem solving.

The following premises underlie this type of reasoning and will be discussed in detail later, Kolodner (1993):

- Reference to old cases is *advantageous* in dealing with situations that recur. Reference to previous similar situations is often *necessary* to deal with the complexities of novel situations. Thus, remembering a case to use in later problem solving (and integrating that case with what is already known) is a necessary learning process.

- Any form of reasoning requires that a situation be elaborated in enough detail and represented in enough clarity and with appropriate vocabulary to allow the reasoner to recognise the knowledge it needs (whether general knowledge or cases) to reason about it. (See feature extraction, matching and revision in figure 3.1.)

- Because no old case is ever exactly the same as a new one, it is usually necessary to *adapt* an old solution to fit a new situation. Adaptation compensates for the differences between an old situation and a new one. (See solution adaptation in figure 3.1.)

- Learning occurs as a natural consequence of reasoning. The new situation, stored in the case library, embodies a refinement or modification of the reasoning knowledge found in the original case. Its indexes designate when it is useful, and indexes associated with the old case are refined, based on this analysis, so that it is retrieved only when its procedure is known to be appropriate. This incremental learning process results in the learning of new procedures, their refinement, and the learning of when each is appropriately used.

- Feedback and analysis of feedback through follow-up procedures and explanatory reasoning are necessary parts of the complete reasoning/learning cycle. Without evaluation processes based on feedback, learning could not happen, and references to previous experiences during reasoning would be unreliable. Follow-up procedures include explaining failures and attempting to repair them. (See feedback loop from case storage to case indexes and case files in figure 3.1.)

In addition, the quality of the case-based reasoner's reasoning is based on:

- The experiences it has had.
- Its ability to understand new situations in terms of those old experiences.
- Its adeptness to adapt.
- Its adeptness at evaluation and repair.
- Its ability to integrate new experiences into its memory appropriately.

Due to the continuous learning of the system, it becomes more efficient and more competent with the more problems it attempts. Case-based reasoning achieves most of its learning through accumulation of new cases and assignment of indexes. Due to the continuous learning of the system, it becomes more useful for solving new problems.

3.3 Advantages and Disadvantages of Case-Based Reasoning

The advantages and disadvantages of case-based reasoning are outlined in the following sections.

3.3.1 Advantages of Case-Based Reasoning

Case-based reasoning allows a reasoner to propose solutions in domains that are not completely understood by the reasoner. It also allows the reasoner to propose solutions to problems quickly, avoiding the time necessary to derive those answers from scratch. Previous experiences are particularly useful in warning of the potential for problems that have occurred in the past, alerting a reasoner to act to avoid repeating past mistakes.

Cases help a reasoner to focus its reasoning on important parts of a problem by pointing out what features of a problem are the important ones. They are also useful in interpreting open-ended and ill-defined concepts.

These systems are easier to develop than other knowledge based systems, as knowledge acquisition is simplified to a task of adding the cases to the library and indexing them appropriately.

3.3.2 Disadvantages of Case-Based Reasoning

A case-based reasoner may be tempted to use old cases blindly, relying on previous experience without validating it in the new situation or to allow old cases to bias the line of reasoning. For example when designing a high pressure reactor vessel, the designer may use past experience to decide the size, shape, and mechanical properties of the vessel. However, if the designer's experience is in designing reactor vessels for low pressure use, some of the experience will be useless as the mechanical design of the vessel will be different.

Many case-based reasoning systems are based in domains where case information can be represented at a chosen level of abstraction. However, for many design cases it is necessary to decompose the case into multiple levels of abstraction. For example a simple reactor system has, at one level, overall mass, component and energy balances, whereas at another level what actually happens inside the reactor is not included in the detail in the overall balances. (See also Tsatsoulis & Alexander, 1997.)

3.4 Case Indexing

A very important component of a case-based reasoning system is the indexing scheme used to index and compare cases. The indexing scheme needs to be as detailed as possible to cover all possible eventualities of cases. This leads to a very precise description of a case, which can also lead to easier case matching and retrieval.

An indexing scheme is judged by the following four properties (Maher *et. al.* (1995), pp. 87-88):

- Prediction: the index must capture the aspects of the cases that tend to predict solutions and outcomes of cases. The indexing vocabulary should include the aspects of the problem that may be critical in determining the problem solution.

- Specificity: the vocabulary must be specific enough to make all useful discriminations between items in the library. The indexes should also be concrete enough to be easily recognisable later. The level of detail in the vocabulary should be at a level where two similar designs can be distinguished.

- Generality: the vocabulary must be general enough to capture the similarities between cases. This is so that cases can be used in a number of situations.

- Usefulness: recalling cases from the case base should produce useful cases. This is possibly the most difficult part of the index vocabulary to design as the usefulness is usually only apparent to the user of the case-based reasoning tool.

Indexing can be done automatically or manually, and are discussed below.

3.4.1 Automatic Indexing

The four main automatic indexing methods used in existing packages are: indexing by features or dimensions, by differences, by similarity, and by using inductive learning methods. These are described in more detail below.

- The first method is to index the cases by using features or dimensions that tend to be predictive across the entire domain. This is a very useful approach and there are many problems where this can be used, e.g. CHEF (Hammond, 1986) is a system that creates new recipes from existing ones, indexing on features like taste

and texture. Unfortunately, when the problem cases are based on more abstract ideas, the features do not usually encompass the whole problem and so are only partially useful in indexing the problem.

- Difference based indexing, which selects indexes that differentiate one case from another. This is very useful where cases are very similar and it is easier to list the differences than the similarities, for example, in CYRUS (Kolodner, 1983a, 1983b) events were stored and then retrieved from the life of Cyrus Vance when he was secretary of state of the United States (under President Jimmy Carter). All the events were similar and so indexing on their differences is a logical way of dealing with the problem.

- Similarity and explanation-based generalisation methods produce a set of indexes for abstract cases, from cases that share some common set of features. The unshared features are then used as indexes for the original cases. This method needs a large case base of similar cases that the indexer uses as a base line.

- Inductive learning methods identify predictive features that are then used as indexes. Some features that run through the cases to be used are selected (by induction) and these features are used for the indexing of the cases. Due to the inductive nature of the process the features can be modified, the more cases there are in the case library. This method, again, needs the cases to be similar in certain ways for the features to be picked up by the indexer.

3.4.2 Manual Indexing

Manual indexing is a fine art. To create an index that incorporates all the qualities listed above (prediction, specificity, generality, and usefulness) manually is an extremely complex task. The methods used for this task can be split into either description or relationship schemes (Maher, *et. al.* (1995)), which are explored in more detail below.

3.4.2.1 Description Schemes

If designs are indexed by a set of surface features, e.g. length, material, mass, etc., then these are considered as descriptive indexing schemes. This type of scheme often

adopts a fixed structure to describe and represent cases. The basic structure used to organise these descriptive features is a list. This is explored below.

In a list organisation of the indexing vocabulary, features are regarded as elements of a list. An element of a list can be represented by a single indexing feature or by a set of indexing features. Cases are assigned to elements of the list, according to the description of cases. Each element indicates a set of cases that can be descriptively labelled by it (figure 3.2).

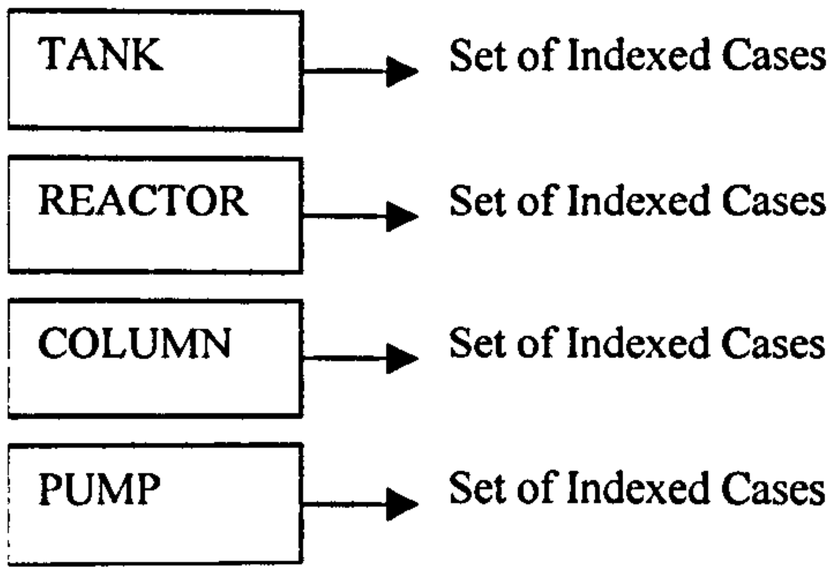


Figure 3.2. Example of a list.

3.4.2.2 Relationship Schemes

In many situations, the relevance of one case to another does not always depend on the surface features, but may depend on abstract relationships. These relationships may be feature-based, object-based, or graph-based. (For examples see table 3.1.)

Feature-based	The way two features are related, e.g. mass and volume.
Object-based	The way two objects are related, e.g. a tank and a reactor.
Graph-based	Two items are related by a graph, not an equation, e.g. flowrate and N_{RE}.

Table 3.1. Examples of abstract relationships.

The object of the relationship indexing scheme is to add several models or rational knowledge to facilitate the recognition of the relevance of cases. This type of scheme requires some form of domain knowledge. Based on multiple indexing paths to the same cases, relationship indexing schemes provide flexibility in indexing cases. The basic structure used in this method is a tree or hierarchy.

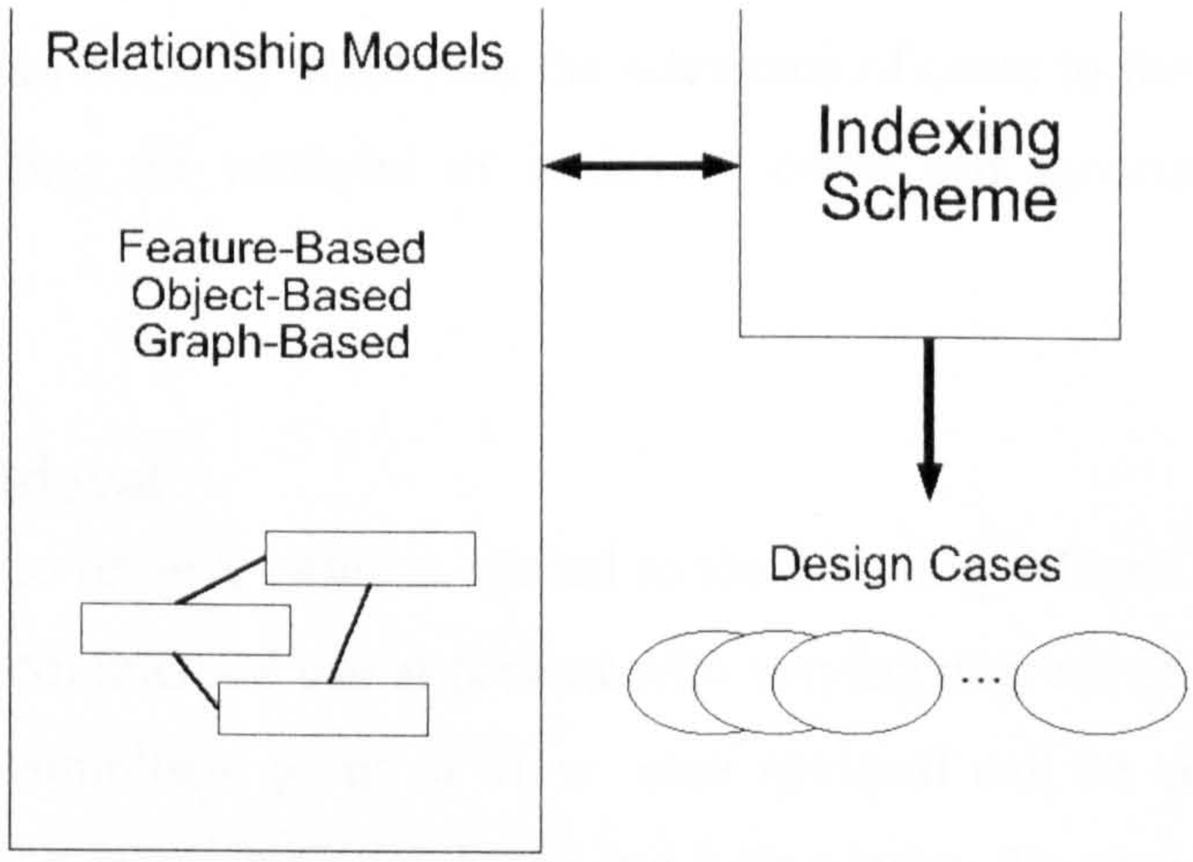


Figure 3.3. The relationship-based indexing scheme.

In a tree, features of the indexing vocabulary are distributed in the nodes of a hierarchical tree. The tree can be generated by clustering cases according to the indexing features. Cases are indexed in the tree from more general to more specific descriptions. The search of this indexing scheme can also be simplified more by only including the 'leaf' nodes of the tree.

Using hierarchies can simplify the indexing of a problem by automatically assuming some form of property inheritance from parent to child down the hierarchy, is-a relationships. For example, (see figure 3.4.) a **Jacketed Vessel** is a special form of **Tank**, and as such has many of the **Tank**'s attributes. Likewise, **Tank** will have many of the features of **EQUIPMENT**, its parent, and so **Jacketed Vessel** will have many of the properties of **EQUIPMENT**.

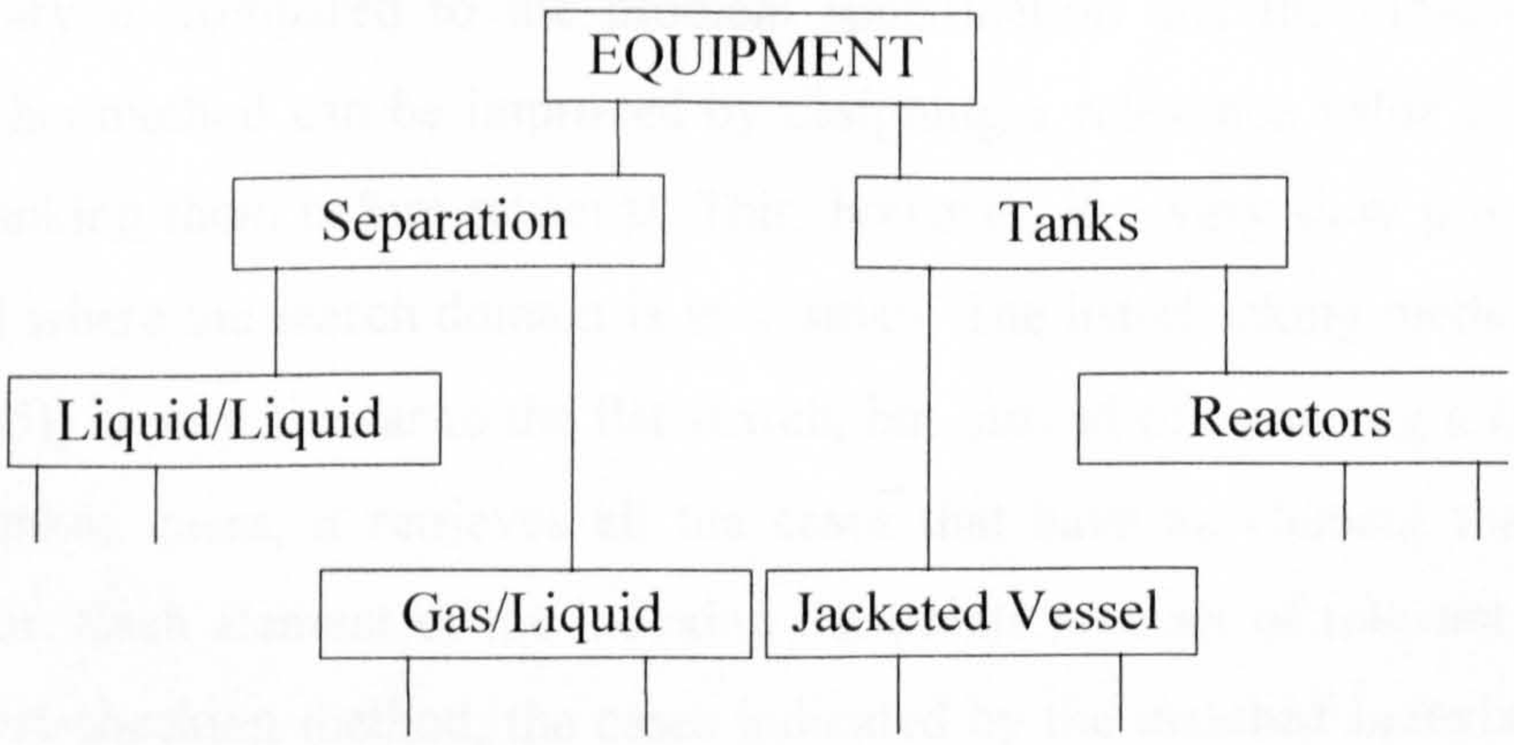


Figure 3.4. Example of a hierarchy.

Using a set of surface features from the description of cases is the simplest way to index cases. However, there is one main problem when using this simple approach.

Surface features can not fully anticipate the relevance of cases to the current problem, consequently causing the retrieval of irrelevant cases and ignoring some relevant ones.

3.5 Case Retrieval

The method used to retrieve cases is related to the indexing scheme in use. There are many schemes in commercial use at present with varying degrees of usefulness to our problem. From a simplistic point of view, case retrieval can be seen as a massive search problem (like searching a database), but with a twist. No case in the library can ever be expected to match a new situation exactly, so the search must result in retrieval of close partial matches. Due to this problem, there also needs to be some way of ranking the relevance of the retrieved cases to the new situation. Database search methods are used to compare values in fields to those in the database. A search for primary colours would yield *red*, *yellow* and *blue* in the colour field, and a search for *blue* in the same field would only retrieve those entries with *blue* in that field. Entries with *navy* or *cobalt* in that field would not be retrieved, though both are close matches to *blue*. The retrieval method, therefore, should also be capable of selecting and ranking partial matches to the initial problem. Likewise, when searching fields for numerical values some tolerance needs to be included in the search.

3.5.1 Flat Search & List Checking

As Kolodner (1993) stated, a flat search is the simplest search to perform. Each case in the library is compared to the problem specification and the closest match is selected. This method can be improved by assigning a relevance value to each case and thus ranking them before retrieval. This, however, is a very slow process and is only useful where the search domain is very small. The list-checking method (Maher, *et. al.* (1995)) is very similar to the flat search, but instead of retrieving a single case, or some ranked cases, it retrieves all the cases that have an element that is being searched for. Each element of the indexing list points to a set of relevant cases and using the list-checking method, the cases indicated by the matched indexing element are retrieved as relevant cases. Both these strategies are based, in essence, on a feature match.

3.5.2 Concept Refinement

Concept refinement uses a tree indexing representation. The search starts at the most general feature in the tree and progresses downwards only when a match is possible. A match is where the description of the new problem matches the indexing features of that particular node. This method can be approached in two ways: a breadth first or depth first search. These are shown in the following diagrams. A breadth first search would search across the hierarchy before moving down to the next level, so using figure 3.4 the search order would be: EQUIPMENT, Separation, Tanks, Liquid/Liquid, Gas/Liquid, Jacketed Vessel, Reactors, etc.

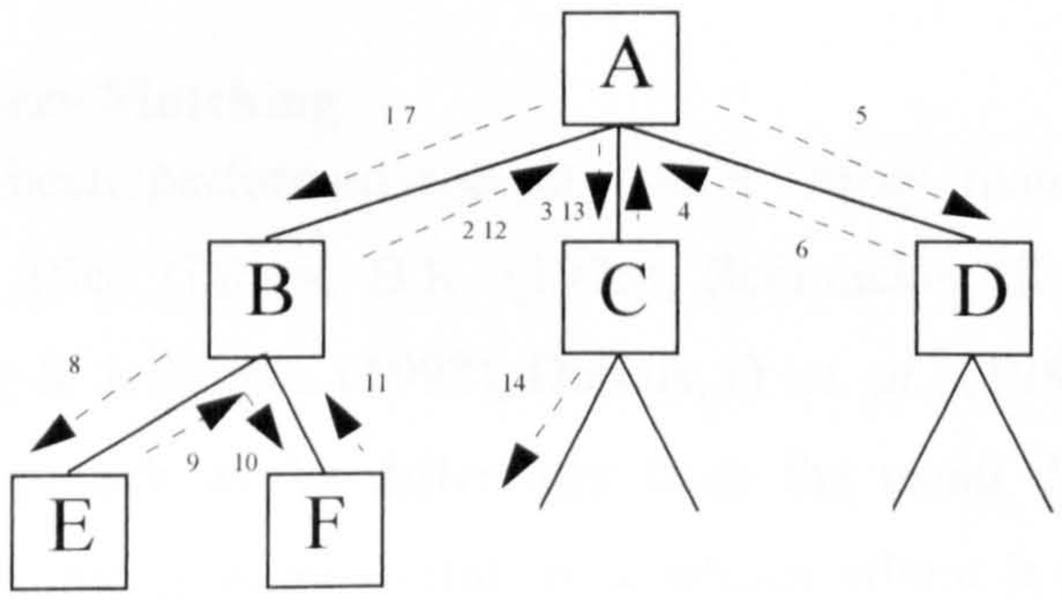


Figure 3.5. Breadth-first search.

The depth first search runs down a branch before returning to the highest unexplored branch, again for example, using figure 3.4: EQUIPMENT, Separation, Liquid/Liquid, …, Gas/Liquid, …, Tanks, Jacketed Vessel, …, Reactors, etc.

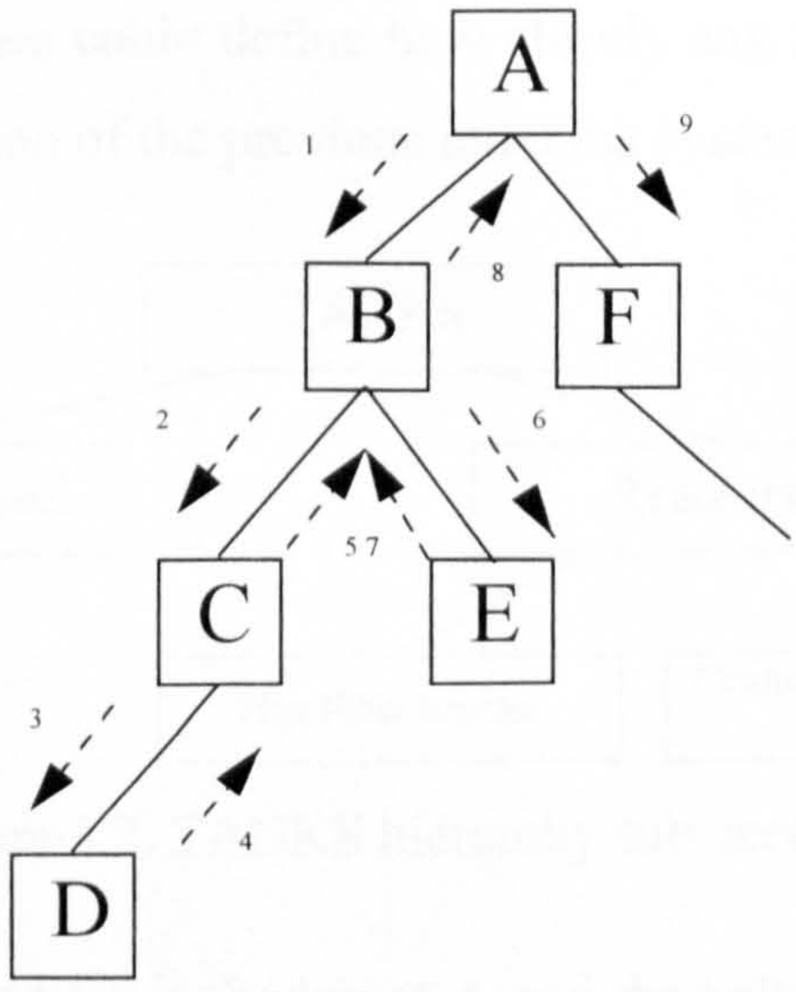


Figure 3.6. Depth-first search.

3.5.3 Associative Recall

This method considers the relationship-based indexing schemes. (See Maher, *et. al.* (1995)) Cases are recalled through elaboration and explanation of the new problem specification, based on the relevant generalised models. In this method, cases are recalled based on a basis of the most relevant match, which is not necessarily the closest match. This is due to the retrieved cases' relevance to the current problem not being based on just surface features, but also on a deep feature match based on domain knowledge. For this, domain knowledge is vital to guide the search to the most relevant matches.

3.5.4 Partial or Fuzzy Matching

Much research has been performed recently on a 'fuzzy matching' approach to searching databases. (See Gaines, B.R. (1976), Schmucker, K.J. (1984), Jeng & Liang, (1995), Chung & Jefferson, (1998), Dubois, D. *et. al.*, (1988), Koiranen, T. *et. al.*, (1998).) This approach works differently from the usual database search and match string methods and is more useful in situations where it is unlikely that the same keywords/values will be used to describe similar, but slightly different, problems. The two types of data are dealt with in slightly different ways.

3.5.4.1 Symbolic Data

One way of searching and ranking symbolic data follows. If the items in the index were in a hierarchy then we could define how closely any two items are related. For example the TANKS section of the previous indexing system (figure 3.4).

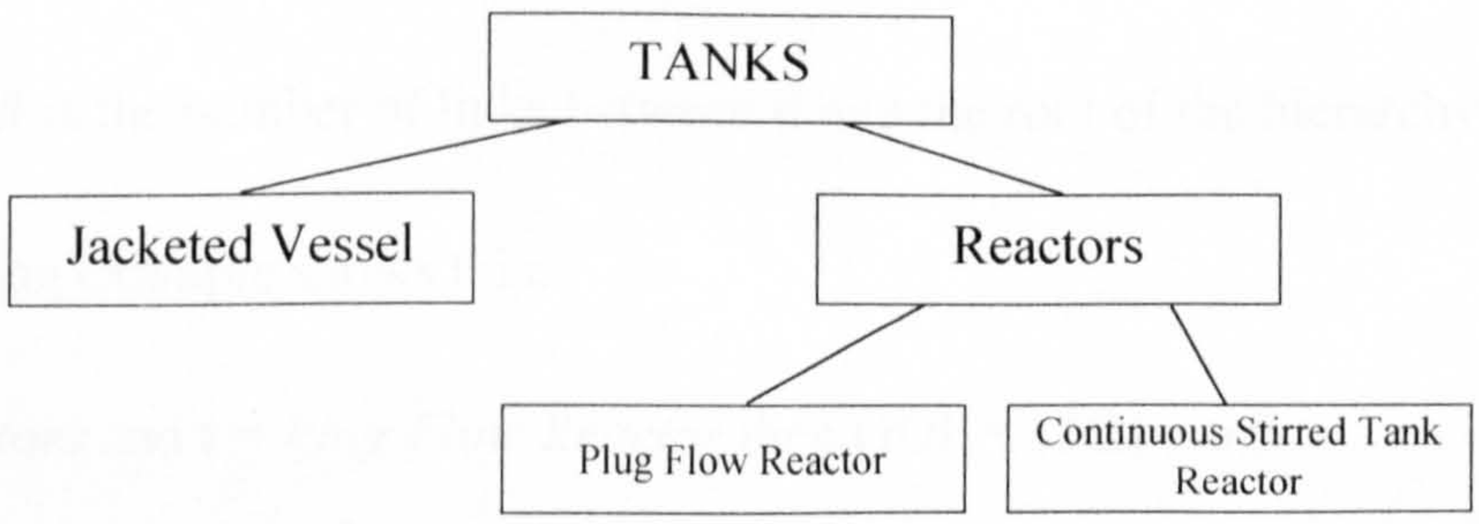


Figure 3.7. TANKS hierarchy sub-section.

The attribute to be searched for is the target, **t**, and the value stored for that attribute, **d**. The goodness of fit (GOF) between these two attributes can then be calculated.

Chung & Jefferson, (1998) found four categories with four different methods for calculating the GOF.

d is the same as **t**.

d is a descendant of **t** in the hierarchy.

d is an ancestor of **t** in the hierarchy.

d and **t** are on different branches in the hierarchy.

For category 1, it is obvious that **d** matches **t** with GOF = 1.

In category 2 **d** is a narrower term than **t**. Therefore, we can also define **d** matches **t** with GOF = 1.

In category 3 **d** is more general than **t**. In this case **d** may or may not be what the user is searching for. Using the approach used in ReMind (a case-based reasoning tool) the GOF can be calculated as below:

$$\text{GOF} = \frac{(IP + ER)}{(IR + ER)}$$

Where *IP* is the number of links between **t** and the common parent between **t** and **d**.

EP is the number of links between **d** and the common parent between **t** and **d**.

IR is the number of links between **t** and the root of the hierarchy.

ER is the number of links between **d** and the root of the hierarchy.

The following example shows this:

If **d** = *Reactors* and **t** = *Plug Flow Reactor* then GOF = (1/2) = 0.5

If **d** = *Tanks* and **t** = *Reactors* then GOF = (1/3) = 0.3

If **d** = *Tanks* and **t** = *Plug Flow Reactor* then GOF = (1/2) × (1/3) = (1/6) = 0.17

In category 4 **d** and **t** are on different branches in the hierarchy. It is inappropriate to apply a general rule to determine the GOF value in this category as nodes on different

branches in a hierarchy may or may not be related. It is therefore left to the expert to determine how closely the nodes are related, if at all.

When returning results from a query there needs to be some form of ranking to allow the user to look over the more relevant results first. This can be achieved by a number of methods. When searching for a number of attributes an average or weighted GOF could be used to determine the relevance to the problem. (As discussed by Dubois *et. al.*, (1988).)

3.5.4.2 Numerical Data

When matching numerical values the GOF is calculated in relation to how close the value is to that being searched for. When matching a target, *t*, against a data value, *d*, the value of the GOF should decrease as *d* moves away from *t* (Chung & Inder, (1992). A linear approximation could be used falling from a maximum of 1 to a minimum of 0, e.g.:

$$\text{GOF}(t,d) = \max\left(0,1 - \frac{|t-d|}{K}\right)$$

Functions that are more complex have been derived to remove the sharp cut-off point or to move the emphasis to near misses, e.g. (Schmucker, (1984)):

$$\text{GOF}(t,d) = \left[1 + \left(\frac{|t-d|}{\sqrt{t}}\right)^2\right]^{-1}$$

If *t* represents the upper bound then if $d <= t$ the GOF = 1, otherwise the above equation (or another) holds, and likewise if *t* represents the lower bound, the reverse behaviour is expected.

3.6 Case Adaptation

As for indexing, case adaptation can be automatic or manual.

3.6.1 Automatic Adaptation

Another vital role of the system is the ability to adapt past solutions to solve the present problem. This adaptation is recorded via feedback either from the user or

automatically on set parameters. The logging of how successful the adaptation was is also vitally important as the system can then learn from its mistakes.

Three methods of adaptation are substitution, transformation, and derivational analogy methods. Substitution methods choose and install a replacement for some part of the old case that does not meet the current problem requirements. Transformation methods use heuristics to replace, delete, or add components to an old case in order to make the old case fit into the new situation. Derivational analogy methods use the method of deriving the old solution to then derive a solution to the new problem.

Kolodner (1993) noted that many design problems are very hard to decompose due to many of the components having strong relationships between them. But, to ease the solving of large or complex design problems, the problem needs to be decomposed. Thus, the solving of many large and complex problems starts with the need to decompose them into smaller, more easily manageable parts. One problem leading from this is that if one part of the problem is solved it could change the values and properties of other parts of the total problem.

CHEF (Hammond, 1986) is a menu/recipe planner that has an automatic adaptation step. Its input is a collection of sub-goals that need to be achieved and its output is a meal plan. CHEF recalls old plans that satisfy most sub-goals and then adapts them to fit the other sub-goals. CHEF indexes its plans by the goals that they achieve, e.g. some of the goals, beef and broccoli, are indexed by *include beef, include a crisp vegetable, use method stir-fry,* and *achieve taste savoury*. This increases the chances of getting a close match to the problem from the case library.

The next step CHEF undertakes is to adapt the old plan to fit the current situation. First it reinstantiates the old plan. That is, it creates an instance of the old plan that substitutes new objects for the ones used previously, e.g. if it is creating a chicken recipe from a beef recipe, it substitutes chicken for beef. For this, CHEF needs to know something about the roles of each of the recipes constituent parts. It knows that both chicken and beef are defined as meat and so the substitution is valid.

The second adaptation step involves applying special 'object critics' to the plan. These add special preparation steps to the plan that may be only relevant to the new

constituents, e.g. deveining shrimps, defatting, deboning, etc. These critics are the way CHEF stores specialist knowledge about its domain.

After these two steps are complete, the plan is ready to be used. CHEF has completed its job, but without some form of feedback it will not learn from its success/failure and so will not improve over time and use. CHEF has its own simulator that can run a planned meal and gives feedback similar to that which would be given in the real world.

3.6.2 Manual Adaptation

This is the easiest method to apply as the user adapts the cases retrieved by the package. It is important, though, that the user keeps a record of the changes made and the successes and failures that arise from them. It is from these successes and failures that the reasoner can learn which cases are more relevant for some abstract problems. This form of adaptation assumes the user has some form of knowledge about the problem at hand and so is not particularly useful for applications to be used by novices.

3.7 Conclusions

This chapter has highlighted the benefits of case-based reasoning techniques when used to draw knowledge and cases from a library. As the natural way of approaching most engineering problems is to call upon the experience of the modeller or another expert to help with the creation and solving of the problem, an automation of this activity would enable solutions to be produced faster and with fewer errors.

Case indexing is more of an art form than a science. It takes a great deal of knowledge to create a predictive, robust, abstract, and useful index that will cater for all problems encountered. Most indexing schemes involve a great deal of evolution from the initial ideas to a useable index. Whether an automatic or manual indexing system is used is highly dependent on the domain of the study. For domains with rigid guidelines, or fixed parameters to be known, then the automatic generation of the indexes is possible. However, within many domains the automatic generation of indexes is far too complex. The translation from a natural language problem description to a description that can be easily indexed is one where the human brain is still far more

efficient than other, automatic methods. Within our domain of interest, the cases are to be stated in a very structured and formal way with the information written in pre-defined fields in a database, and so these fields can be used as the index for the system.

Cases can be retrieved using one of two search methods: the usual 'database' field search, and a partial or fuzzy search. As the possibility of a case in the library being exactly the same as the current problem is virtually zero, then a search that draws out all cases that might be slightly relevant is better than one searching for the exact answer. For this reason, a partial or fuzzy search is a far better way of drawing the most appropriate cases from the library. This retrieval procedure should also have a method of ranking the retrieved case so that the more relevant ones are drawn to the user's attention before those that are less similar. The search should differentiate between numerical and symbolic values and vary its search techniques accordingly.

Automatic case adaptation is a very powerful method of solving problems, but only within a limited domain (such as that in CHEF). To automate the process over our domain seems, at present, virtually impossible with current methods and technologies. It would therefore seem more appropriate to concentrate on manual adaptation of cases and concentrate on the indexing and retrieval side on the case-based reasoning tool.

4. System Overview

4.1 Introduction

This chapter outlines the system produced. The system is a front end (or interface) for users of process simulators who may, or may not, have had previous experience in their use. The front end is interfaced to a database of past cases through which the user may search and then use some or all of those retrieved cases as a base for their problem solution. The front end incorporates a mechanism to create a valid search query, a system to view all retrieved cases, and translators to create simulator input files.

The layout of this chapter is as follows: In §4.2 the system components are outlined, §4.3, §4.4, §4.5, §4.6, and §4.7 outline the main stages for the process, i.e. case specification, case retrieval, case review, case adaptation, and case translation, and finally in §4.8 the procedure is demonstrated for a general case.

4.2 System Components

The system outline is shown in figure 4.1. This shows the three sections of the system, and the output produced (a simulator input file).

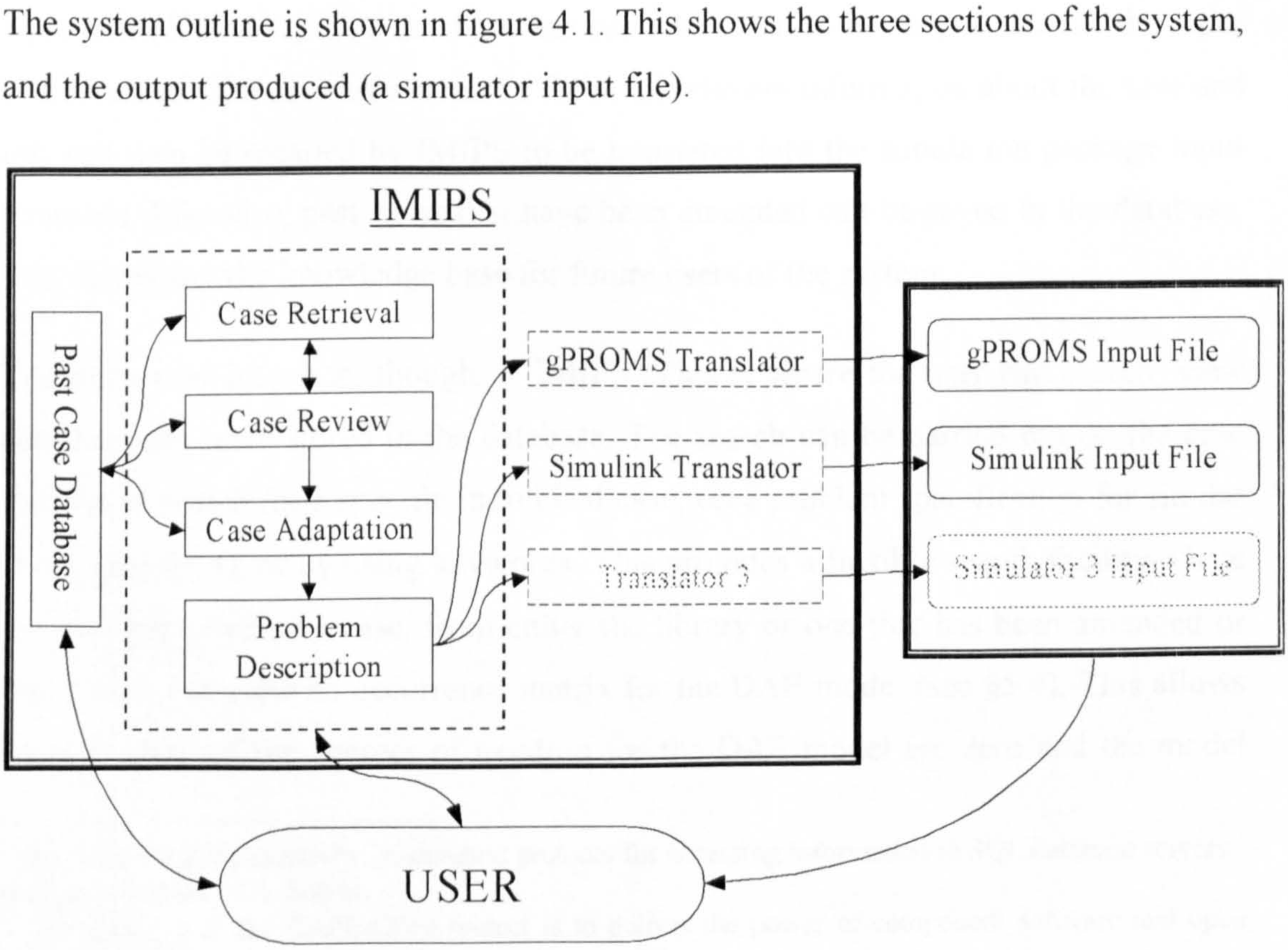


Figure 4.1. System diagram.

The sections are: the past case database, the retrieval and review mechanisms, and the translators, (these sections form IMIPS, Intelligent Modelling Interface for Process Simulators). IMIPS is implemented using the CLIPS language (http://web.ukonline.co.uk/julian.smart/wxclips/). CLIPS is NASA's expert system shell, allowing rule-based, functional and object-oriented programming. Our package uses wxCLIPS, which includes a set of CLIPS functions for graphical user interface creation. The database is constructed in Access (Microsoft, 1997), but could be written in any ODBC[3] compatible format with the same database structure. This structure could be altered in the future if applicable standards were introduced, and work on this standardisation is being done by the CAPE-Open[4] project. The mechanism used to create the search queries (Illiffe *et. al*, 1998) incorporates the ability to search for exact matches, and also matching for the parents and/or children of the selected item. It also allows searches for related objects.

The past case database is a store of previous cases through which the user can search for cases similar to the problem they are interested in. The user could also input a new case from scratch if they knew the relevant information to create a model in terms of its differential algebraic equations (DAE's), or integral and partial differential equations (IPDAE's). The database has been designed to be easy to use for the input of new cases. The user uses forms to fill in all relevant information about the case and this can then be recalled by IMIPS to be translated into the simulation package input language. Likewise, past cases that have been amended can be saved in the database, thus increasing the knowledge base for future users of the system.

The main user interface, though, is IMIPS. This is where the user can search, view and translate cases stored in the database. The search can be carried out on the case number to search for a specific individual case, on a problem specification for similar cases, (see §4.4), or by using keywords. This provides a flexible search facility. Once the user has selected a case, from either the library or one that has been amended or input, they can view an occurrence matrix for the DAE model (see §5.4). This allows them to check if the degrees of freedom for the DAE model are zero and the model

---

[3] Open Database Connectivity. A standard protocol for accessing information in SQL database servers, such as Microsoft SQL Server.

[4] The objective of the CAPE-Open project is to deliver the power of component software and open standard interfaces in computer-aided process engineering.

can be solved, i.e. the number of equations is equal to the number of the unknown variables. The user can then select the option to automatically create an input file (or files) suitable for use with their chosen simulator. Two translators have been implemented so far: gPROMS (PSE, 2000 a & b) and Simulink (MathWorks, 1999). Each part of the system is discussed in the following sections.

4.3 Case Specification

When stating a case in the past case database the user must be able to give a high level specification of the problem, in a general form. By stating the cases in a general form it allows IMIPS to search through these case specifications and retrieve all cases that are similar to the search conditions. Each specification in the database is referred to as a 'case'.

In the database, each case is represented in four parts: general information, equations, variables, and constants. A case contains all the information necessary to create a DAE model that can then be solved. IMIPS allows some checking of the case with an occurrence matrix (see §5.4). Each part of a case is described in the following subsections.

The index scheme is based on a hierarchical approach to representing data, thus allowing parent/child and other more specific relationships to be included.

Figure 4.2 shows part of the heating and cooling branch of the equipment hierarchy and is used in the indexing and searching of cases. Although no attributes are attached to any part of the hierarchy, the search engine can be instructed to search for exact matches, or include the children of, or parent of the item searched for. A search for *heating equipment*, for example, would retrieve all cases where the equipment was included in *heating equipment*, but by instructing the search engine to include the children would retrieve all cases where the equipment was a *heating equipment*, *drier*, *heater*, or *concentrator*. Likewise if the search was carried out for *heating equipment* and parent was included then all items from *Heating and Cooling Equipment* down would be included. The more general relationships, like those that would not fit into the hierarchical representation, include word associations, e.g. *cold* and *refrigerated*, and also where an item of equipment is of one type, but the type is not necessarily that

item, e.g. a pump may be an agitator, but an agitator is not a pump. Thus, a search for *cold* would also search for *refrigerated*.

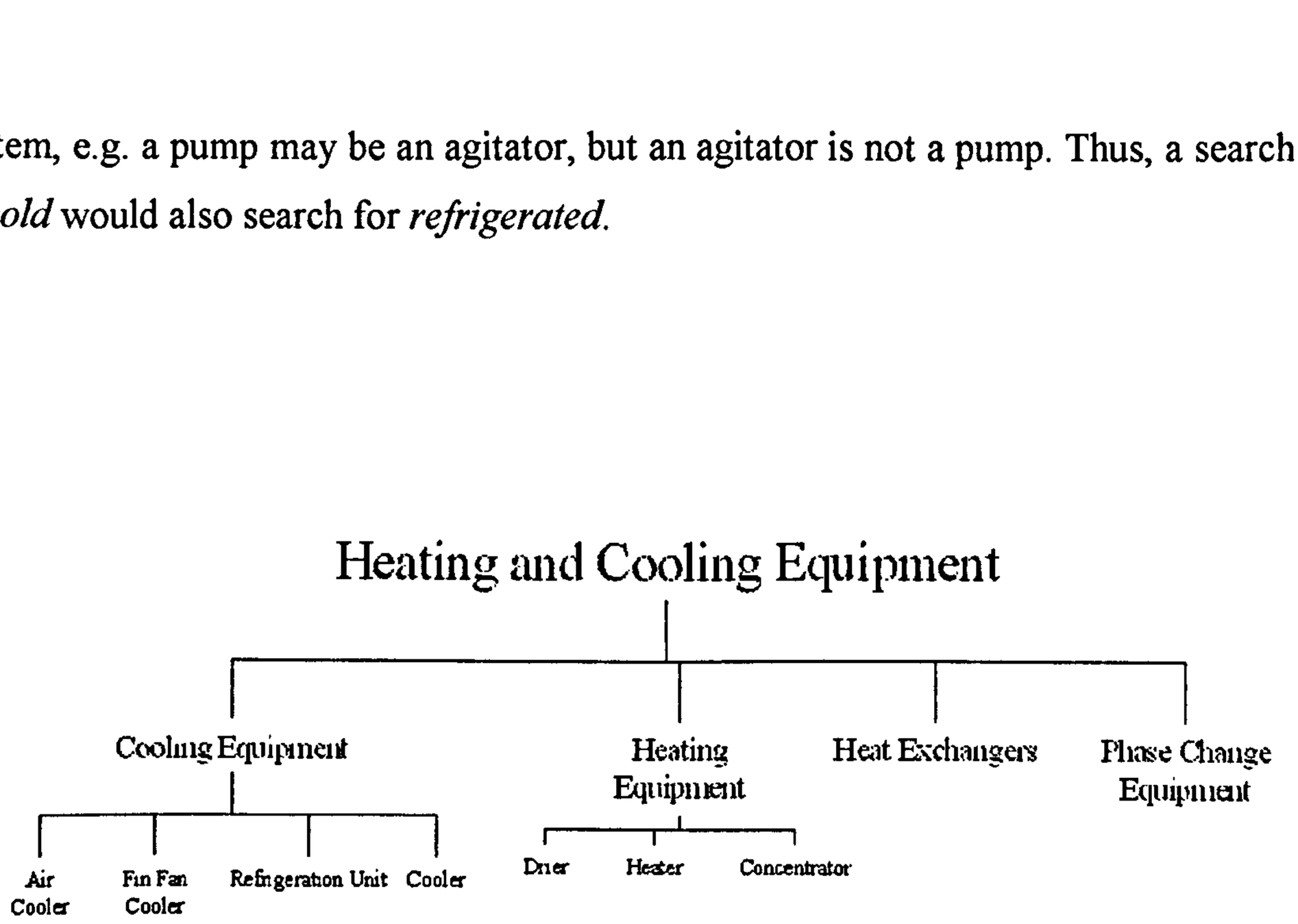


Figure 4.2. Part of the Heating and Cooling Equipment Hierarchy.

4.3.1 General Information

This part contains the information describing a problem and it is this information that is initially searched by IMIPS to find a matching case for the given problem. There are several fields to this part. (See figure 4.3.) Each field has a description box so that any notes can be included. These could be general notes or important assumptions about the problem. The descriptions aid in the completeness of the problem statement, which help other users to see why the problem was modelled that way. There is also a large field to input the overall problem statement.

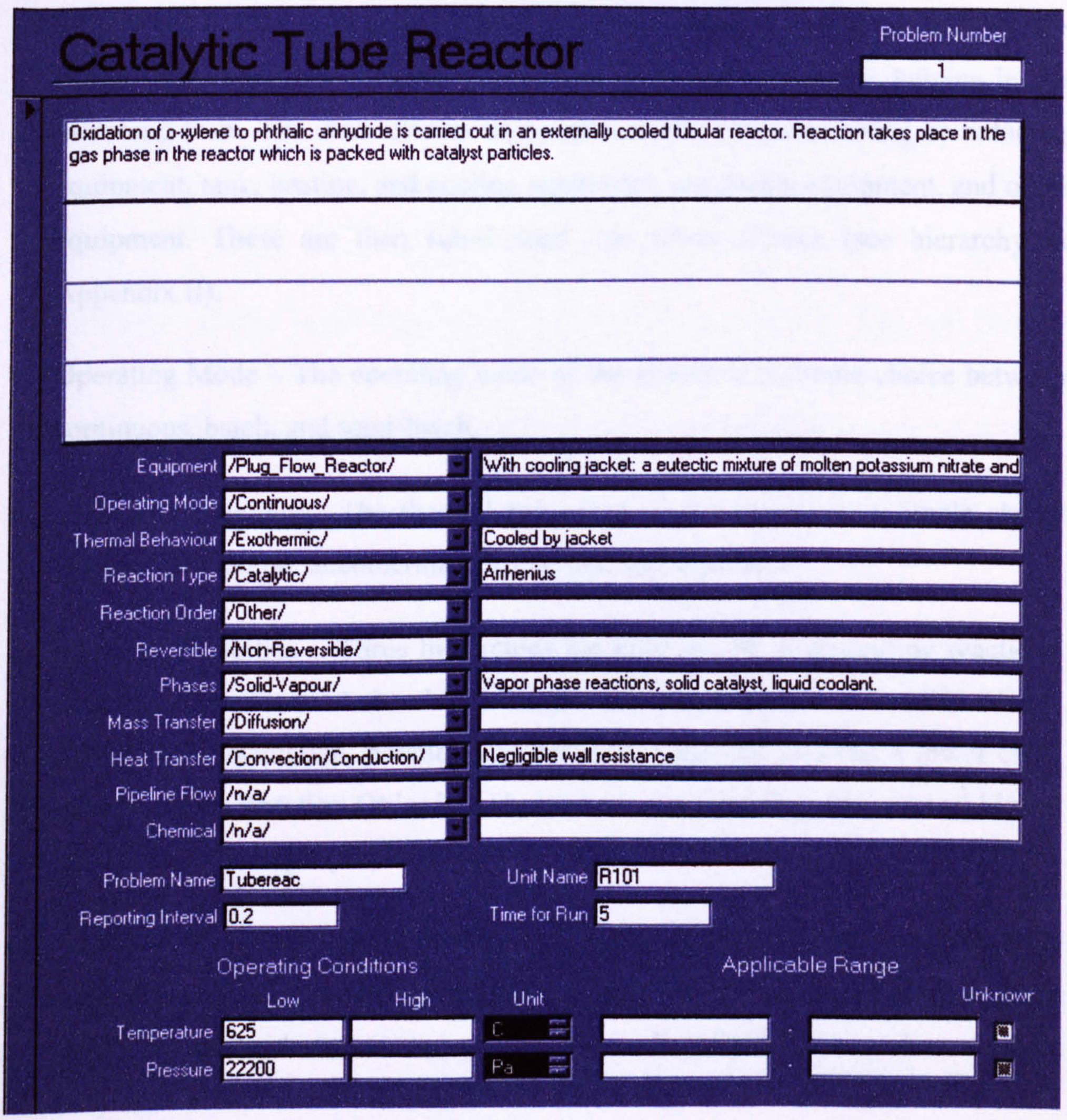


Figure 4.3. Database general information form.

Not all fields are applicable to a given problem, and so an input of n/a is valid. However, the more information that is given the better the problem is described for the search engine.

To help filling in the various fields, choices of options are provided. These choices are either in a list form or a classification hierarchy. All lists and hierarchies can be added to and improved upon as more problems are stored in the database. This allows for the continual adaptation of the system as more cases are stored and different index elements are needed. These fields are:

- Equipment – The equipment hierarchy is one of the largest hierarchies in the system. Items of equipment are grouped into like types, with this helping in the searching later. The main classes of equipment are: pressure raising or reducing equipment, tank, heating, and cooling equipment, separation equipment, and other equipment. These are then sub-divided into lower classes (see hierarchy in Appendix II).

- Operating Mode – The operating mode of the system is a simple choice between continuous, batch, and semi-batch.

- Thermal Behaviour – The thermal behaviour of the system is a simple choice between isothermal, endothermal, exothermal and superheat.

- Reaction Properties – Three hierarchies are only of use in describing reactions. Reaction Type is a choice between general, polymerisation, Van de Vusse, anaerobic, fermentation, parallel, and catalytic. Catalytic also has a lower class, auto-catalytic. Reaction Order is a choice between zero, first, second and higher. Reversible is a simple yes, no choice.

- Phases – The phases present in the problem is a choice between vapor, liquid or solid. If the problem were, for example, a jacketed reactor, then the main phase would be that inside the reactor, and not the cooling fluid. This can be a multiple entry choice.

- Mass and Heat Transfer – The mass transfer hierarchy is, at present, very limited as there are no mass transfer problems included in the database. The three main types of heat transfer listed are conduction, convection, and radiation.

- Pipeline Flow – When modelling flow down a pipeline the flow regime consideration is very important. The three main regimes listed are laminar, turbulent and plug/slug.

- Chemical – This section has the potential to be highly complex. Incorporating all chemicals is a virtually impossible task. The form chosen is a list structure, although relationships can still be stated, i.e. a chemical that has a widely used common name can be listed twice and linked together.

There are six other pieces of information that are included on the General Information page. A name is stated for the case, and this is used as the default name of the input file created by IMIPS. The correct file name extension is added by IMIPS on translation (e.g. *filename*.gPROMS for a gPROMS input file and *filename*.mdl for a Simulink input file).

The unit name is used in the input file to give the unit an identifier. This will become more useful in multiple unit systems, as, for example, different reactors will need different identifiers, but will be based on the same type of unit.

Two fields specify the simulation time (or time for run), and the interval to report the results.

The final two fields are for the statement of the operating conditions of the case (both temperature and pressure) and the applicable ranges for the case (again, both temperature and pressure). The operating conditions are for additional information, but the applicable range is more important. These ranges are used so that if part of the case is only valid over a certain range it can be stated. If the field is left blank then it is assumed that the case is valid over the total temperature and pressure range. The 'unknown' box can be checked if the user is unsure if the case is universally applicable, and these cases will be ignored in a search that includes a temperature or pressure range.

4.3.2 Equations

In the equations part the user states all equations for the solution of the problem. It is also where all information connected with the equations is stated, i.e. boundary conditions, limits, etc. (See figure 4.4.) The different sections are described in more detail below.

1	Equations	Fixed Bound	Lower Bound	Upper Bound	Description
	-BedVoid*Dz*PARTIAL(C(NoComp,ax,rad):ax) = u*(Cfeed(NoComp)-C(No	ax=0			Boundary condition at reactor entrance (z=0)
	-kz*PARTIAL(T(ax,rad):ax) = rhof*Cpf*u*(Tfeed-T(ax,rad))	ax=0			Boundary condition at reactor entrance (z=0)
	PARTIAL(C(NoComp,ax,rad):ax) = 0	ax=L			Boundary condition at reactor exit (z=L)
	PARTIAL(T(ax,rad):ax) = 0	ax=L			Boundary condition at reactor exit (z=L)
	PARTIAL(C(NoComp,ax,rad):rad) = 0	rad=0	ax < L	0 < ax	Boundary condition at reactor centre (r=0)
	PARTIAL(T(ax,rad):rad) = 0	rad=0	ax<L	0<ax	Boundary condition at reactor centre (r=0)
	PARTIAL(C(NoComp,ax,rad):rad) = 0	rad=R	ax<L	0<ax	Boundary condition at reactor perimeter (r=R)
	-kr*PARTIAL(T(ax,rad):rad) = hw*(T(ax,rad)-Tc)	rad=R	ax<L	0<ax	Boundary condition at reactor perimeter (r=R)
	$C(NoComp,ax,rad) = -u*PARTIAL(C(NoComp,ax,rad):ax) +BedVoid*Dz*f		ax<L;rad<R	0<ax,0<rad	Component mass balance
	rhof*Cpf*$T(ax,rad) = -rhof*Cpf*u*PARTIAL(T(ax,rad):ax) +kz*PARTIAL(T		ax<L;rad<R	0<ax,0<rad	Energy balance
	P(NoComp,ax,rad) = C(NoComp,ax,rad)*Rg*T(ax,rad)				Ideal gas law
	k(ax,rad) = A*EXP(-E/Rg/T(ax,rad))				Reaction constant
	Rate(ax,rad) = k(ax,rad)*P(1,ax,rad)*P(2,ax,rad)				Reaction rate
	rhoc*Vc*Cpc*$Tc = Fc*Cpc*(Tcin-Tc) + Q				Coolant energy balance
	Q = OverallU*Area*INTEGRAL(ax=0,L,T(ax,R)-Tc)				Heat transfer relationship
	Pfeed(NoComp) = Cfeed(NoComp)*Rg*Tfeed				Feed conditions
	END				

Figure 4.4. Database equations form.

Equations are written very much as they would appear on paper. There is limited syntax to understand (see Appendix I). The only limitation is that each equation must be less than 255 characters long. This is due to a limitation of the database, Access (Microsoft, 1997), but can be easily overcome by splitting an equation into parts and using dummy variables. Array variables and constants can be included in the equations and are explained in §4.3.3 and §4.3.4.

4.3.2.1.1 Boundary Conditions

If an equation is only valid at one place in the system, it is stated in the fixed bound section as:

$$Identifier = Bound$$

Identifier is a variable present in the equation, and *Bound* is a value or constant where the equation is fixed.

The lower boundary for the validity of an equation can be stated as:

$$Identifier1 < Bound1 \; ; \; Identifier2 < Bound2$$

There can be as many boundaries as necessary, the semi-colon is used as a separator.

The upper boundary for the validity of an equation can also be stated.

$$Bound1 < Identifier1 \; ; \; Bound2 < Identifier2$$

As for the lower bound, there can be as many boundaries as necessary, separated by a semi-colon.

A description of the equation can be added alongside the equation. This can include what the equation is for, and any assumptions that were made in the choice and formulation of the equation.

4.3.3 Variables

In this part, all variables are declared along with their properties. (See figure 4.5.) Variables can be fixed and used as constants, however, constants cannot be used as variables.

1 Variable	Initial	Set	Units	Lower Bound	Upper Bound	Distribution	Solution Method	Description
ax			m			0:L	BFDM, 2, 40	Axial position
rad			m			0:R	OCFEM, 3, 5	Radial position
C(NoComp,ax,rad)	0		kg/mol	0 < ax ; 0 < rad	ax < L ; rad < R			Concentration
P(NoComp,ax,rad)			Pa					Partial pressures
T(ax,rad)	625		K	0 < ax ; 0 < rad	ax < L ; rad < R			Temperature
k(ax,rad)			mol/kg.s.Pa2					Reaction constant
Rate(ax,rad)			mol/kg.s					Reaction rate
Cfeed(NoComp)			mol/m3					Feed concentration
Pfeed(NoComp)		1100,21100	Pa					Feed partial pressures
Tfeed		625	K					Feed temperature
u		0.877	m/s					Superficial gas velocity
Fc		0.1	kg/s					Coolant flowrate
Tc	625		K					Coolant temperature
Tcin		625	K					Coolant temperature in
Q			W					Total heat load absorbed by coolant

Figure 4.5. Database variables form.

Arrays can also be used to simplify some equations. There can be arrays of variables and arrays of constants. An array has a name and a list of identifiers for the array elements. An array can have unlimited dimensions, so arbitrarily large arrays can be written quite easily.

Variable_Name(Array_Var_1,Array_Var_2,……,Array_Var_n)

with the array element identifiers, *Array_Var_n*, being declared in the constants part.

There are four basic properties than can be filled in for each variable. These are its name, its initial value or set of values, the units and a description. The initial value or set of values can be written as a single value or as an array. Even if the variable is an array it can still have a single value and this will be translated into an array of elements, all of the same value for the simulator. Array values are written with the

individual values separated by commas, e.g. 12,23,45,23. There are four other properties that the variables may have and these are described in more detail below.

If a variable is distributed then the range for the distribution needs to be stated. (Distributed variables are variables whose properties vary, not only with time, but also with position, for example the temperature in a tubular reactor varies with its axial and radial positions within the reactor.) The distribution domain is written as: *0:L*. Here the lower limit for the domain is zero and the upper limit is a constant, *L*. The limits can be either values or constant identifiers.

If a variable is only valid between certain points then these should be stated here. The syntax for this section is the same as that in the equation part.

For problems that use distributed variables, the solution method should be stated. The methods discussed here are based on the gPROMS nomenclature and ideas (PSE, 1999b, p.70), as gPROMS is the only simulator being considered in the thesis that can cope easily with distributed systems. gPROMS provides five numerical methods, which are shown in table 4.1.

A chosen method is indicated by stating the *Method*, *Order*, and *Interval* in exactly this order. For example an entry of BFDM, 2, 40 means use the backward finite difference method, order 2, over a uniform grid of forty intervals is to be used.

Numerical Method	**IMIPS**	**Order**	**Partial Derivatives**	**Integrals**
Centred finite difference method	CFDM	2,4,6	✓	✓
Backward finite difference method	BFDM	1,2	✓	✓
Forward finite difference method	FFDM	1,2	✓	✓
Orthogonal collocation on finite elements method	OCFEM	2,3,4	✓	✓
Gaussian quadratures		5		✓

Table 4.1. Available mathematical methods

4.3.4 Constants

A constant has a fixed value that can be either a single value or an array. (See figure 4.6.)

1	Constant	Value	Units	Inc. ?	Description
	L	3	m	Y	Reactor Length
	R	0.0127	m	Y	Reactor Radius
	rhob	1300	kg/m3	Y	Bed Density
	rhof	1.293	kg/m3	Y	Fluid Density
	Cpf	992	J/kg.K	Y	Fluid Specific Heat Capacity
	Dz	0.01	m2/s	Y	Axial Diffusivity
	Dr	0.0001	m2/s	Y	Radial Diffusivity
	kz	0.5	W/m.K	Y	Axial Conductivity
	kr	0.05	W/m.K	Y	Radial Conductivity
	BedVoid	0.35		Y	Bed Voidage Fraction
	deltaH	-1.20E+06	J/mol	Y	Reaction Enthalpy
	A	1.15E-02	mol/kg.s.Pa2	Y	Pre-Exponential Arrhenius Constant
	E	113370	J/mol	Y	Activation Energy
	hw	96	W/m2.K	Y	Wall Heat Transfer Coefficient
	Rg	8.314	J/mol.K	Y	Ideal Gas Coefficient
	rhoc	2000	kg/m3	Y	Coolant Density
	Cpc	123900	J/kg.K	Y	Coolant Specific Heat Capacity
	Vc	4.56E-03	m3	Y	Cooling Jacket Volume
	Area	0.239	m2	Y	Overall Heat Transfer Area
	OverallU	9.6	W/m2.K	Y	Overall Heat Transfer Coefficient
	NoComp	2		N	Number of Components Present
	Nu(NoComp)	-1,-1		Y	
				Y	

Figure 4.6. Database constants form.

Constant arrays are written in the same syntax as for variables.

Constant_Name(Array_Var_1,Array_Var_2,……,Array_Var_n)

With the array element identifiers, *Array_Var_n*, being declared elsewhere in the list of constants, e.g. see figure 4.6, *NoComp* is an array element identifier for the constant *Nu*.

Five properties can be set for each constant: name, value, units, description, and 'Include'. The first four are obvious. The 'Include' property value can be either yes (Y) or no (N). This is to distinguish between a constant and an array identifier. An array identifier can be used to indicate which component, for example, the variable or constant is talking about, e.g. *P(NoComp)* for pressure of each component. *P(1)* is the pressure of component 1, *P(2)* the pressure of component 2, etc. With *NoComp* being the array identifier with a value for the number of components present. This

completes the description of the past case database and how to specify a case. The next sections discuss how the case is retrieved and then reviewed.

4.4 Case Retrieval

Facilities are provided for the user to search through the library of past cases. By using the hierarchies defined, a search can be made more intelligent by considering similar or related cases (discussed in §4.4.2). The following subsections describe the different aspects of case retrieval, namely: the search query specification, and the search query mechanism.

4.4.1 Search Query Specification

A search query is input through a form in IMIPS that is very similar in layout to the general information form for the database (see figure 4.7.). The query is then used by the database as the basis from which to search through the cases. Each section has a hierarchical list that the user can use to select various key words to search from. These are menu representations of the hierarchies shown in Appendix II. There is the opportunity for the user to input related terms to be searched too. The user can tell the search engine to look for the parents or children of the selected key words, or can disable the relationship search mechanism. This makes the search more intelligent so results that are more general can be retrieved to give the user a wider scope of examples to view. For example, a simple search for *Tank* in the equipment box would retrieve all cases stored with *Tank* as the equipment type. A search with the 'children' tab marked would also retrieve cases with equipment types *Jacketed Vessel* and *Reactor Vessel*, and *Plug Flow Reactor* and *Continuous Stirred Tank Reactor* which are the children of *Reactor Vessel*. If the search were for *Reactor Vessel* with the 'parent' tab marked then the retrieved cases would have an equipment type of either *Reactor Vessel* or *Tank* (see Figure II.3).

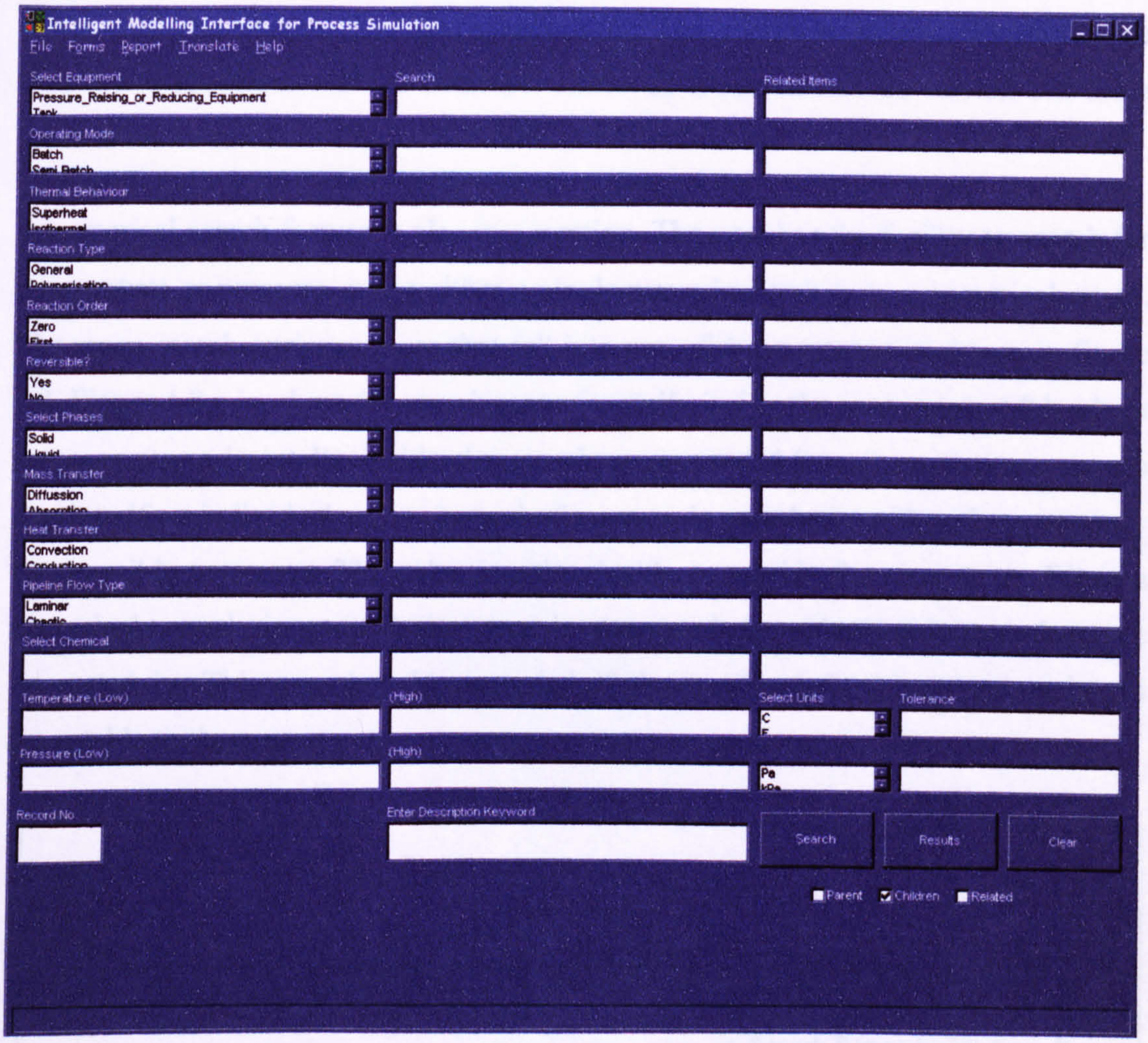


Figure 4.7. IMIPS query specification form.

4.4.2 Search Query Mechanism

The search query mechanism allows a search to be written for complete matches, and for related units (Illiffe *et. al*, 1998). The user can also search through (either up or down) the hierarchy that the unit is in for similar cases. The mechanism uses the stated hierarchies and relationships to enlarge the search specification. By allowing the user to decide to search through wider fields, the user has a higher chance of retrieving a case that is likely to be of use to them.

If the user leaves an item empty the search mechanism will assume the user has no preference for that menu. The user can choose whether to do an exact match search, or to search through the children or parents of the items chosen. The user may also search for a related item by inputting it in the related text box on the right.

A keyword may also be input and searched for. This search takes place in each of the description text boxes in the database. Likewise a search can be carried out for a specific case number.

A numerical search feature is also in operation. The user has the facility to search for temperature or pressure, using either a single numerical value, or a numerical range. The range search retrieves cases that fall into one of three categories shown in figure 4.8. Figure 4.8.a is where the search range is totally inside the case range, 4.8.b where the case range is totally inside the search range, and 4.8.c where the two ranges overlap. If an individual value, instead of a range, is searched for then there are only two possible outcomes. The value is either inside or outside the case range. When a numerical search is created the user is prompted for which category they are interested in. This can then be amended if the number of retrieved cases is not acceptable to the user.

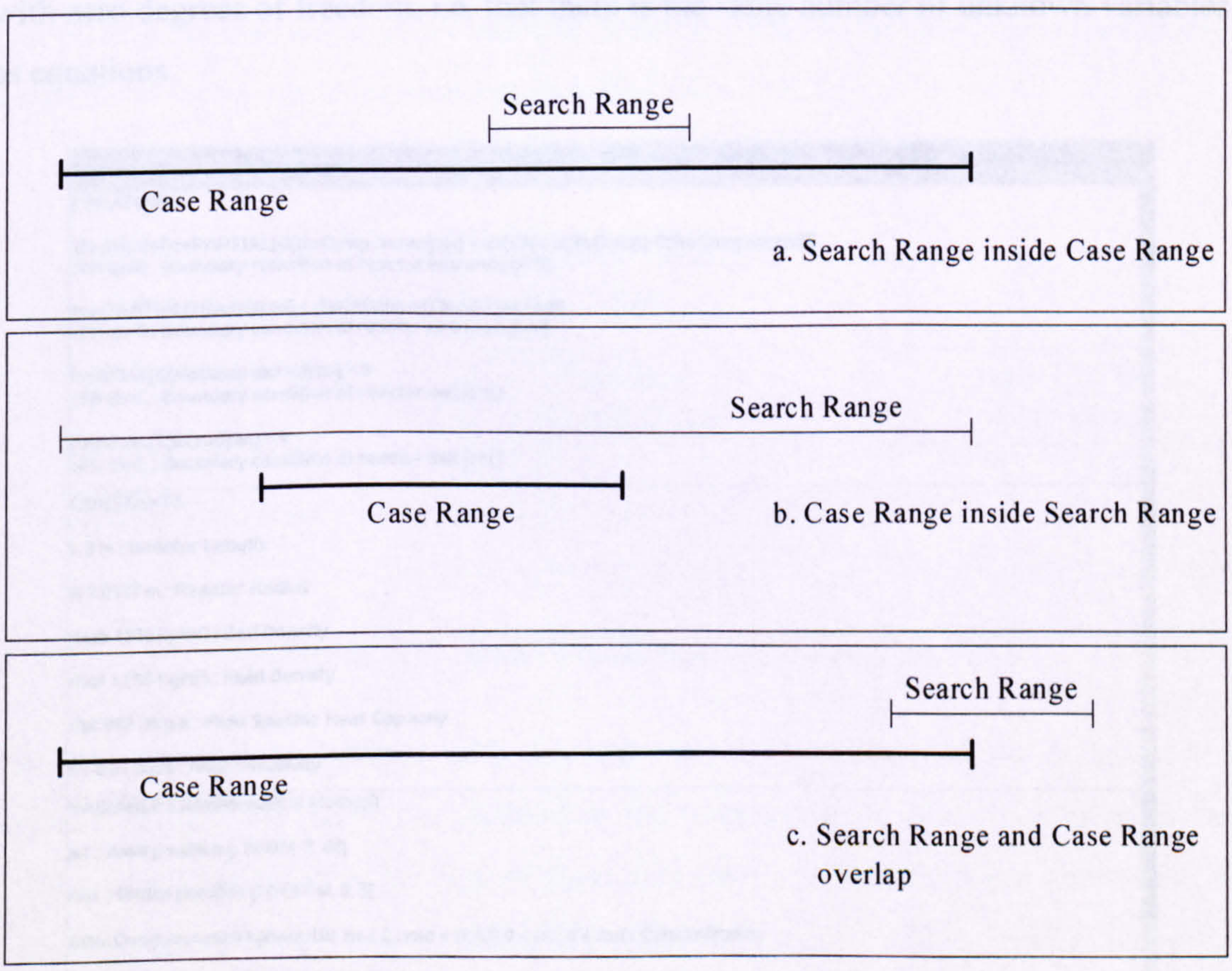


Figure 4.8. Numerical Search Retrieval Possibilities.

Once the search has been completed and the cases are retrieved from the database, the user is able to review them. The user needs to then decide which, if any, are suitable for describing the process they wish to represent.

4.5 Case Review

Once the search is completed, the review screen shows the first retrieved case from the database. The retrieved cases can be scrolled through so the user can decide on their appropriateness. Initially the general information about the retrieved cases is shown. Each retrieved case can then be looked at in more detail by clicking the 'more info' button. This brings up a screen (figure 4.9) with the variables, constants, and equations with their properties. Showing the retrieved cases in this way saves time as the user will not have to search through hundreds of equations and problems to find the suitable cases. The user can also see an occurrence matrix for the selected case (see §5.4). This is useful when checking that the case model has been stated correctly with zero degrees of freedom, i.e. that there is the same number of unknown variables as equations.

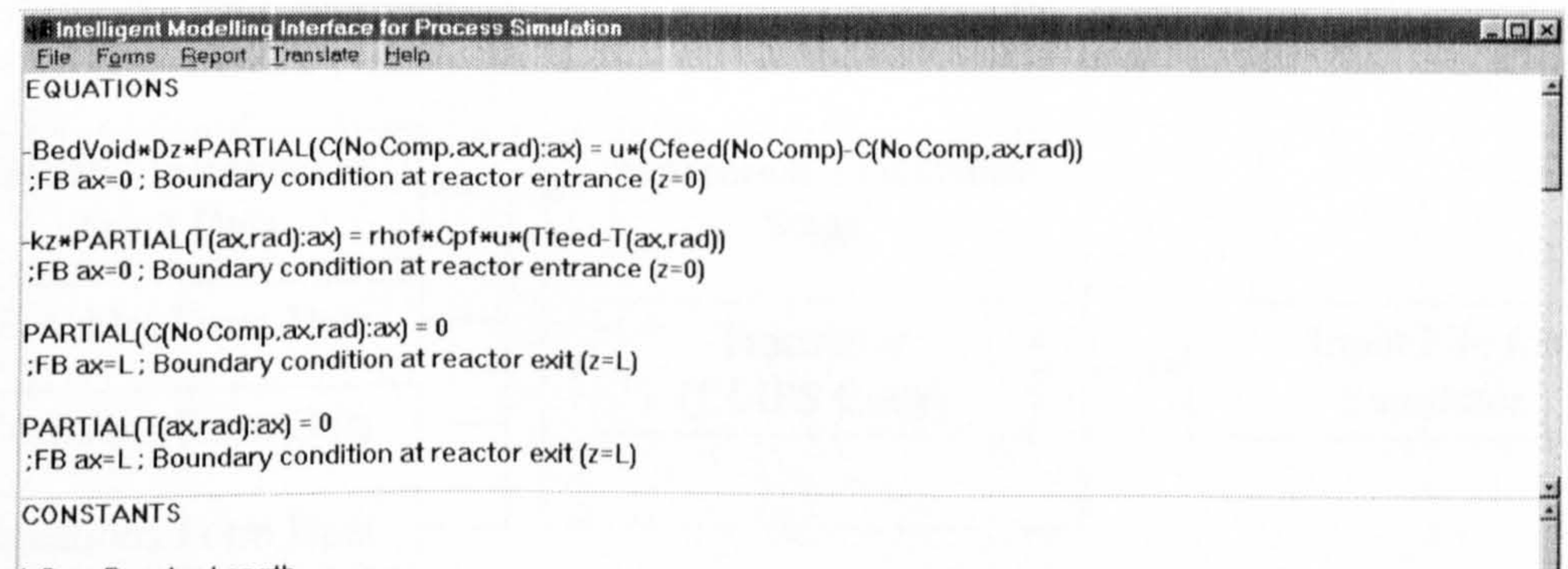


Figure 4.9. More Info Screen

4.6 Case Adaptation

Once the retrieved cases have been reviewed the user can either adapt one of the retrieved cases or input a new case with the specification of the problem of interest. The adaptation of previous cases is completed through the database interface, where the similar case can be copied and amended accordingly.

This action is carried out, at present, by the user without any support from IMIPS. In the future this feature is to be included in IMIPS.

4.7 Case Translation

Once the user has retrieved the case that they wish to use, they can decide which simulator is most appropriate for their particular problem. With the click of a button, the automatic translation of the selected case from the database begins. The translators convert the case retrieved into an input file to be used with the selected simulator (see figure 4.10). The first step that is consistent for all translators is the initialisation step. During this the case information that was stored in fields in the database is turned into instances of equations, constants and variables (See §5.6.1.)

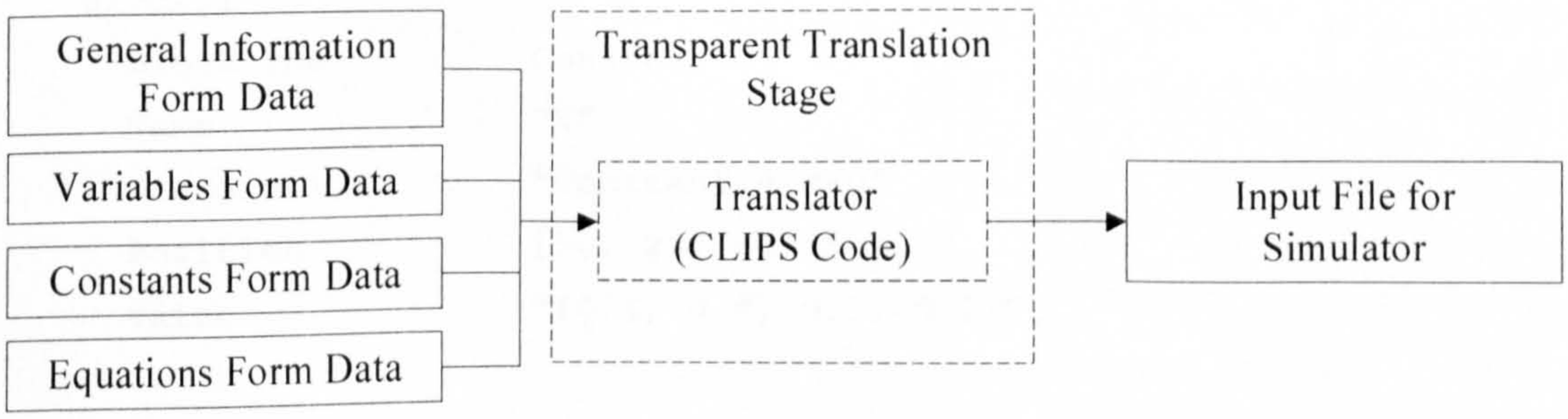


Figure 4.10. Basic Translation Flow Diagram.

The translation is transparent with the user only being prompted for an input file name, although if a name was defined in the database then this will appear as the default, and, if applicable, the type of numerical search to be used. Once translation is complete, the user may create another input file for a different simulator by the same method. Thus allowing them to use the same mathematical case specification to create different simulator packages' input files for a particular problem, and then to compare the different results gained.

At present, there are two translators available for the simulation packages Simulink and gPROMS. How these translators have been created is now described in the following subsections.

4.7.1 Simulink

Simulink has many lines of default code, which are automatically created by the translator. The information for some of these lines comes from the case selected, e.g. name, stop time, and current time. These lines include the general properties of the blocks and links in the model, and the page that the model is shown on. These lines of code are similar for each simulation run through Simulink. The other issues the translator has to consider are now reviewed.

4.7.1.1 Block Selection

Initially the constants are declared and a block is created for each constant. If a constant is an array, this is stated by enclosing the array values with square brackets before stating it as the block value. For example: X (Constant Array) = 0.1, 0.2, 0.5, 0.2. Then the translator would create the following code:

```
Block {
  BlockType          Constant
  Name               "X"
  Description        "Constant Array"
  Position           [50, 20, 80, 50]
  Value              "[0.1, 0.2, 0.5, 0.1]"
}
```

The constant blocks are created one after each other and are placed on the page with set gaps between them. Once all the constants have been displayed, the equation decomposition starts.

Each equation is then systematically decomposed, based around the constants and the known mathematical, or logical, syntax. As the equations are decomposed, a block is drawn for each symbol or logical operator that is present. As there is only one block for each constant, before any linking is done, the blocks that the constant is attached to are stored in a multi-variable array in the instance of that constant so that the links can be created effectively.

4.7.1.2 Block Linking

The block linking is the most difficult part of the translation. The links from each block have to be written together, so cannot be written until all the equations have been decomposed and all block code is present. To enable this the use of multi-variable arrays is needed to describe the links from each constant. As each symbol block code is written, the link to it is either drawn (if a link from another symbol block) or stored in the instance of the source constant.

4.7.1.3 Results Viewing

The user is asked during translation to select, from the list of variables, those they are interested in viewing results for, and the type of display they want; a numerical display (a numerical screen) or a scope display (a graph plot). This allows the inclusion of displays in the Simulink model to show the results. Depending on the type of simulation, the correct display can be selected. (For a steady state model, a numerical display would suffice, and for a continuous model, a scope display would be preferable.)

4.7.1.4 Equation Handling

Each equation is automatically checked, first for brackets, then for mathematical symbols. As the checking proceeds, if a constant is part of the equation, the symbol it is linked to is recorded in a multi-variable array attached to that constant.

At present partial differential equations cannot be dealt with easily in Simulink, but differential equations can. These are discussed in the following subsections.

4.7.1.4.1 Differentials

Simulink represents differentials in the following way. If the equation includes a differential then the simulation must include an integral. So, if the equation were $\frac{dx}{dt} = -\frac{K(x-b)}{(M \cdot Vol)}$ then the Simulink input would be shown as in figure 4.11. Simulink has to solve the differential equation by integration and so an integrator (represented as $\frac{1}{s}$) is used instead of a derivative.

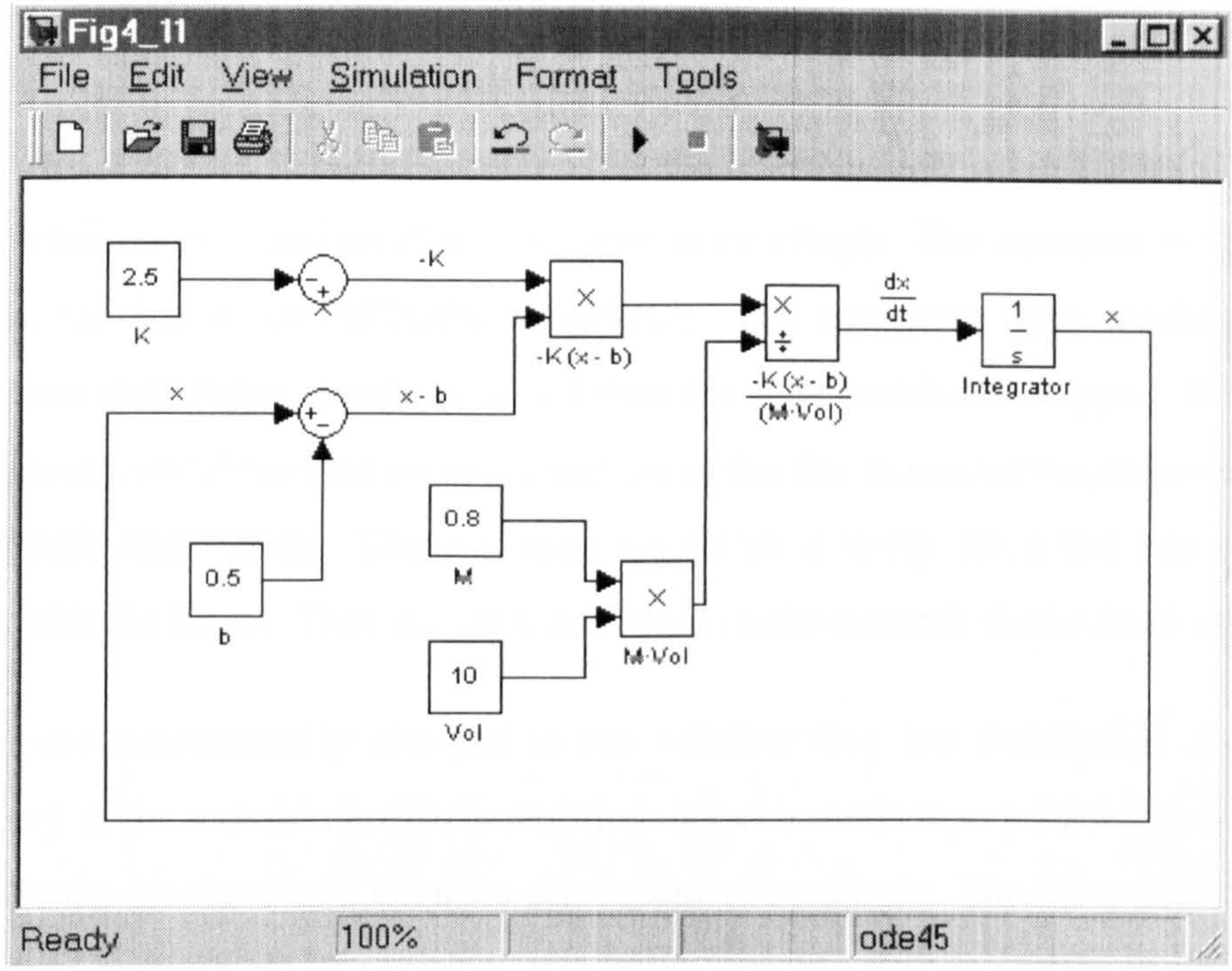


Figure 4.11. Differential equation, as seen in Simulink.

As the translator looks through the equations if a differential is found then the integration block is drawn. Once the integration block has been drawn the translator searches to see if the variable to be integrated had an initial condition. If so, this is then included in the block code.

4.7.1.4.2 Integral Partial Differential Equations

At present, Simulink does not have the ability to solve IPDAE's. Hence, if the user tries to translate a case including an IPDAE into Simulink, there is an error returned to show this facility is not available.

This completes the discussion of the translator for Simulink. Next, the translator for gPROMS is described.

4.7.2 gPROMS

First, the variables are DECLAREd and given a TYPE. Each TYPE has an initial value, upper and lower bounds, and the units associated with the variable for which it was created.

4.7.2.1 MODEL

The model name is recalled and stated and the constants are listed in the PARAMETER section. Each constant is automatically checked to see if they are an array (described in more detail below), and whether the value is an integer. The constant is then listed as a REAL number or an INTEGER accordingly. The translator then checks to see if there are any distributed variables. If not then the next section is skipped. If there are any distributed variables then an extra section in the file is needed to allow them to be declared with their limits. These values are taken directly from the instances and declared with the limits. They are then not listed in the variable declaration section.

Variables are automatically checked to see whether they are distributed or an array (see below). If the variable is distributed then it has bounds over which it is valid. The program looks for these and changes the output accordingly. The information written is dependent on the type of variable, if a basic variable then its TYPE is recalled and stated directly, otherwise the format appropriate to the form of variable is written.

If an array is detected (or a distributed variable) then the format of the output differs from retrieving and simply rewriting the variable/constant name. The variable is originally written as *VarName(A,B,C,...)* and for the declaration in the input file the variable name is separated from the remaining, bracketed, part. The bracketed part is then written later in the line, e.g. A variable: *Conc(NoComp)* is the concentration of component *NoComp*. This has been declared in the database with an initial value, units, and a description. The initial value is used in the PROCESS section of the input file (described below) and the units are used in the TYPE section described above. This information would appear in the gPROMS input file as:

```
Type1  =  1  : -1E6 : 1E6  UNIT = "kg/m3" # Concentration
Conc   AS ARRAY(NoComp) OF Type1 # Concentration
```

Once all the constants and variable have been declared the equations are dealt with. There are two sections used in gPROMS for the declaration of equations, BOUNDARY for the statement of boundary condition equations and EQUATION for the statement of the other equations. Equation handling is described in more detail in § 4.7.2.3.

4.7.2.2 PROCESS

After the declaration of all the equations the problem is still unfinished, as without the initial conditions and setting parameters the problem is only half complete. gPROMS includes all this information in a PROCESS section. The name for the unit was stored during the initialisation stage and is recalled to give the unit a unique identifier. This will be more important when modelling multiple units as each unit will need an original identifier. A SET section is automatically written in which the constants are listed with their values, and units if present. In addition, in this section, the mathematical methods chosen for solving the distributed domain are stated. In this section, and in the section setting the variable initial values, if the values were listed as an array, they are dealt with slightly differently. They are read and brackets are included in the input file statements. The variables can be either set or have an initial value attached to them. If they are set then the modelling package uses them as constants (as constants are just specialist forms of variables). The ASSIGN section sets variable values so they are treated like constants and the INITIAL section gives the variables initial values.

Finally, the length of time for the problem to run and the interval for results to be plotted is stated. These values are read from the database, but if none are present, default values set within the database are used.

4.7.2.3 Equation Handling

The translation of equations is the most troublesome, with three main problems being identified and their solutions are detailed below.

4.7.2.3.1 Arrays and Loops

An array (usually of variables or constants) needs to be identified within an equation, and then a loop is created for it so that an equation is written (internally) for each instance of each variable.

The translator identifies the equations that include arrays after searching for partial differentials. The variables with brackets after them are identified and using a specially written function each array variable (within the brackets) is replaced and a FOR loop (see below) inserted before the equation. Every time the equation changes,

it is re-checked for arrays, IF and WHILE constructs, and partial derivatives. In this way, the correct number of loops associated with array variables is ensured.

Loops are needed for arrays and are useful for repetitive tasks. They can be stated in the equation definition in the database, or are produced automatically by the translator. To create the correct expansion for an equation that includes arrays the translator searches the equation for an array variable. If it finds one it writes the first part of a FOR loop. It then changes all occurrences of the variable to the loop variable, and then searches again. This iterative step continues until the equation does not change and then the translator writes the END lines for the FOR loops, e.g. the equation as it is written in the database:

```
k(ax,rad)=A*EXP(-E/Rg/T(ax,rad))
```

This is then translated to:

```
FOR i1 := 0 TO L DO
  FOR i2 := 0 TO R DO
    k(i1,i2)=A*EXP(-E/Rg/T(i1,i2));
  END #For
END #For
```

By the searching for arrays, and then the replacement of the array variables, `ax` and `rad`, with the loop variables, `i1` and `i2`, the loops are created.

For boundary conditions where the limits are just inside some values, e.g. $0 < x < L$, and not to the boundary, e.g. $0 <= x <= L$ the translator can insert the correct symbol to the FOR loop definition. The values are input in the database and these are read by the translator who then adds the appropriate symbols. For example the equation:

`PARTIAL(C(NoComp,axial,radial):radial) = 0`

`radial = 0, axial < L, 0 < axial` (where `L` is the reactor length).

Would be translated to the gPROMS code:

```
# Boundary condition at reactor centre (r=0)
FOR i1 := 0|+ TO L|- DO
 PARTIAL(C(,i1,0),rad) = 0;
END #For
```

This equation also shows the use of a partial derivative. The way the translator deals with these is described in the next section.

4.7.2.3.2 Partial Differentials

Partial differentials are a special case inside a FOR loop. To convert the user input, e.g. `PARTIAL(C(NoComp,axial,radial):radial) = 0, at radial=0`, to a form gPROMS will use involves formation of FOR loops and manipulation of the original statement. For this example, the gPROMS code generated is:

```
FOR i1 := 0 TO L DO
 PARTIAL(C(,i1,0),radial) = 0;
END #For
```

The normal FOR loop function cannot be used for this purpose, as this would replace all occurrences of axial and radial with loop variables (`i1`, `i2`, etc.). Likewise, the fixed variable (`radial`) cannot be just replaced throughout, as this would not leave the final occurrence in the equation.

A function was written to solve this problem. It first searches each equation for the string `PARTIAL` and if this exists it triggers the function. The function then looks for the `:` and the close bracket after this. It removes whatever is in this range and then treats the equation as any other, replacing for fixed values (`radial`), removing non-included (array identifiers, see §4.3.4) constants (`NoComp`), and creating FOR loops accordingly. Then the removed string is replaced before the function is terminated.

4.7.2.3.3 Integrals

Integrals are dealt with in a similar way to partial differentials; the translator cannot just search the line and replace all instances of variables that may usually be used to from loops, e.g.

```
Q = OverallU*Area*INTEGRAL(ax:=0:L;T(ax,R)-Tc)
```

as `ax` would normally be used to create a loop it is replaced and the translation is:

```
Q = OverallU*Area*INTEGRAL(IntNo1:=0:L;T(IntNo1,R)-Tc);
```

The replacement is triggered with a search for `INTEGRAL`, with the term between the first bracket and the `:=` being replaced throughout the line with `IntNo1`, or similar (i.e. `IntNo2`, etc.). This way loops can still be formed if necessary with the integration variable not being affected.

4.7.2.3.4 IF and WHILE Constructs

IF and WHILE constructs are quite easy to deal with. The IF...THEN...ELSE statement is split into its three parts and the WHILE...DO statement into two, and then

each part is dealt with as an individual equation. This way nested IF or WHILE loops can be dealt with and the correct nesting is created for the output code.

The application of IMIPS to a general example is now described in the following section.

4.8 General Cases – From Definition to Translation

The aim of this section is to run through the basic stages of the procedure: the initial case statement in the database, the search creation, the review and adaptation, and the translation to an input file.

The user can use IMIPS for a variety of scenarios:

1. To create a case description from scratch in the database, and then use IMIPS to create simulator input files.

2. To use IMIPS to search and retrieve an existing case, accept this as representative of the user's process and carry out translation into simulator input file/s.

3. To use IMIPS to search and retrieve cases similar to that of the user's process. Then adapt one (or some) of the retrieved cases and then use this as the basis for the creation of simulator input file/s.

Each of these scenarios are now considered.

4.8.1 Scenario 1: A new case from scratch.

Step 1. Write down the mathematical description of the problem.

Step 2. Create the case description in the case database from the mathematical description in step 1.

Step 3. Use IMIPS to retrieve known case and initiate translation of case to simulator program input code.

4.8.2 Scenario 2: Retrieval and use of an existing case.

Step 1. Use IMIPS query form to automatically create search query.

Step 2. Use IMIPS to search and retrieve relevant cases.

Step 3. Select appropriate case and use IMIPS to translate into simulator program input code.

4.8.3 Scenario 3: Retrieval, Adaptation, and use of an existing case.

Step 1. Use IMIPS query form to automatically create search query.

Step 2. Use IMIPS to search and retrieve relevant cases.

Step 3. Select appropriate case and use database to create copy of selected case.

Step 4. In the database, adapt the copied case to represent the user's process.

Step 5. In IMIPS select new, adapted case and translate into simulator program input code.

A more detailed description of these processes is given in Chapter 6 where case studies are presented.

4.9 Summary

This chapter has outlined the system under development as the major part of this project. The chapter has shown how IMIPS can be used to store case statements of problems in a database for use at a later date. IMIPS has been designed to allow the novice user to easily create a search and then gives them the ability to review and amend the retrieved cases with ease. The user then has the ability to create simulator input files at the touch of a button. These files can be run and the results compared.

How a case can be specified, retrieved, reviewed, adapted and translated was discussed. How the translators for Simulink and gPROMS were created was also described. And finally, the application of IMIPS in a variety of scenarios was described.

The next chapter outlines how the system has been designed and implemented.

5. System Design and Implementation

5.1 Introduction

This chapter highlights the design and implementation issues encountered and decisions made during the development of IMIPS. The layout of this chapter is as follows: The database is covered in §5.2, §5.3 looks at the retrieval mechanisms used, §5.4 covers the retrieved case review capabilities of the system, §5.5 looks at how to adapt the retrieved cases, and the translator design and implementation is shown in §5.6.

5.2 Past Case Database Implementation

The greatest problem in the design of a case database is that of designing a structure for a case fully so that others have sufficient information to understand it. But, at the same time, keeping the case small enough so that it is not difficult to understand or keep track.

The representation of a case in the database is a complex task. Ensuring the case is small enough whilst including all the relevant information has proved troublesome. The initial statement of the case was relatively straightforward once an indexing scheme had been decided upon. Many of the extra fields used were created once a need for more information had been found through the testing of the translation mechanisms. These extra fields contain information important to the complete statement of the case to ensure the correct translation and, hence, solution of the problem.

The indexing system used in the general information page of the case base is not exhaustive and can be amended as more cases and the need for improved matching are included.

5.3 Case Retrieval

The retrieval of the relevant cases is an area that uses some of the ideas from the case-based reasoning tools described in Chapter 3.

When retrieving the relevant cases the retrieval mechanisms used must ensure that all relevant cases are returned to the user. Likewise, the search engine should not retrieve those cases that are of little relevance to the user's search parameters. The second point is related more to the way the search specification is represented than to the search mechanisms used and so the first point is explored in more detail below.

The search query generation mechanisms used in IMIPS are based on those of Illiffe *et. al* (1998) and include the ability to improve the search by not only looking for identical matches (very unlikely), but also look for cases that are related to the search parameters. As was outlined in §3.2 there are many methods that can be used to improve on the basic match search procedure. By improving the way the cases are stored and indexed, relationships can be included in the indexing and, thus, the retrieval of the cases. By using a hierarchical structure, the user can search for the children (further down the tree) or parents (further up the tree) of their search parameter. This improves the matching abilities of the system.

The other form of search implemented in IMIPS is the ability to search for numerical matches of individual numbers or over numerical ranges. This is important as some cases may include equations or boundary conditions that are only valid over certain temperature or pressure ranges. The ability to remove cases where the model is invalid for the user's requirements is, therefore, important for the retrieval of the correct cases. When searching for numerical values there are two outcomes possible: the value is either inside or outside the range, but when searching for numerical values, as was stated in §4.4.2, there are four possible outcomes: the search range is completely inside the object range, the object range is completely inside the search range, the two ranges overlap or the two ranges do not meet. The user can select which range search to use (or choose all) and the search query generator will produce the correct SQL query. The type of range search that is of most use to the user depends on their own search criteria, this allows the user to eliminate cases that are only valid over temperature or pressure ranges outside those of interest.

5.4 Case Review

Once the search is complete, the user needs to have the ability to review the retrieved cases so they may select those that are suitable for their problem. Initially, only the general information (§4.3.1.) of the retrieved cases is shown using the same format as the database. From this initial selection, the user may be able to select the cases that are of interest and move onto the adaptation process. The user also needs the ability to look at the information contained within the case when reviewing and so the user has the ability to look at the equations, constants and variables and to see an occurrence matrix for the selected case. The occurrence matrix (see figure 5.1) is a graphical method that can be used to check that the degrees of freedom for the case are zero, i.e. that there is the same number of unknown variables as equations. The occurrence matrix shows the equations and variables for the case and indicates the presence of a variable in an equation by a *. It also shows the total number of equations and unknown variables present, thus acting as a quick check to ensure the problem is described fully.

IMIPS - Occurence Matrix

File Help

	A	B	C	D	E
1		X	Y	Z	
2	$X = -K*((X-Y)/(M*Vol1))	*	*		
3	$Y = K*((X-Y)/(M*Vol2))	*	*		
4	Z = (X0-X)/X	*		*	
5	**3 Equations: 3 Unknown Variables**				
6					
7					
8					
9					
10					

Figure 5.1. Occurrence Matrix.

5.5 Case Adaptation

The adaptation of the selected case or cases in a process engineering environment is a very difficult task. The user must have a good grasp of the case to be adapted and the case they want to model. To automate this procedure would be incredibly hard as the adaptation of a past case is not as straight forward as in other areas where this technique has been employed, e.g. Menu selection and adaptation (see §3.4.1). The automatic adaptation procedure has not yet been implemented within IMIPS because of the complexity of the task.

Due to these problems and complexities it was decided to use manual adaptation for IMIPS. This allows the user to review and adapt retrieved cases to reach a solution. The user is prompted to ensure they annotate all changes they perform and then record the success or failure of their amendments.

5.6 Translation Procedure

The general translation procedure is described below for the creation of both Simulink and gPROMS input files. The automatically generated gPROMS input file is the same as that which the user would see if the task were done manually (a text file). However, the Simulink input file when seen in Simulink looks like figure 5.2. The user has no direct interaction with the text file used to create this, although this is the file created in translation.

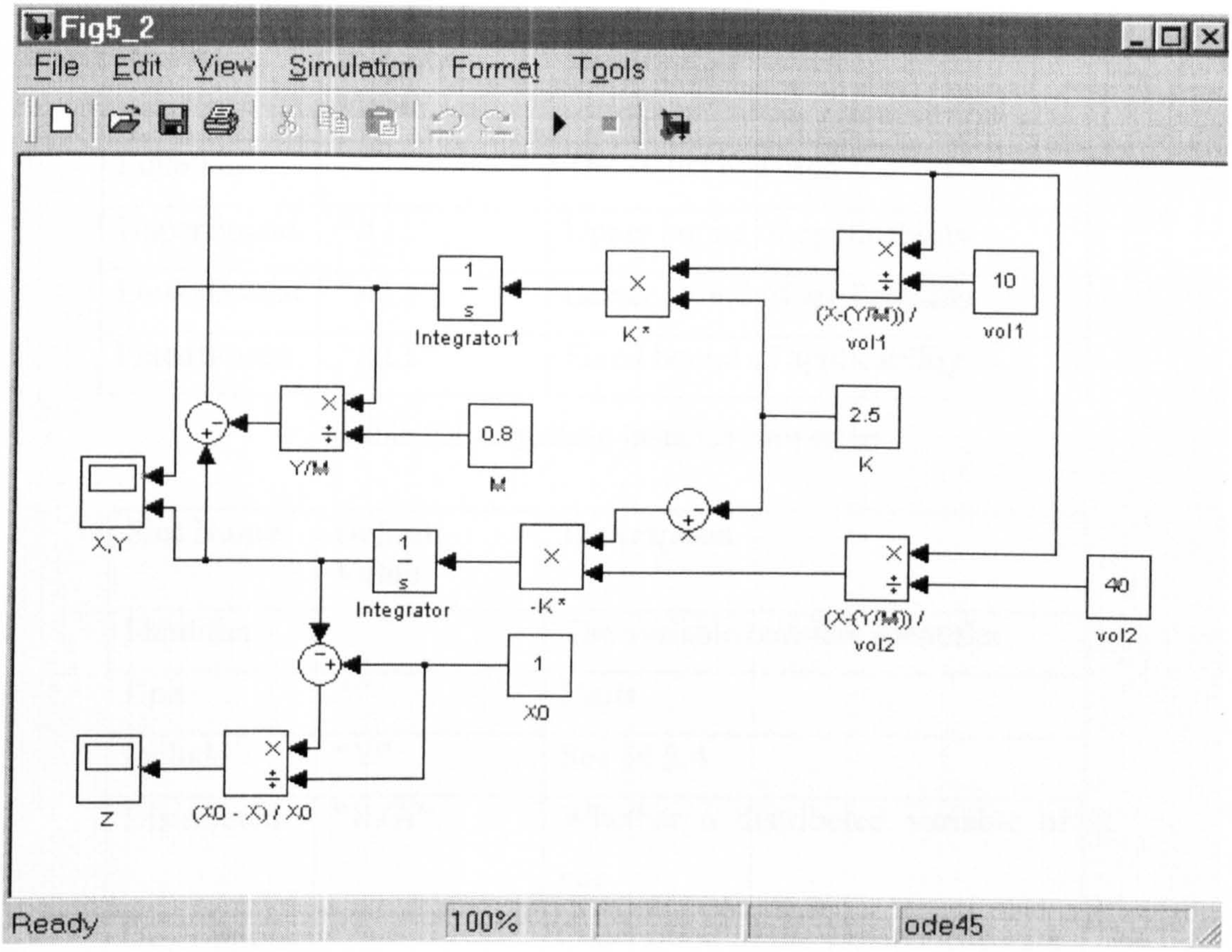


Figure 5.2. Simulink model, as seen in Simulink.

5.6.1 Case Initialisation

The first stage in translating any case into the respective input file is the creation of variable, constant, and equation instances. An instance of an item contains all the information related to the item. The information may be stated explicitly, or may be a default value. By using an object-oriented language, the ability to create, use, and manipulate instances is a great benefit to the later stages of the translation. Once a case has been chosen and the translation stage started the case is transferred to instances containing all the information in the case. This is the initialisation stage. Each variable, constant, and equation has properties associated with it. These properties are set at initialisation and are shown in the tables below. The properties are set from the values given in fields in the database unless the field is left empty. In this case the value is left set as the default (see tables below).

Slot Name	Default Value	Description
Equation		The stated equation
UpperBound	"All"	Upper bound of applicability
LowerBound	"All"	Lower bound of applicability
FixedBound	"All"	Fixed bound of applicability

Table 5.1. Equation instance properties.

Slot Name	Default Value	Description
Identifier		The variable/constant identifier
Unit	""	Units
Include	"Y"	See §4.3.4
Distributed	"N/A"	Whether a distributed variable or not
Description	""	Description of the variable/constant
ArrayVar	0 "none"	Stores the number of elements (0) and their values ("none") if the constant/ variable is an array
Port	0	Stores the input port identifier value used in the Simulink translation
OutPorts	0	Stores the output port identifier value used in the Simulink translation

Table 5.2. Variable and Constant instance properties.

Slot Name	Default Value	Description
Set	"unknown"	Variable set value
Value	"unknown"	Variable initial value
UpperBound	"All"	Upper bound for distributed variable
LowerBound	"All"	Lower bound for distributed variable
MathMethod	"N/A"	Mathematical method used to solve for distributed variable
Type		Used in gPROMS translation

Table 5.3. Variable instance properties.

Slot Name	Default Value	Description
Value	0.1	Constant value

Table 5.4. Constant instance properties.

Where the constant/variable has a value in the case, this overwrites the default value given. All constant or variable arrays are split into separate values and are stored in a multivalue slot with the first value being the number of elements in the array.

The input file is then created using the name given in the General Information form (or other specified by the user). The two translators now differ to create the files needed for each package. These will be explained individually, with the Simulink translator shown first.

5.6.2 Simulink Translator Implementation

The basic layout of a Simulink input file is shown in figure 5.3 and a flow-diagram of the process is shown in figure 5.4.

```
Version Information (including name)
Solver Information (including start and stop times)
History (including creation date and last modified by)
BlockDefaults
AnnotationDefaults
LineDefaults
System
  Block
  Line
  Branch
```

Figure 5.3. Basic Simulink input file outline.

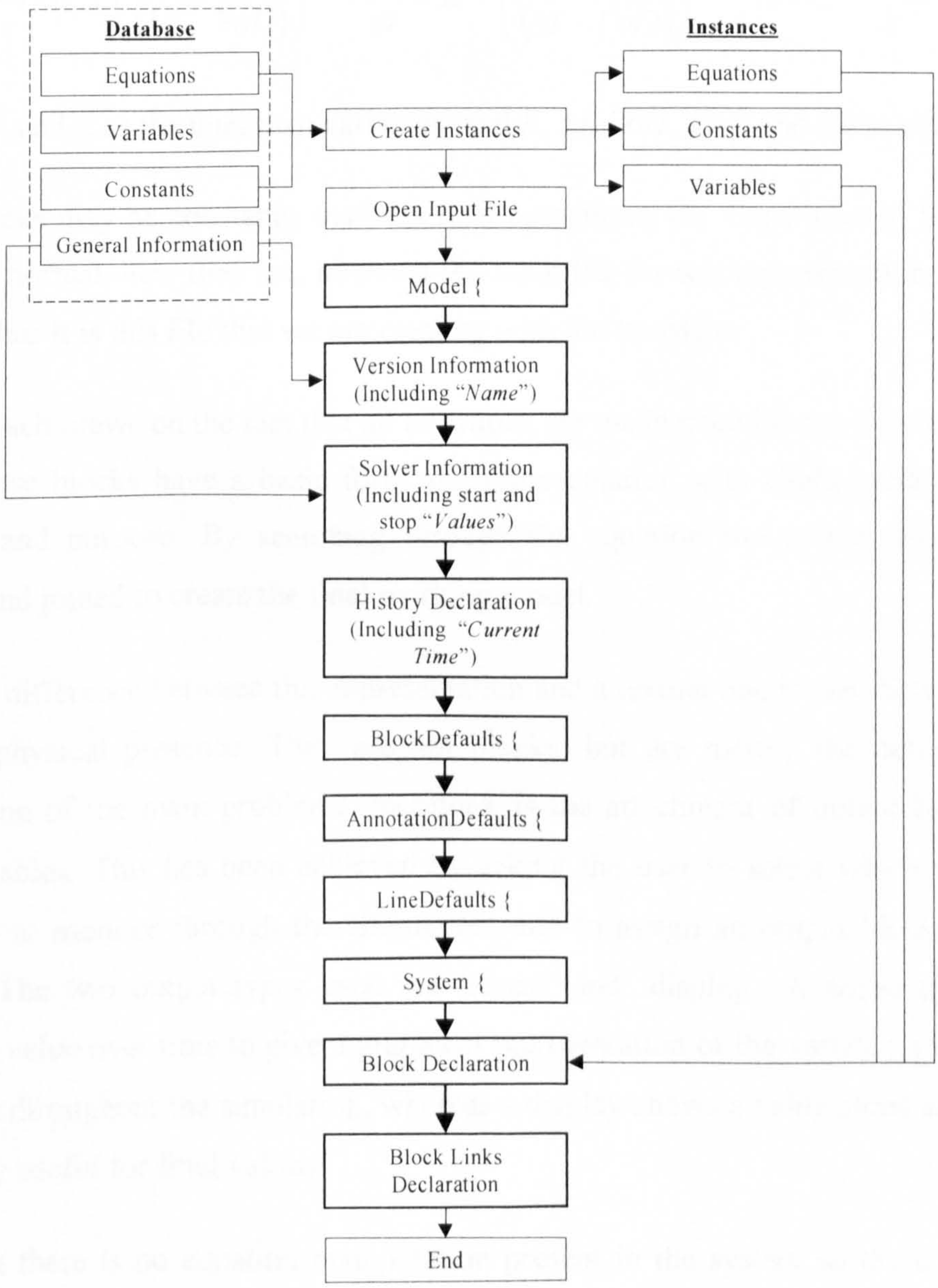


Figure 5.4. Simulink translation flow diagram. (For complete diagram see Appendix III.)

The method for creating the translator follows through the steps of creating a simple model through to a complex problem. Starting with a simple set of equations, through DAE's and onto IPDAE's. When approaching the problem, the initial cause of concern was the different method used by Simulink to represent equations.

As was stated in §5.6, the Simulink method for representing equations is quite different to that of textual interface based simulators. As is shown in figure 5.2 the way an equation is stated is through the linking of various blocks. The equations in this figure are:

$$\frac{dX}{dt} = -K \cdot \left(\frac{(X-Y)}{(M \cdot Vol1)} \right) \qquad \frac{dY}{dt} = K \cdot \left(\frac{(X-Y)}{(M \cdot Vol2)} \right) \qquad Z = \frac{(X0-X)}{X}$$

With X, Y, and Z as the unknown variables, and K, M, $Vol1$, $Vol2$ and $X0$ as constants.

These blocks may be constants, mathematical operations, etc. For a user of Simulink this is the normal view they see, however the input file for this representation is a text file, and thus it is this file that we are creating with the translator.

The approach draws on the fact that all equations are made from blocks linked to each other. These blocks have a basic form and representation with slight variations for position, and purpose. By searching through the equation the correct blocks are selected and joined to create the final working model.

The other difference between this representation and a textual one is that the variables have no physical presence. They are not blocks, but are merely the output from blocks. One of the main problems, therefore, is the attachment of output blocks to these variables. This has been achieved by asking the user to select which variable they wish to monitor through the simulation, and to assign an output block to that variable. The two output types used are 'scope' and 'display'. A scope plots the variable's value over time to give a graphical representation of the variable's value as it changes throughout the simulation, whereas a display shows a value alone and so is only really useful for final values.

At present there is no equation manipulation present in the system so the equations must be written in a certain order to be translated properly (see later in this section).

The initial statements and structure that are in all Simulink files are very easy to include in the translated file, with the time, date and simulation run times added to the default values. The equation checking is the most complicated task for the translator. The selection of the correct blocks is quite straight forward, with the equation being decomposed, brackets first, and the correct symbol being selected and the code written. For each block, two pieces of information are needed to correctly connect it to each input. This information is the block to connect to, and the port from that block to use. These inputs are either a constant or an output from another block. When connecting blocks the input port in use is stored in the 'Port' slot of the instance (See table 5.2).

Each equation is decomposed and the relevant blocks are included in a straight line on the graphical view. There is, at present, no block re-ordering, to give a layout for the blocks so the view the user would see can be quite jumbled.

When solving equations in Simulink it is necessary to state the variable that the equation is to be solved for on the left. This is due to Simulink not having physical entities for the unknown variables. They must be stated as the solution of an equation so that equation can be solved to produce a value for the unknown variable. This can be shown in the following example. If you wished Simulink to solve $Holdup_i = Xout_i \cdot TotalHoldup$ for $Xout_i$ it would have to be written as $Xout_i = \frac{Holdup_i}{TotalHoldup}$ where $i = 1 - n$ for n components, $Holdup_i$ and $TotalHoldup$ are known. This can also be seen when dealing with DAE's. In Simulink when trying to solve a DAE the user does not state the differential equation as it is written. Instead, they must state what the differential is equal to, and then integrate to solve it. So the equation $\frac{dX}{dt} = -K \cdot \left(\frac{(X-Y)}{(M \cdot Vol)}\right)$ is actually stated in Simulink as $X = \int\left(-K \cdot \left(\frac{(X-Y)}{(M \cdot Vol)}\right)\right)dt$ (See figure 5.5, with X and Y being variables, and M, K and Vol constants)

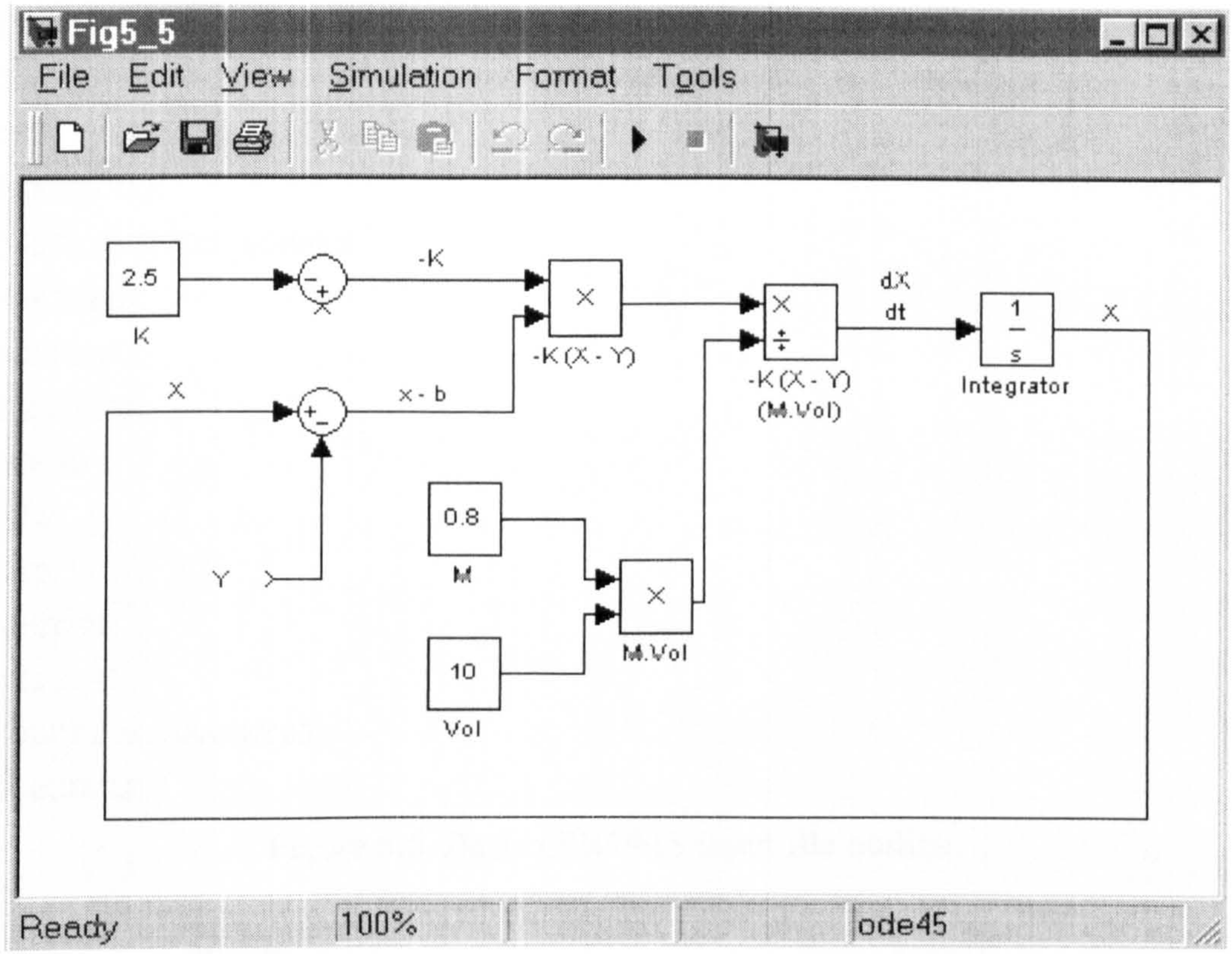


Figure 5.5. Basic DAE, as seen by a user in Simulink.

Therefore, as there is no equation manipulation in the translator at present, the equations must be stated in a certain order to be correctly translated.

Simulink has two main drawbacks associated with it at present, its inability to state constants as a multi value array (larger than a vector) and its lack of solution method for dealing with IPDAE's. The first of these problems may be addressed through future research into a solution, but the second is a problem within Simulink itself. At present, there is no partial derivative solver for Simulink, but MathWorks are writing a solver that should be completed soon.

5.6.3 gPROMS Translator Implementation

The basic layout of a gPROMS input file is shown in figure 5.6 and a flow-diagram of the process is shown in figure 5.7.

```
# Title
DECLARE
MODEL
  PARAMETER
  DISTRIBUTION_DOMAIN
  VARIABLE
  BOUNDARY
  EQUATION
PROCESS
  UNIT
  SET
  ASSIGN
  INITIAL
  SOLUTIONPARAMETERS
  SCHEDULE
```

Figure 5.6. Basic gPROMS input file outline.

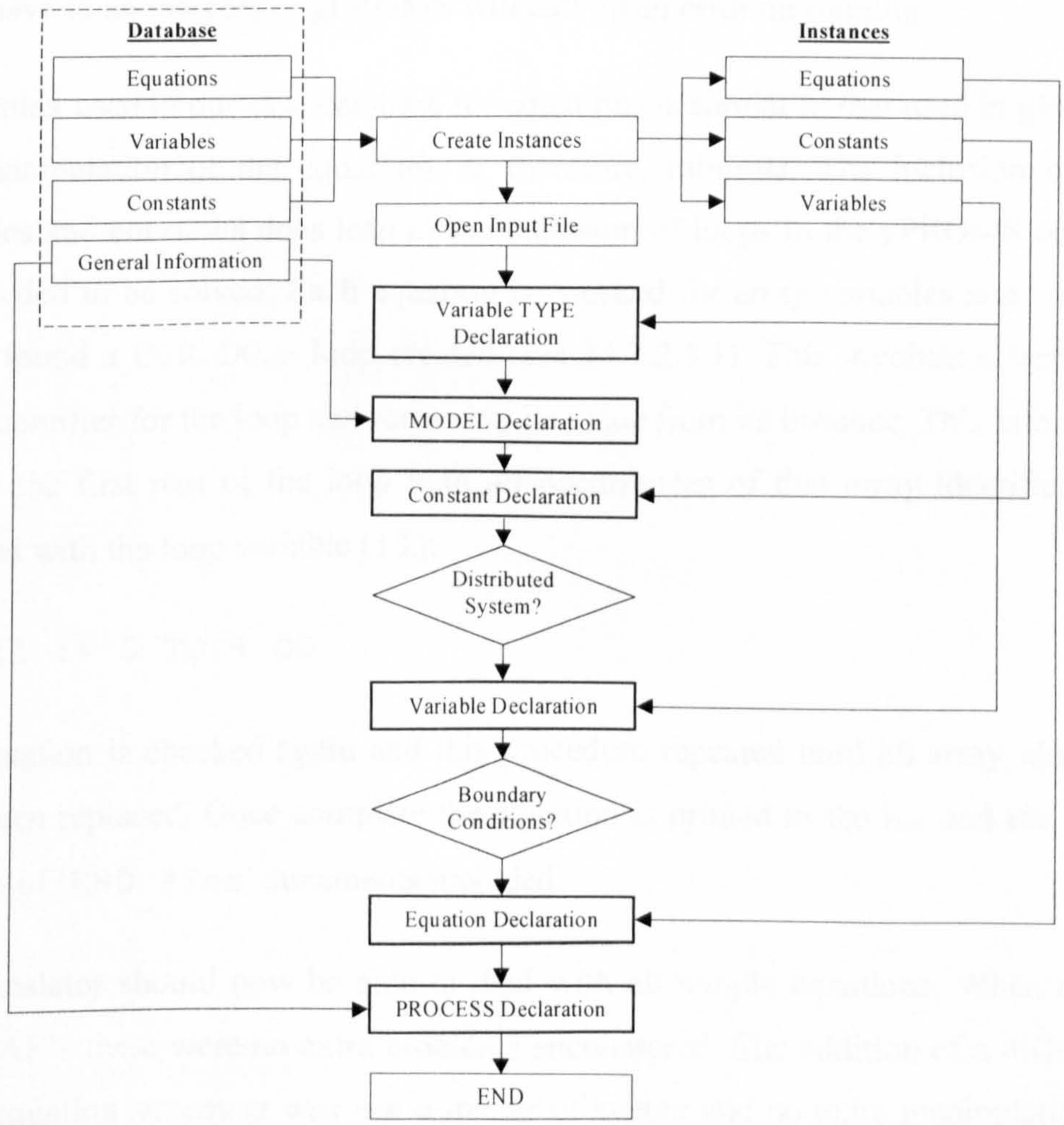


Figure 5.7. gPROMS translation flow diagram. (For complete diagram see Appendix III.)

Like the Simulink translator, the method for creating the gPROMS translator followed through the steps of creating a simple model through to a complex problem. Starting with a simple set of equations, through DAE's and onto IPDAE's. Unlike the problem in Simulink of the representation being very different to that of describing a process by mathematical equations, a gPROMS input file is a textual one and is the file the user would write were they to use gPROMS themselves. The greatest problems to be overcome are the creation of the correct syntax for the statements in the equations.

The inclusion of arrays in both the variable and constant definitions was the first problem to be overcome (see §4.7.2.1). Each constant/variable is checked for sets of brackets. This indicates the presence of an array. From this the correct syntax is used in the variable/constant declaration. The actual values are not needed until much later in the file, but the constants are checked to see if the values are `REAL` or `INTEGER`. This is vital for the 'identifier' values, e.g. number of components (`NoComp`), etc, as these have to be integers or gPROMS will call up an error on running.

The syntax used in our case database for equations is similar to that used in gPROMS. The manipulation of the equations is, therefore, minimal. The inclusion of array variables and constants does lead to the inclusion of loops in the gPROMS code and this needed to be solved. Each equation is checked for array variables and constants and if found a `FOR...DO...` loop created (see §4.7.2.3.1). This involves selecting the array identifier for the loop and retrieving its value from its instance. This information creates the first part of the loop with all occurrences of that array identifier being replaced with the loop variable (`i1`):

```
FOR i1 := 0 TO R DO
```

The equation is checked again and this procedure repeated until all array identifiers have been replaced. Once complete the equation is printed to the file and the correct number of '`END #For`' statements included.

The translator should now be able to deal with all simple equations. When dealing with DAE's there were no extra problems encountered. The addition of a differential to the equation statement was just a matter of syntax and no extra manipulation was needed to incorporate this.

More problems were encountered, however, when trying to deal with IPDAE's. The inclusion of integrals and partial differentials to the equations led to more manipulation of the stated equations and some rearrangement of the input statements. It also led to the inclusion of distributed variables being declared. These are selected if the variable has lower or upper bounds stated in the case. Also the distributed domain variables are selected and declared separately from the other variables, by checking for limits in the distributed slot (see table 5.2) of that instance.

Partial differentials are stated in the database such that when the equations are being written in the input file certain parts of the statement are removed, or changed (see § 4.7.2.3.2). This needed a different approach from the simple replacement of the array identifiers seen in `FOR...DO...` loops. The equation is, therefore, checked for `PARTIAL` statements before the array identifier replacement procedure is called. When checking the partial differentials the limits of applicability also need to be checked to ensure the correct boundary conditions are applied.

Integrals are, again, dealt with slightly differently from the loops and partial differentials (see §4.7.2.3.3). Again, there is the need to replace some occurrences of variable names before the equations are checked for array identifiers. This is completed before the partial derivatives are translated.

The input file has statements for the values of the constants and the initial values (if applicable) of the variables. These are stated at the end of the input file and, for single values, are quite straight forward. For the declaration of distributed variable initial values, loops need to be included and these were created in the same way as for the equations. The array identifiers were replaced and `FOR...DO...` loops written.

5.7 Summary

This chapter has shown the design and implementation issues encountered in the creation and development of IMIPS. Issues relating to the implementation of the past case database were discussed. Then, what needed to be considered during case retrieval, and adaptation was highlighted. Finally, the nuances of the translation procedure for the simulators for Simulink and gPROMS were outlined.

The next chapter presents some case studies to show how IMIPS has been used.

6. Case Studies

6.1 Introduction

This chapter aims to run through some case studies to show the properties of the system and to outline its benefits. The basic use of IMIPS was shown in §4.8. As was outlined in §4.8 IMIPS can, at present, cater for three different scenarios: the creation of a new case from scratch, retrieval of an existing case and the retrieval and adaptation of existing cases. All these scenarios include an automatic simulator input code generation, and, to some extent, a search and review stage.

The interface properties that are to be described are:

- The creation and translation of a case from scratch, §6.2.
- A search for similar, or related, cases for use in solving a plug flow reactor problem (demonstrates the use of PDE's), §6.3.
- A search for similar, or related, cases for use in solving a cooling jacketed vessel problem using specified numerical range(s) (demonstrates additional functionality to §6.3), § 6.4.
- A search for similar, or related, cases for use in solving a continuous stirred tank reactor problem (demonstrates use of DAE's), §6.5.

The results shown in the following case studies are from the default solvers within the simulators. More information on the solution methods used can be found in the following: Simulink, §9, MathWorks, (1999); gPROMS, §I-16, PSE (1998).

For each case the figures mentioned in the text are grouped at the end of each section.

6.2 Case 1 – Case from Scratch, Simple Batch Extraction

This case study looks at how to use IMIPS to create a case from scratch, and then recall and translate the case into input files for gPROMS and Simulink. The following procedure outlines how this is achieved.

1. Case Statement – writing the new case in the database, §6.2.1.

2. Case Retrieval and Review – retrieving and reviewing the case in IMIPS, §6.2.2.

3. Case Translation – translating the case into gPROMS and Simulink input files, §6.2.3.

6.2.1 Case Statement

When stating a case from scratch the user inputs all data through the database interface forms. These forms and the information required to state a case fully have been outlined in §4.3. The case to be investigated is the modelling of a simple batch extraction process (Ingham *et al.*, 1995, pp. 527-530).

The process takes place in a column, where, at time 0, two liquid layers are present, with all the product in layer X (see figure 6.1). The volumes of the two layers are known, as are the mass transfer coefficient, the equilibrium constant, and the initial concentration of the product (shown below). All this information can be stated in the database (see figures 6.2, 6.3, 6.4, and 6.5). For this simple case the equations needed to model the system are:

$$\frac{dX}{dt} = -K \cdot \left(\frac{\left(X - Y/M\right)}{Vol1} \right)$$ Change in concentration of product in Vol1. (6.2.1.1)

$$\frac{dY}{dt} = K \cdot \left(\frac{\left(X - Y/M\right)}{Vol2} \right)$$ Change in concentration of product in Vol2. (6.2.1.2)

$$Z = \frac{(X0 - X)}{X0}$$ Fractional extraction of product from Vol1 to Vol2. (6.2.1.3)

With the constants and variables being:

K	2.5	Rate Constant ($m^3.hr^{-1}$)
M	0.8	Equilibrium Constant
Vol1, Vol2	10, 40	Batch Volumes (m^3)
X0	1	Initial Concentration in X phase
X	Initially 1	Concentration in X phase ($kg.m^{-3}$) (unknown)
Y	Initially 0	Concentration in Y phase ($kg.m^{-3}$) (unknown)
Z		Fractional Extraction (unknown)

The general information relating to this case is stated and the user has the opportunity to write the details of the case down. In this example, the user has kept the default values for the Unit Name, and the Applicable Range. The user has stated the following: *Equipment type:* Other_Column, *Operating Mode:* Batch, *Phases:* Liquid, and *Mass Transfer:* Absorption. They have also given the problem a name, CS1, and changed the default *Reporting Interval* and *Time for Run* fields.

The equations are written in the database using the syntax shown in Appendix I.

The user needs to ensure their problem has zero degrees of freedom, i.e. that there are the same number of unknown variables as there are equations. This is easy to check in this example as there are only three variables. With larger problems the user may have to wait until the problem is specified and retrieved in IMIPS and use the occurrence matrix feature to check this.

Once the user is happy with the case statement in the database they move to IMIPS to retrieve and translate the case.

6.2.2 Case Retrieval and Review

To retrieve the new case the user may search using key-words, or using the case number. The case number approach will ensure that their case is retrieved without having to view many possible search results. From here the user can view the information about the case (figure 6.6) and can also check the occurrence matrix (figure 6.7) for the case to ensure that it has zero degrees of freedom (§5.4). If this is

not the case the user must return to the case description in the database and alter the case to solve the problem. This is the only check the system can perform on a new case to see if it is correctly formulated. If the degrees of freedom equals zero this does not imply that the system of equations will be solvable, only that they could be. For more complete problem checking the user needs to look at the feedback given by the simulators on running the translated input files, should the problem not run correctly. Automatically retrieving this feedback and relating it to the case in the database is a topic for future work (see §7.2.2).

The occurrence matrix states that there are zero degrees of freedom and so the case may be solvable. The user now moves to the translation stage.

6.2.3 Case Translation

The retrieved case is translated to the two simulator input files. These files are shown in Appendices V.1 and V.2. The Simulink file as opened in Simulink is shown in figure 6.8, and the results of running the simulations are shown in figures 6.9 and 6.10.

Ingham *et al.* (1995) include a simple modelling tool with their book, ISIM. Using this tool the problem was solved and the results are shown in figure 6.11.

As can be seen from these plots, the three sets of results agree.

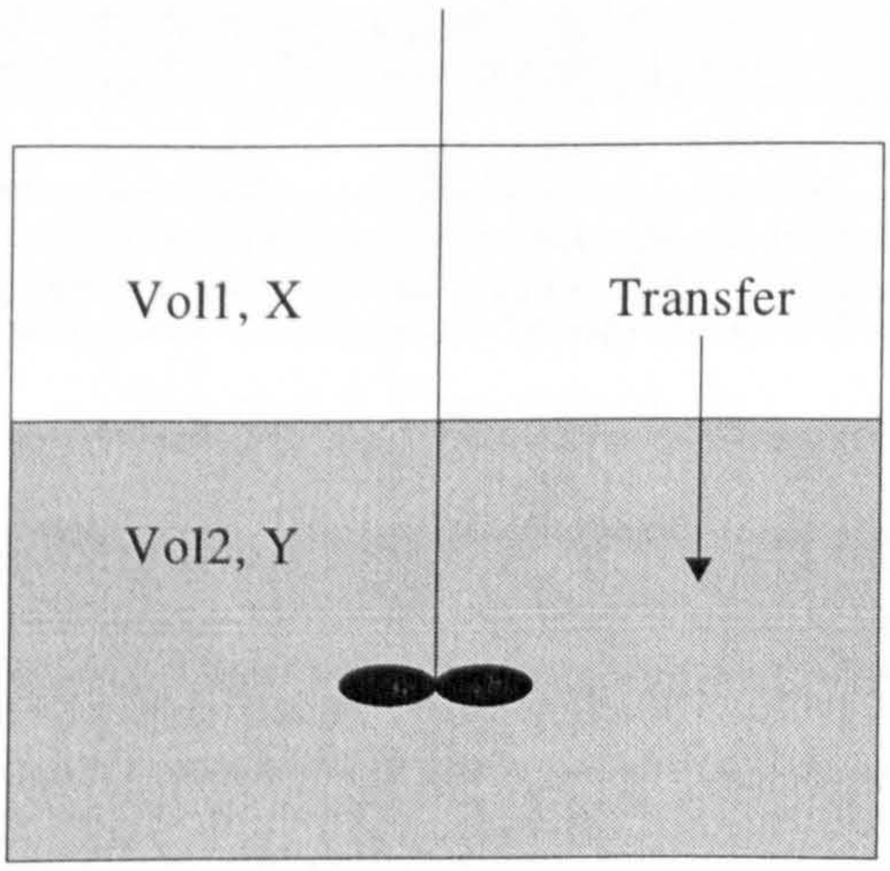


Figure 6.1. Simple batch liquid-liquid extraction.

Data

Simple Batch Extraction

Problem Number: 4

Simple batch extraction. Extracting product from Vol1 to Vol2. Case Study 1

Equipment	/Other_Column/	
Operating Mode	/Batch/	
Thermal Behaviour	/n/a/	
Reaction Type	/n/a/	
Reaction Order	/n/a/	
Reversible	/n/a/	
Phases	/Liquid/	
Mass Transfer	/Absorption/	
Heat Transfer	/n/a/	
Pipeline Flow	/n/a/	
Chemical	/n/a/	

Problem Name: Simbatex Unit Name: CS1

Reporting Interval: 0.5 Time for Run: 20

Operating Conditions / **Applicable Range**

	Low	High	Unit			Unknown
Temperature			C	-273	2147483647	
Pressure			Pa			

Record: 4 of 18

Figure 6.2. Case general information form.

Equations

	Equations	Fixed Bound	Lower Bound	Upper Bound	Description
1	$X = -K*((X-(Y/M))/Vol1)				Change in concentration in Vol1
2	$Y = K*((X-(Y/M))/Vol2)				Change in concentration in Vol2
3	Z = (X0-X)/X0				Fractional Extraction
4	END				

Record: 4 of 19

Figure 6.3. Case Equations.

Constants

	Constant	Value	Units	Inc. ?	Description
1	K	2.5		Y	Mass Transfer Coefficient
2	M	0.8		Y	Equilibrium Constant
3	Vol1	10		Y	Batch Volume
4	Vol2	40		Y	Batch Volume
5	X0	1		Y	Initial Concentration
				Y	

Record: 4 of 17

Figure 6.4. Case Constants.

Variables

	Variable	Initial	Set	Units	Lower Bound	Upper Bound	Distribution	Solution Method	Description
1	X	1							Concentration in liquid
2	Y	0							Concentration in Vapor
3	Z								Fractional Extraction
4									
5									

Record: 4 of 10

Figure 6.5. Case Variables.

Intelligent Modelling Interface for Process Simulators

File Forms Report Translate Help

EQUATIONS

$X = -K*((X-(Y/M))/Vol1)
; Change in concentration in Vol1

$Y = K*((X-(Y/M))/Vol2)
; Change in concentration in Vol2

Z = (X0-X)/X0
; Fractional Extraction

CONSTANTS

K 2.5 ; Mass Transfer Coefficient

M 0.8 ; Equilibrium Constant

Vol1 10 ; Batch Volume

Vol2 40 ; Batch Volume

X0 1 ; Initial Concentration

VARIABLES (Mathematical Method)

X 1 ; Concentration in liquid

Y 0 ; Concentration in Vapor

Z ; Fractional Extraction

Figure 6.6. More Info… Screen.

IMIPS - Occurence Matrix

File Help

	A	B	C	D	E
1		X	Y	Z	
2	$X = -K*((X-(Y/M))/Vol1)	*	*		
3	$Y = K*((X-(Y/M))/Vol2)	*	*		
4	Z = (X0-X)/X0	*		*	
5	**3 Equations: 3 Unknown Variables**				
6					
7					

Figure 6.7. Occurrence Matrix.

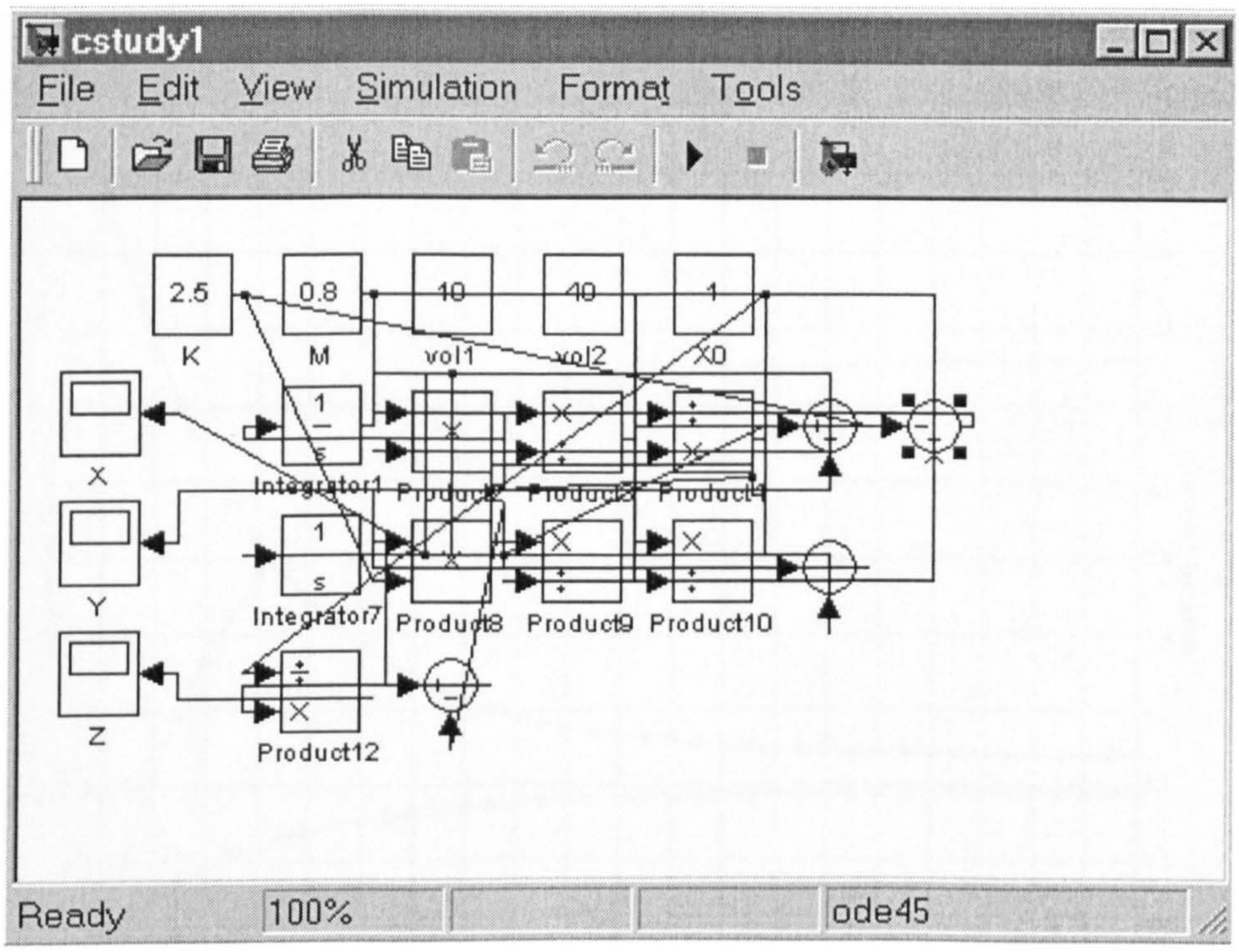


Figure 6.8. Simulink file as seen in Simulink.

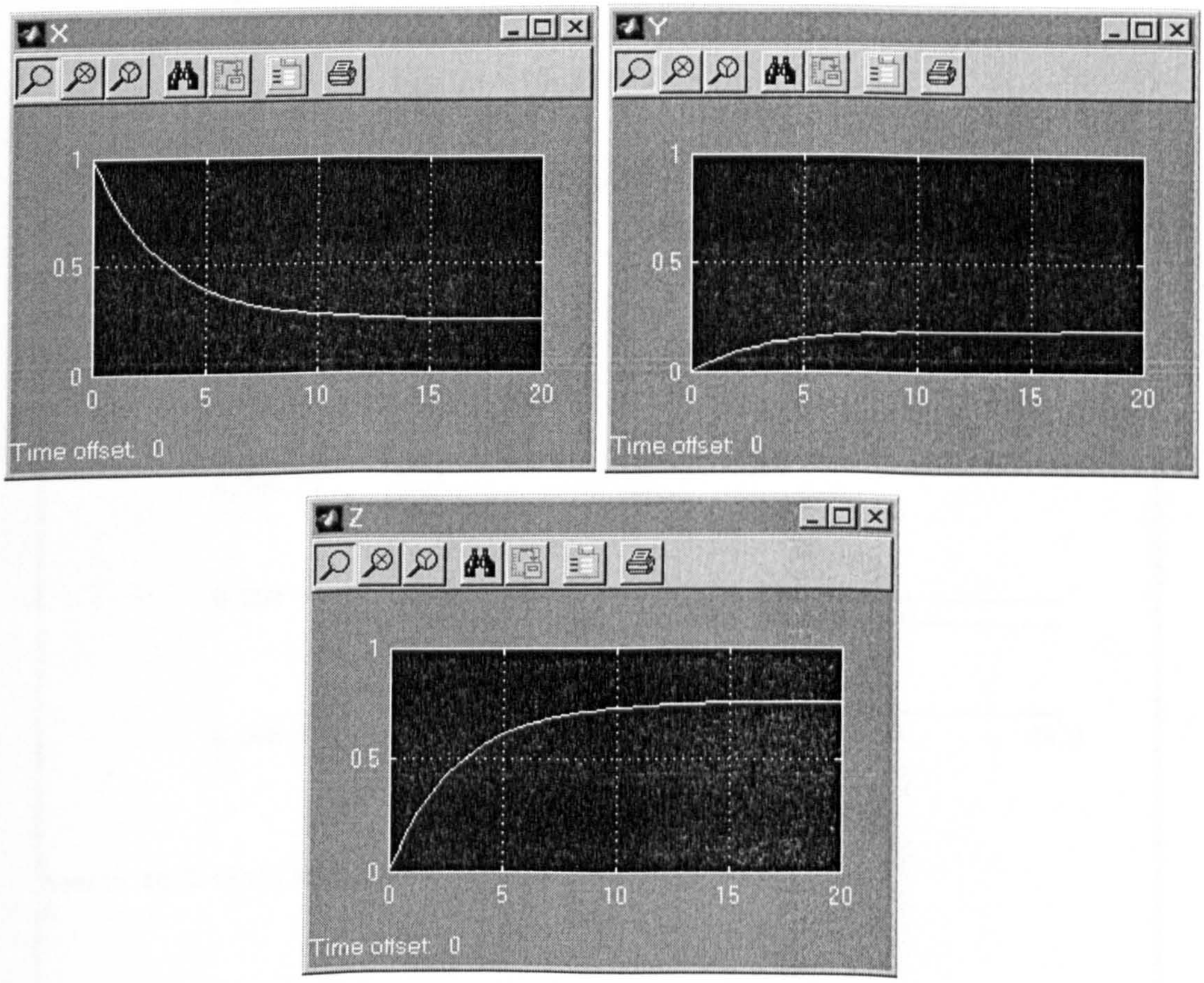


Figure 6.9. Simulink results.

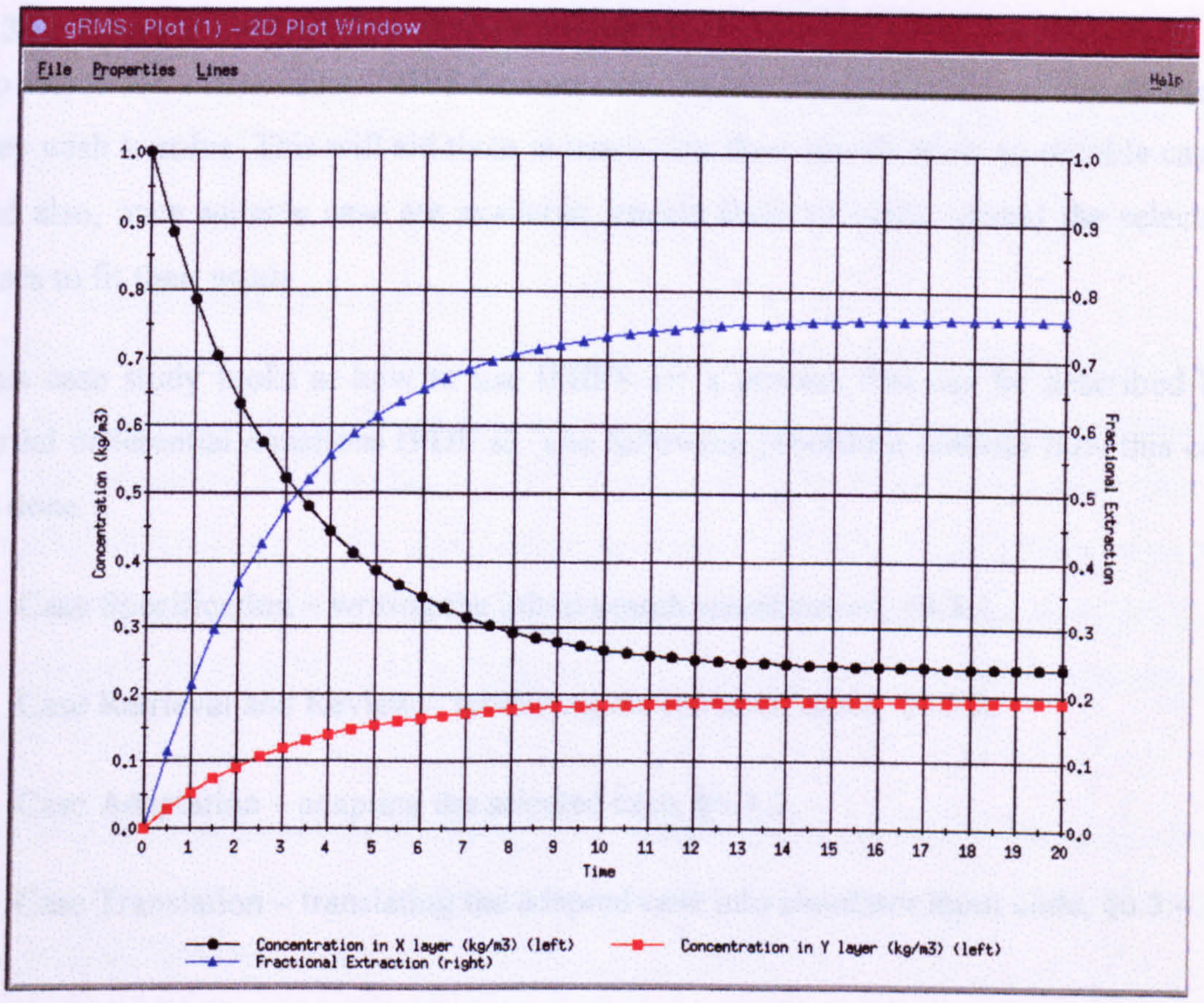


Figure 6.10. gPROMS results.

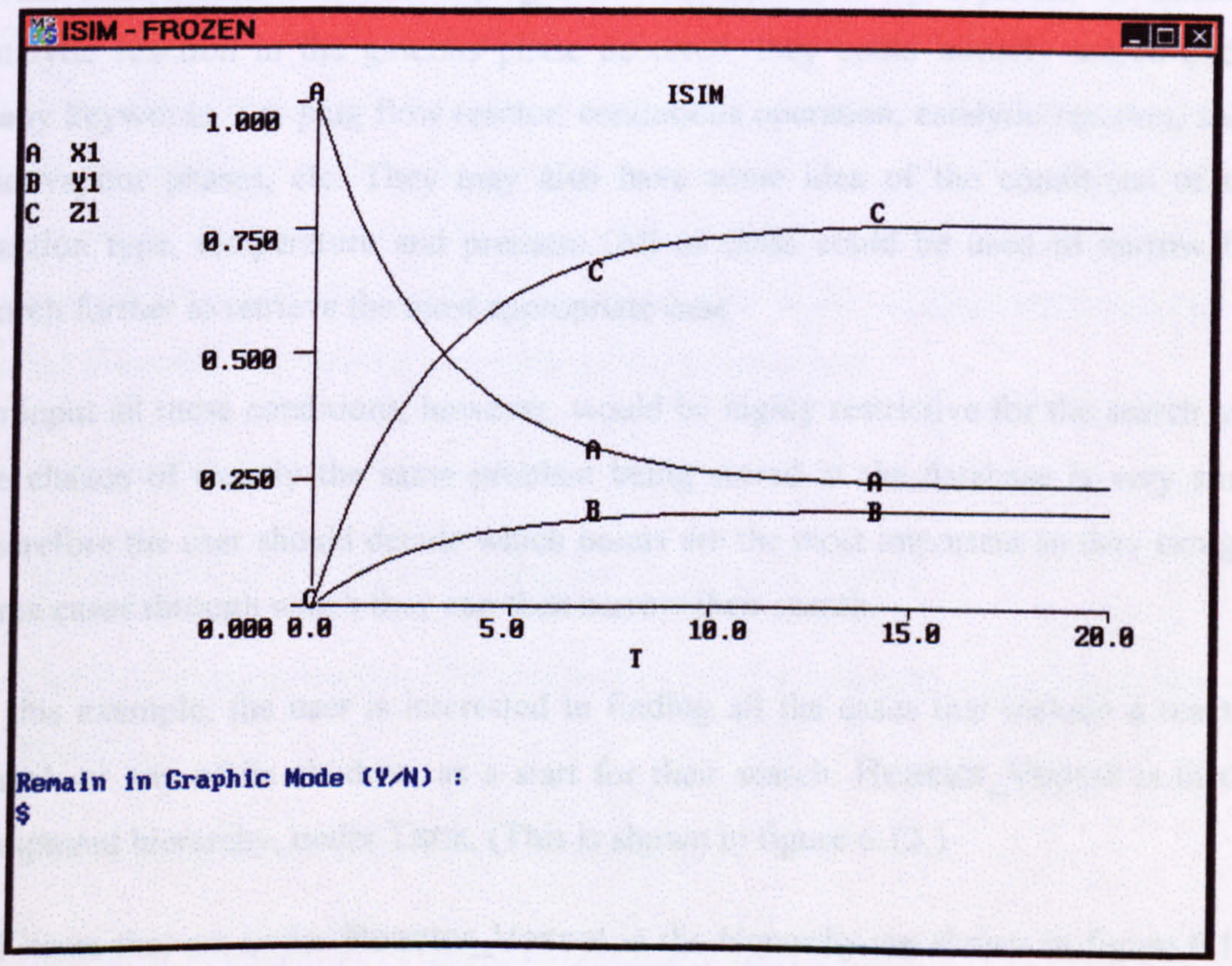


Figure 6.11. ISIM results.

6.3 Case 2 – Plug Flow Reactor Search

To search for a case using IMIPS the user must have a reasonable idea of the problem they wish to solve. This will aid them in narrowing their search down to suitable cases and also, once suitable case are available, enable them to easily amend the selected cases to fit their needs.

This case study looks at how to use IMIPS for a process that can be described by partial differential equations (PDE's). The following procedure outlines how this can be done.

1. Case Specification – writing the initial search specification, §6.3.1.
2. Case Retrieval and Review – reviewing the retrieved cases, §6.3.2.
3. Case Adaptation – adapting the selected case, §6.3.3.
4. Case Translation – translating the adapted case into simulator input code, §6.3.4.

6.3.1 Case Specification

If the user wished to model a plug flow reactor, cooled by a jacket, in which a catalytic reaction in the gaseous phase occurred, they could initially search under many keywords, e.g. plug flow reactor, continuous operation, catalytic reaction, solid and vapour phases, etc. They may also have some idea of the conditions of the reaction type, temperature and pressure. All of these could be used to narrow the search further to retrieve the most appropriate case.

To input all these conditions, however, would be highly restrictive for the search and the chance of exactly the same problem being stored in the database is very slim. Therefore the user should decide which points are the most important so they can get some cases through which they can then narrow their search.

In this example, the user is interested in finding all the cases that include a reactor vessel, or any of its children, as a start for their search. Reactor_Vessel is in the equipment hierarchy, under Tank. (This is shown in figure 6.12.)

All items that are under Reactor_Vessel in the hierarchy are shown in figure 6.13. The search specification for this is shown in figure 6.14. It can be seen that the

'children' box is checked. This allows a search query to be created to check for cases that have equipment specified as Reactor_Vessel or a type lower down the indexing hierarchy (see figure 6.13).

6.3.2 Case Retrieval and Review

After pressing the 'Search' button the results are shown in a form very similar to that in the database (figure 6.15). At the bottom of the form is a box showing the number of cases retrieved, for this search specification, five. The retrieved cases may then be scrolled through and if the user wishes to inspect the case more closely then the equations, constants and variables can be seen (with their associated properties) by pressing the 'More Info...' button (figure 6.17), or using the drop down menu (figure 6.16).

The five cases retrieved are: two plug flow reactor cases and three continuous stirred tank reactor cases. These five cases are returned because their equipment specification is a child of Reactor_Vessel.

6.3.3 Case Adaptation

If a retrieved case is of use then the user may go back to the database, copy the retrieved case and then amend it for their specific parameters. The user may amend any of the attributes of the case to create a case with their desired specifications. In this example, the user only wishes to amend some of the values of the constants and some variable set and initial values. This is achieved in the database where the case to amend is copied and then the appropriate changes are made.

The amendments made to the original case are:

- The changing of the reactor dimensions (length and radius).
- The bed voidage fraction, the temperature of the reactants and coolant into the reactor.
- The velocity and mass flowrate into the reactor.
- The initial temperatures within the reactor and cooling jacket.

- The time for the run and the reporting interval for the results.

The different values used are shown in §6.3.4.

Once the user is satisfied with the new case (based on the retrieved case) they save it in the database and return to IMIPS, where they may retrieve their new case using the case number. (It is quicker to search for individual case numbers, where known, rather than searching as for a new case.)

6.3.4 Case Translation

To ensure the translation of the new case is correct the original case can be translated and compared to the results produced by the original file (PSE, 1998). If the retrieved case is translated the following graph can be plotted (figure 6.18) and when compared to the original results (figure 6.19) it can be seen that they are the same.

The new case is then recalled (figure 6.20) and, once selected, can be translated into either or both of the modelling package input files using the drop down 'Translate' menu (figure 6.20). For this particular case (based on an example from PSE, 1998) since the equations are PDE's only the gPROMS translator can be used. At present, as was outlined towards the end of §5.6.2, Simulink does not have the necessary blocks to solve IPDAE's or PDE's as are present in this problem. Therefore, this case can only be translated by one of the translators. If the user tried to translate this case into a Simulink file, an error message would pop up indicating Simulink's shortcoming (figure 6.21).

The gPROMS input file created automatically from the case description input by the user is in appendix V.3. For this case the equations retrieved required no amendments to be suitable for the problem, only the values of some constants and variables were changed. These amendments are (sections from gPROMS input file code shown):

Originally Retrieved Case

```
SET
  WITHIN R101 DO
    L     := 3; # m
    R     := 0.0127; # m
    BedVoid     := 0.35; #
    Vc     := 4.56E-03; # m3
    Area     := 0.239; # m2
  END # Within

ASSIGN
  WITHIN R101 DO
    Pfeed     := [1100,21100]; # Pa
    Tfeed     := 625; # K
    u     := 0.877; # m/s
    Fc     := 0.1; # kg/s
    Tcin     := 625; # K
  END # Within

INITIAL
  WITHIN R101 DO
  FOR i1 := 0|+ TO L|- DO
   FOR i2 := 0|+ TO R|- DO
    T(i1,i2)      = 625; # K;
   END #For
  END #For
    Tc      = 625; # K
  END # Within

SOLUTIONPARAMETERS
  ReportingInterval := 0.2;

SCHEDULE
  CONTINUE FOR 5
```

New Case

```
SET
  WITHIN CS101 DO
    L     := 4; # m
    R     := 0.025; # m
    BedVoid     := 0.25; #
    Vc     := 0.0100; # m3
    Area     := 0.628; # m2
  END # Within

ASSIGN
  WITHIN CS101 DO
    Pfeed     := [1100,21100]; # Pa
    Tfeed     := 550; # K
    u     := 1; # m/s
    Fc     := 0.05; # kg/s
    Tcin     := 550; # K
  END # Within

INITIAL
  WITHIN CS101 DO
  FOR i1 := 0|+ TO L|- DO
   FOR i2 := 0|+ TO R|- DO
    T(i1,i2)      = 550; # K;
   END #For
  END #For
    Tc      = 550; # K
  END # Within

SOLUTIONPARAMETERS
  ReportingInterval := 0.1;

SCHEDULE
  CONTINUE FOR 10
```

After running the translated file in gPROMS the following graphs can be plotted: Figure 6.22 shows the change in temperature down the reactor centre-line with time, figure 6.23 shows the change in temperature down the reactor perimeter with time, and figure 6.24 shows how the temperatures at the centre and perimeter inlet and outlet change over time.

This example was fairly straight-forward. The mathematical description was well formulated and the input file to gPROMS ran first time. This, however, is not that surprising as the basis for the model was an existing case that had no mention of problems running it in it's notes. This may not always be the case and is a topic for future work (see § 7.2.1). Problems with translation are likely to arise when retrieved case equations are changed. Even if it is only the values of some constants that are altered the solution given may not be correct, even if the original case solution was.

Figure 6.12. Tank hierarchy selection menu. Reactor_Vessel selected.

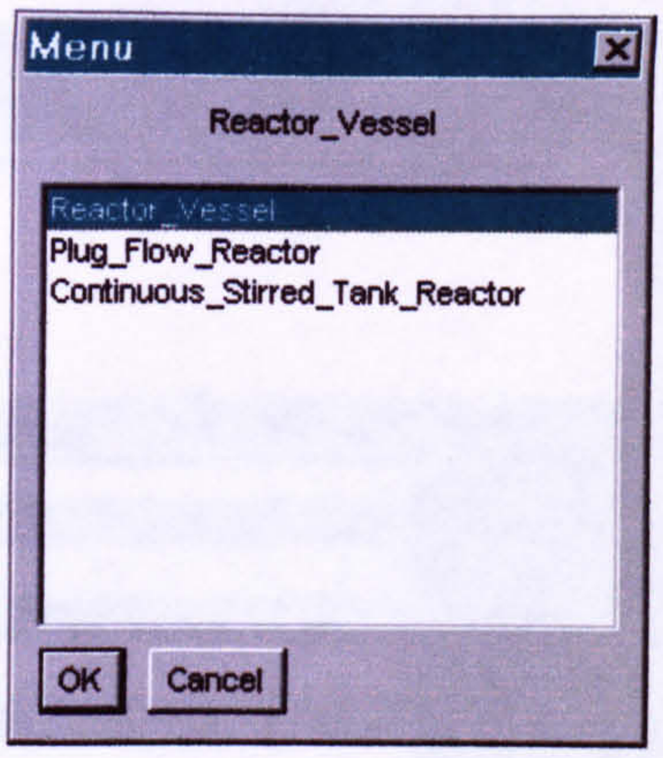


Figure 6.13. Reactor_Vessel hierarchy selection menu. Reactor_Vessel selected.

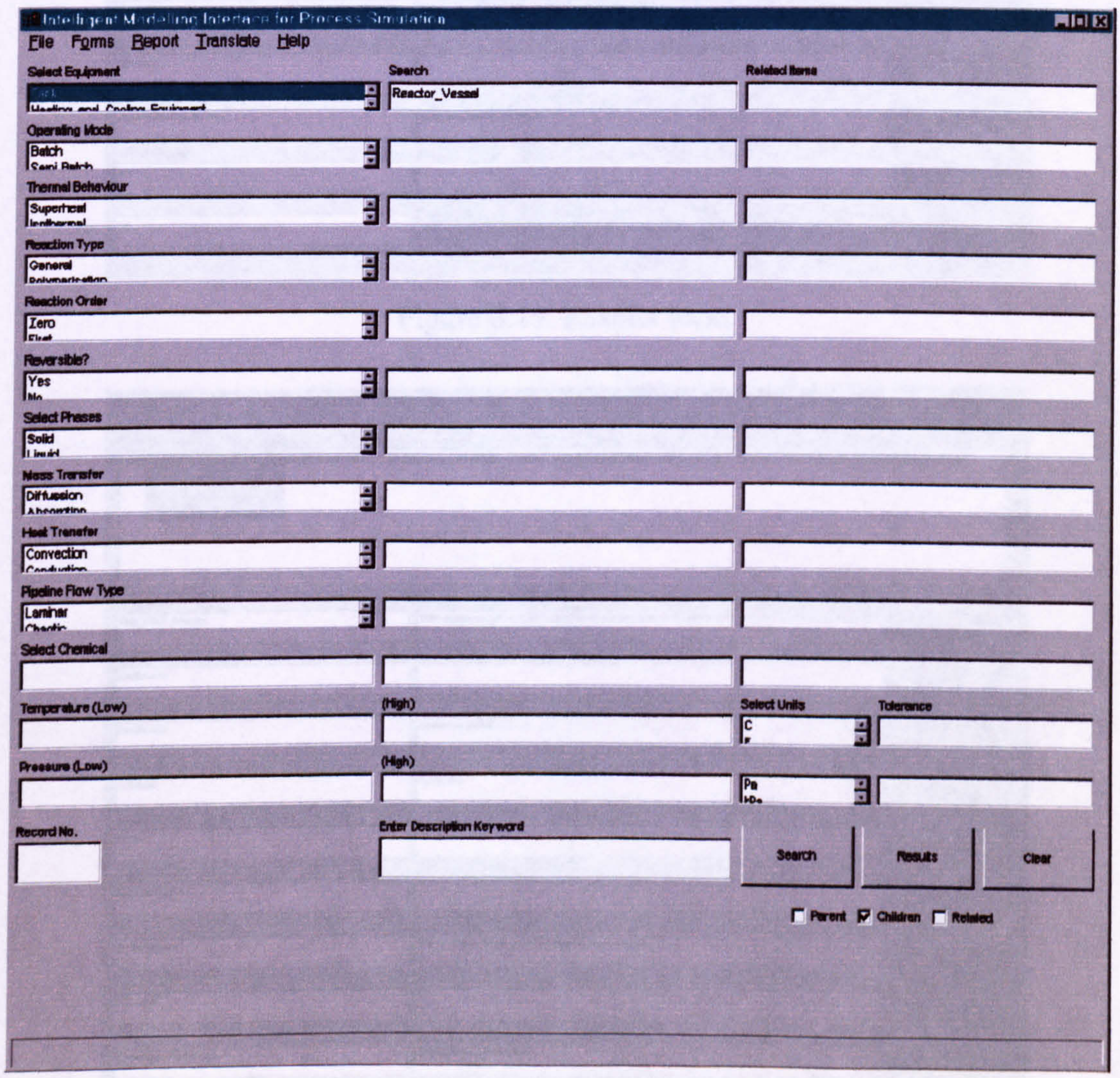


Figure 6.14. Completed search form.

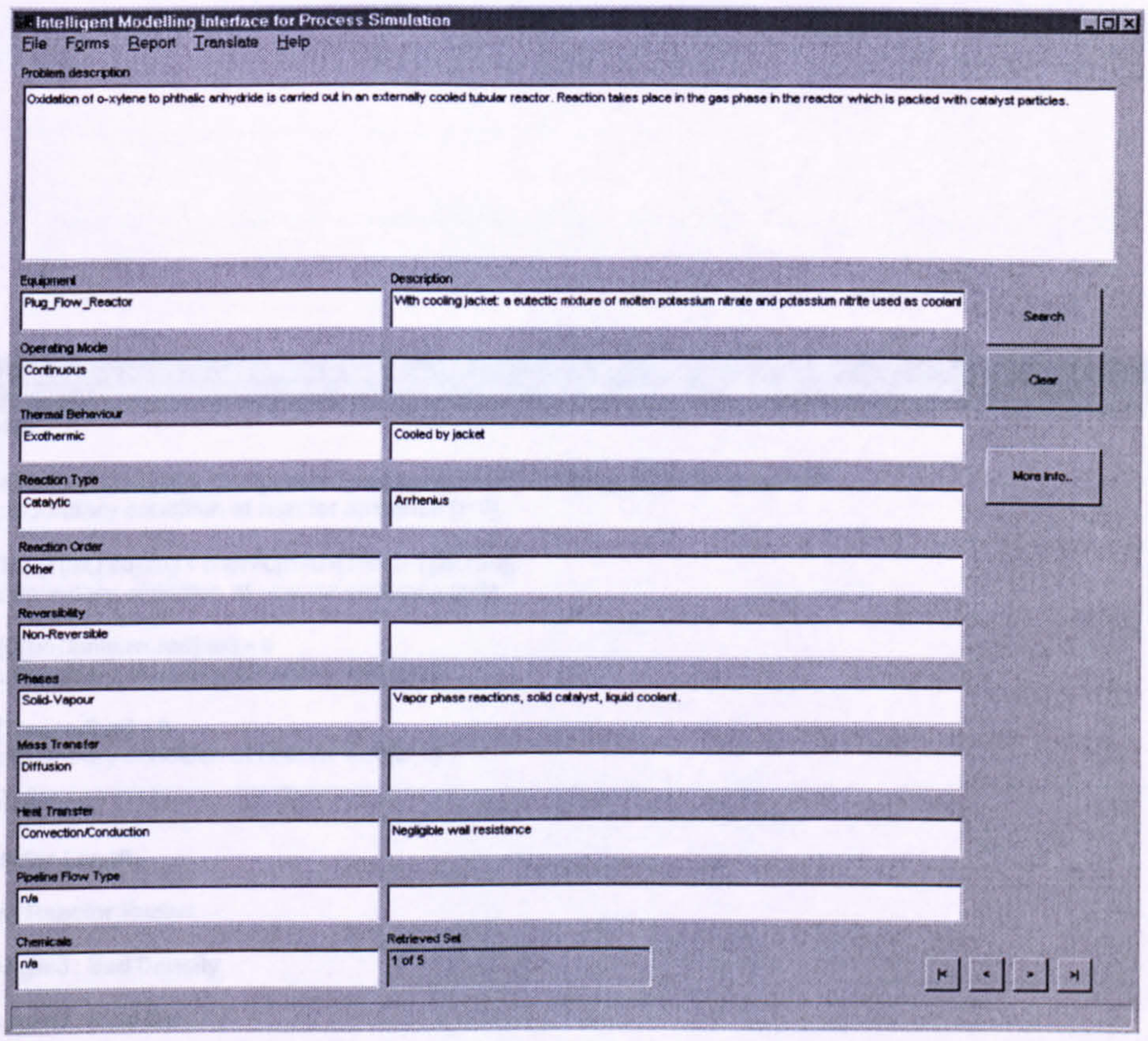


Figure 6.15. Results form.

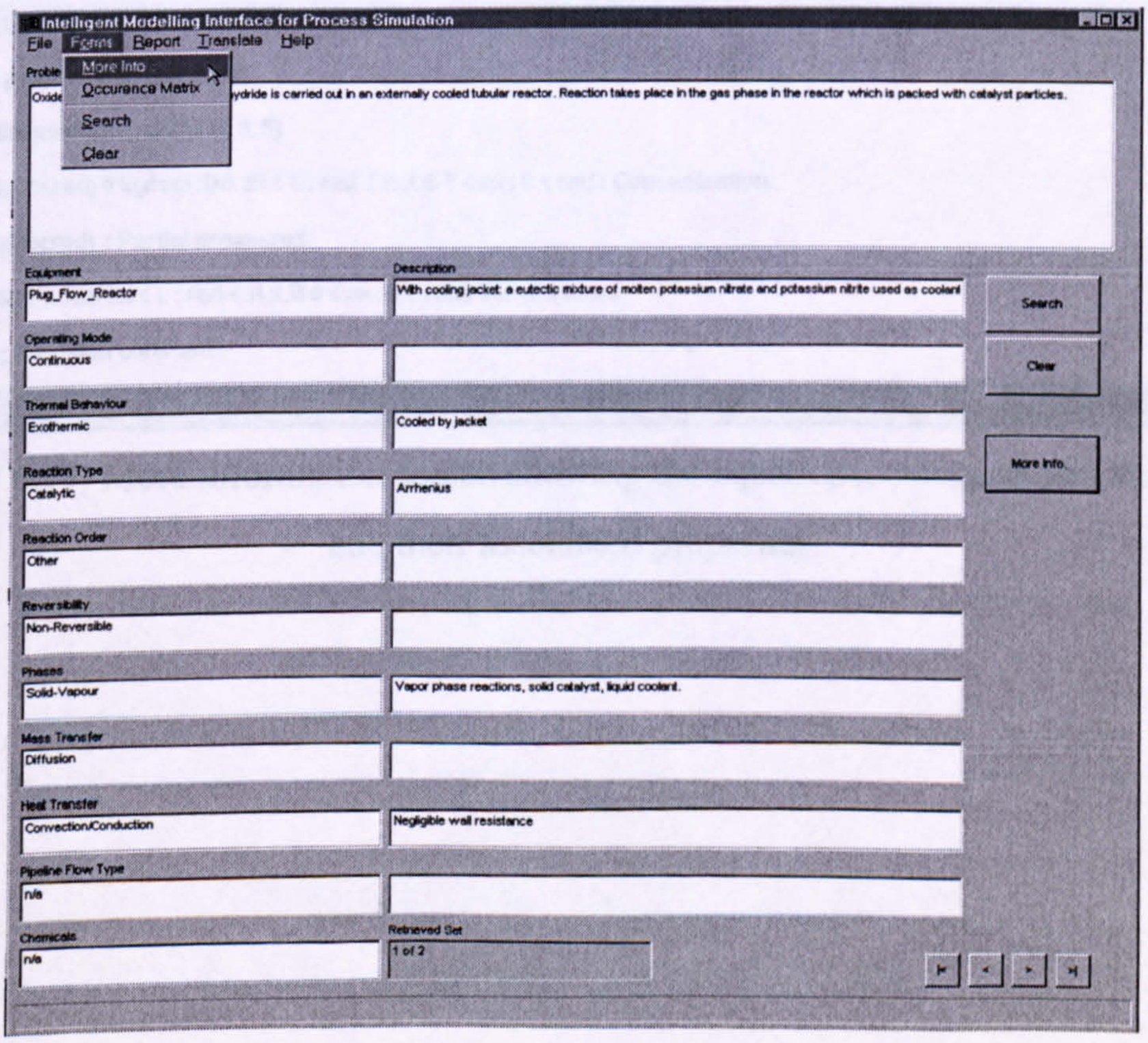


Figure 6.16. Selection menu for more information on this case.

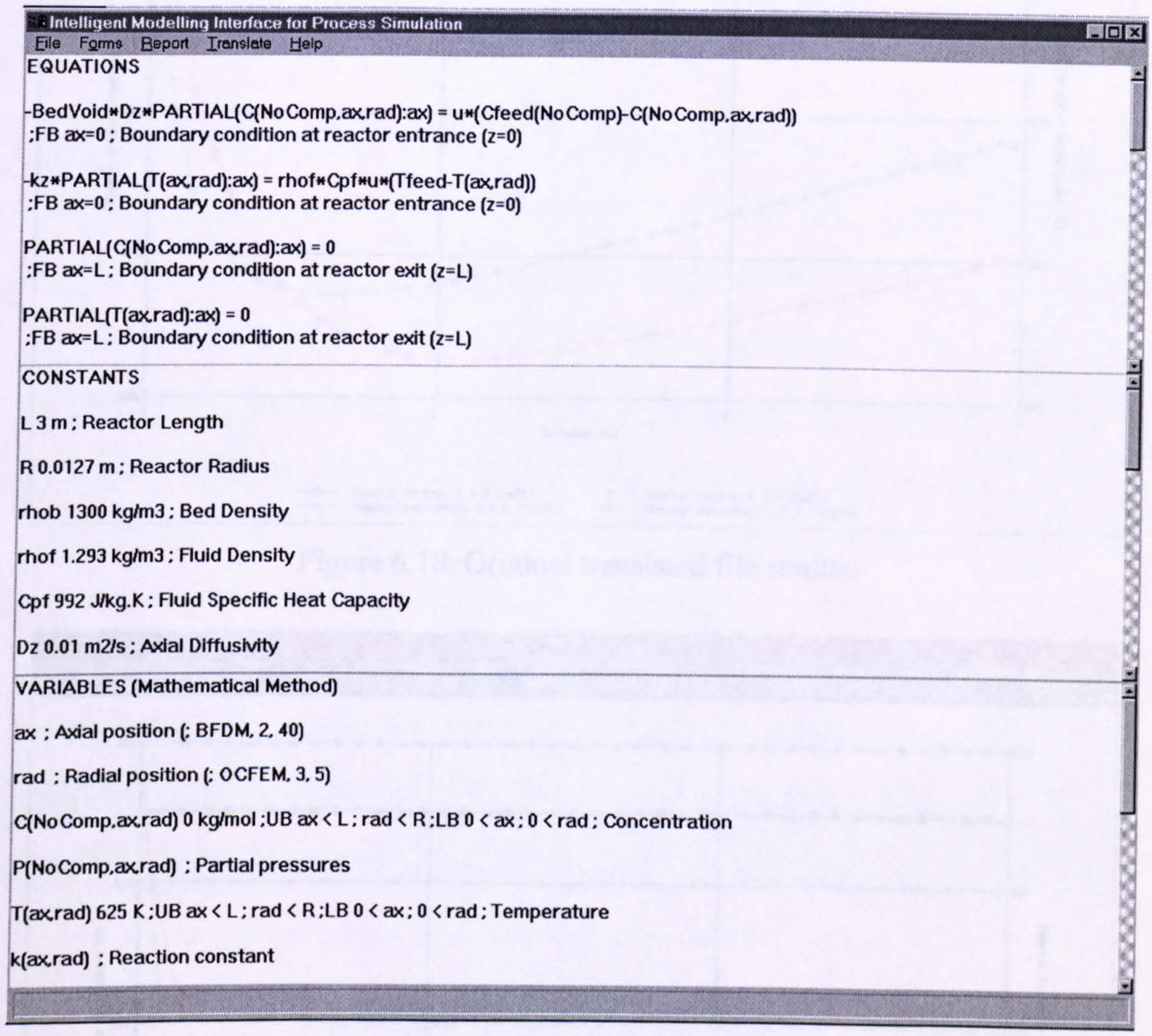


Figure 6.17. More information screen showing the equations, constants and variables and their associated properties.

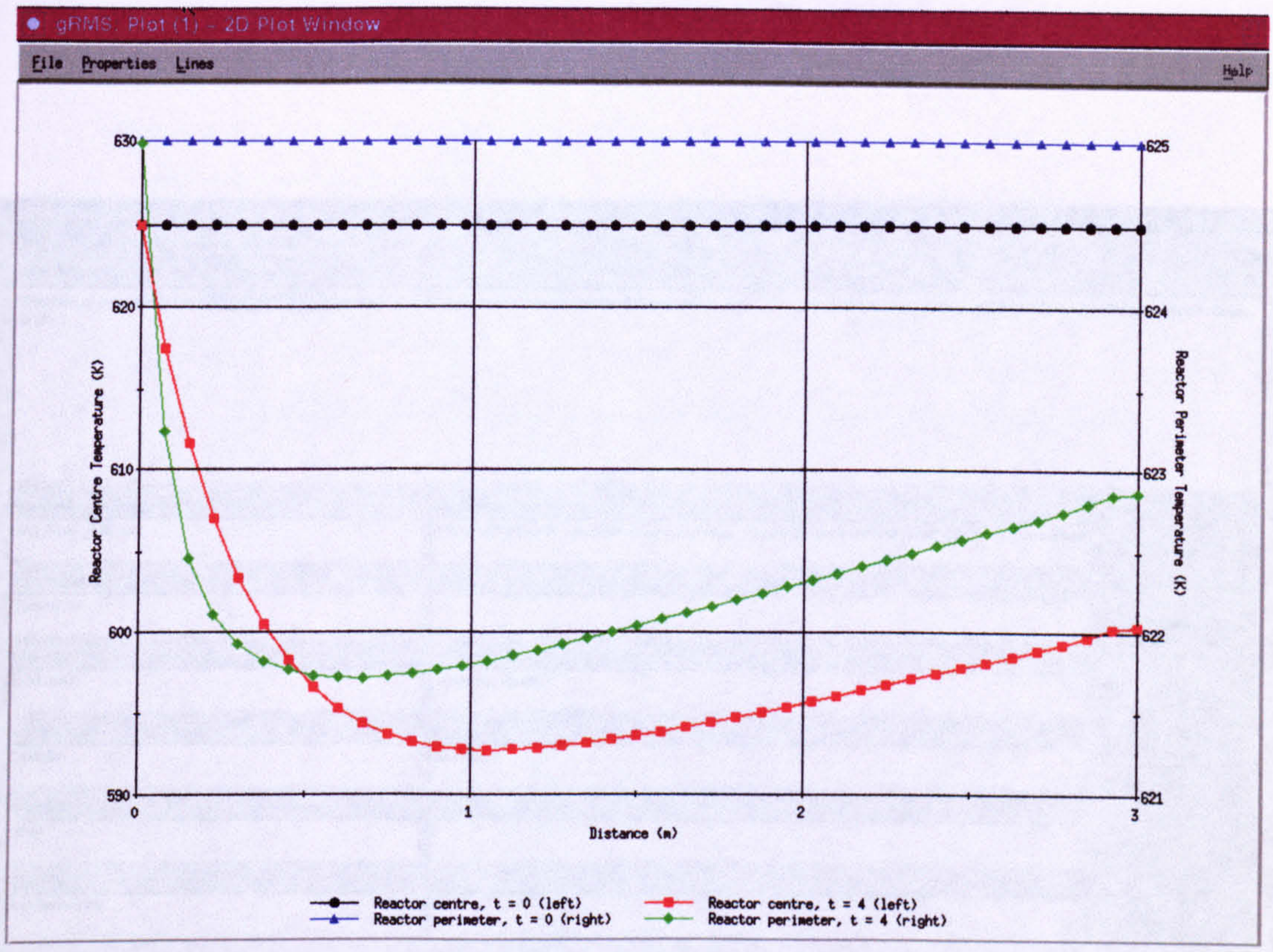


Figure 6.18. Original translated file results.

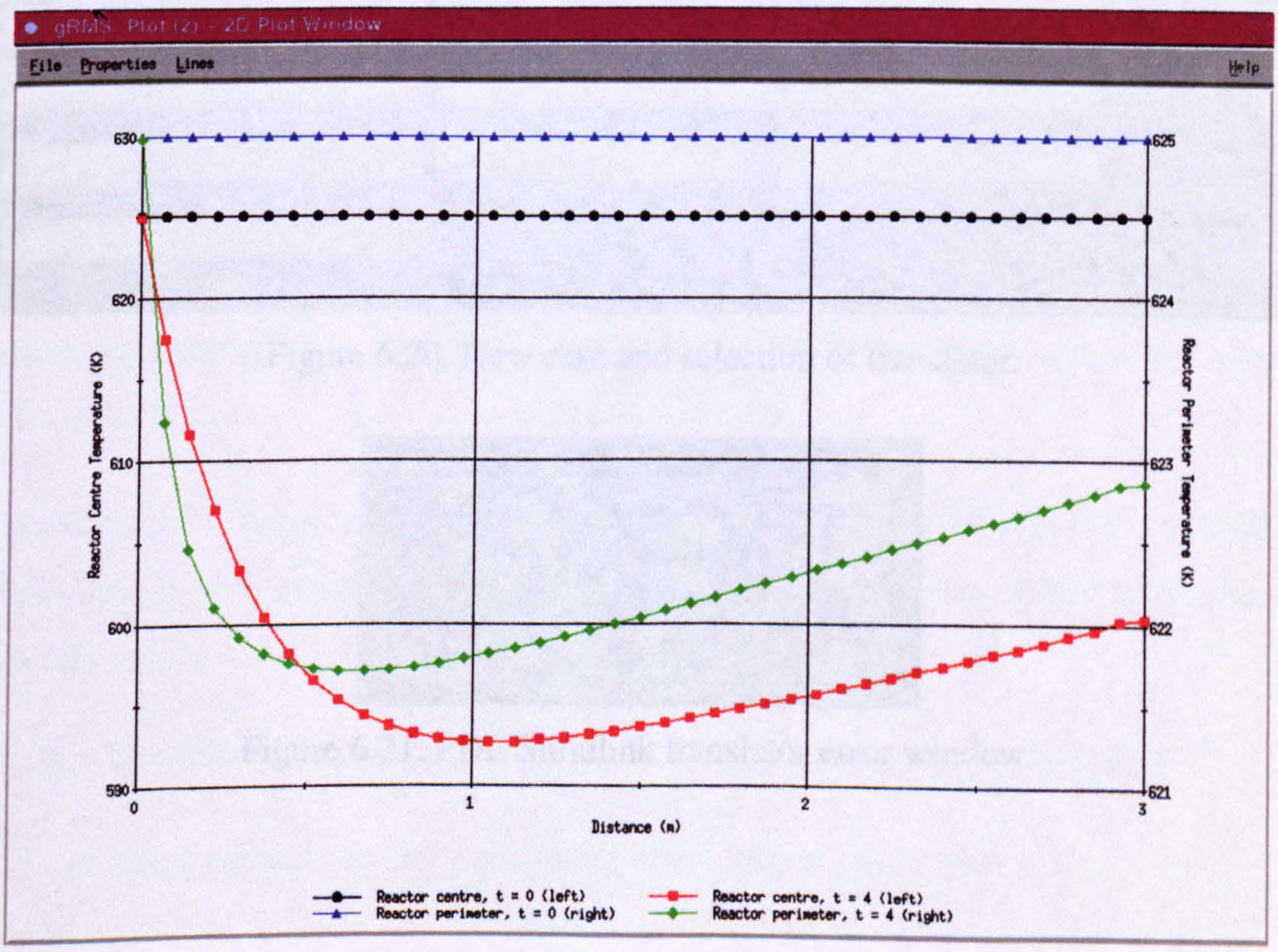


Figure 6.19. Original results from PSE (1998).

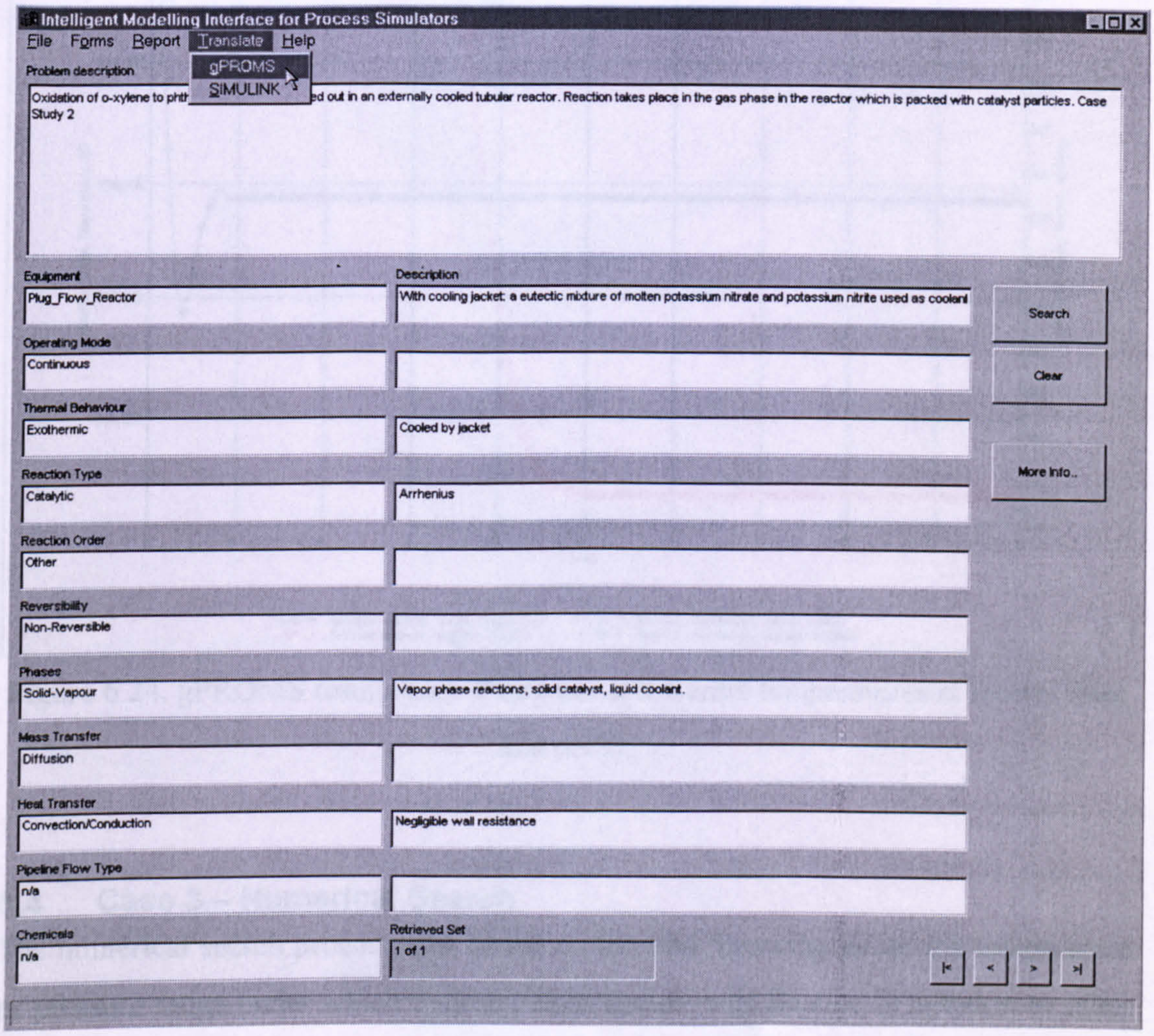


Figure 6.20. New case and selection of translator.

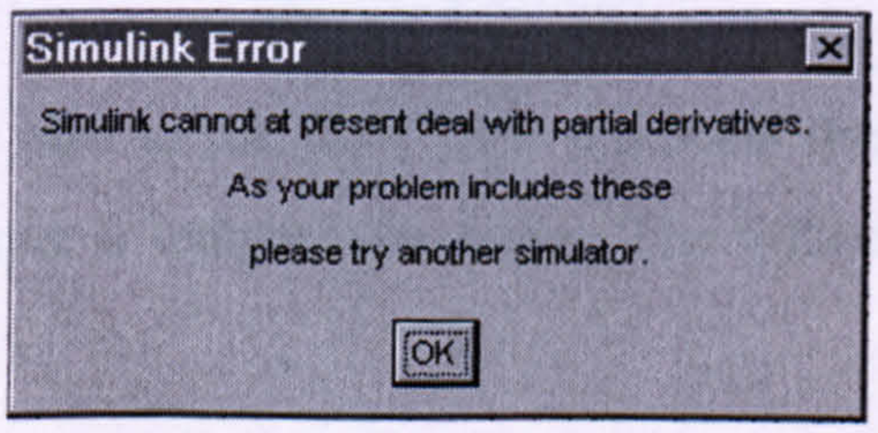


Figure 6.21. PDE Simulink translator error window.

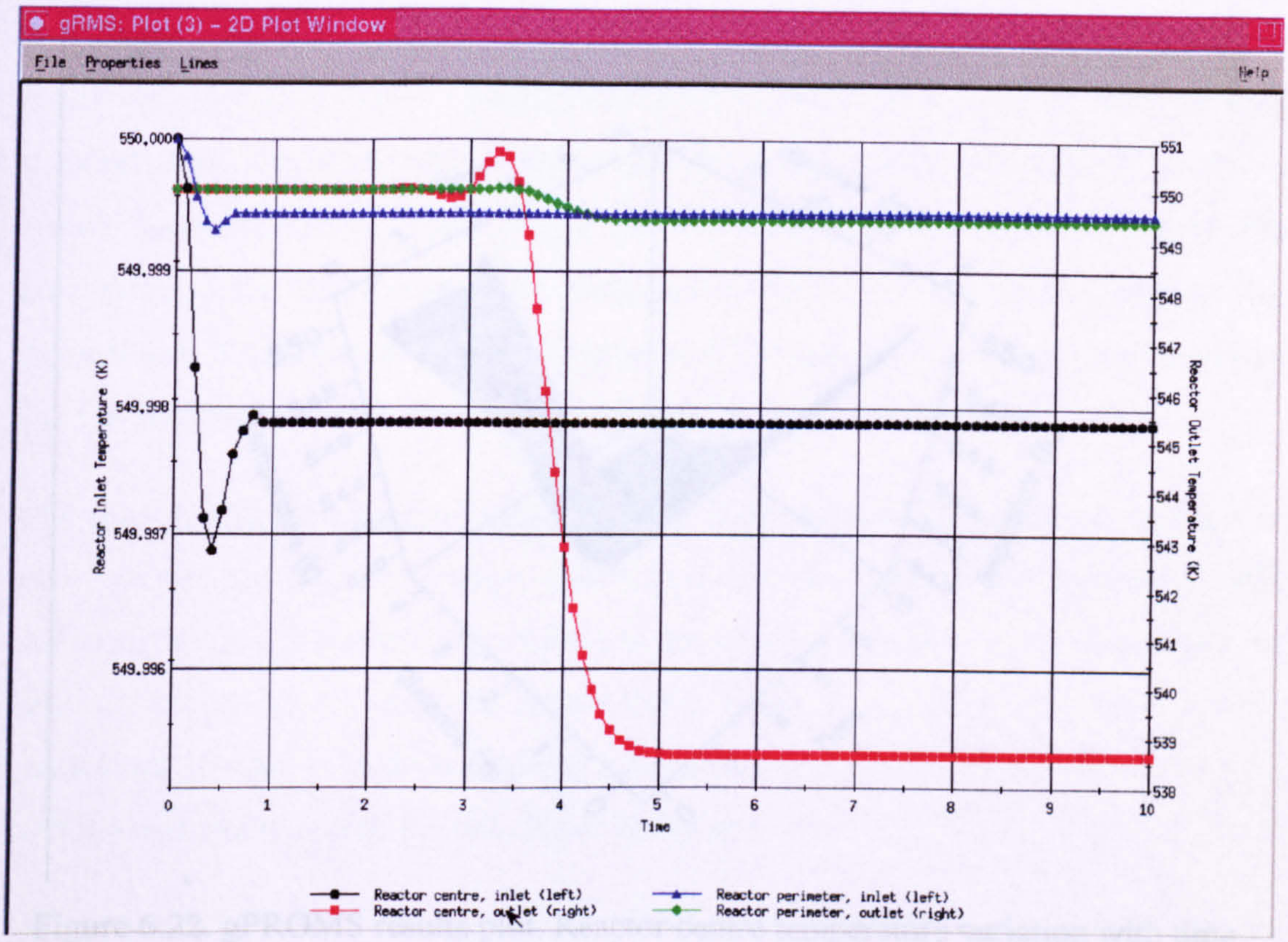


Figure 6.24. gPROMS results plot. Perimeter and Centre temperatures at reactor inlet and outlet.

6.4 Case 3 – Numerical Search

The numerical search procedure is based on the user knowing either the temperature or pressure range under which their problem system will operate in addition to other system information. Without this numerical information, the case search is the same as for case 2.

This case study looks at how to use IMIPS for a process that has specific operating conditions, in this example a temperature range. The following procedure outlines how this can be done.

1. Case Specification – writing the initial numerical search specification, §6.4.1.
2. Case Retrieval and Review – reviewing the retrieved cases, §6.4.2.
3. Case Adaptation – adapting the selected case, §6.4.3.
4. Case Translation – translating the adapted case into simulator input code, §6.4.4.

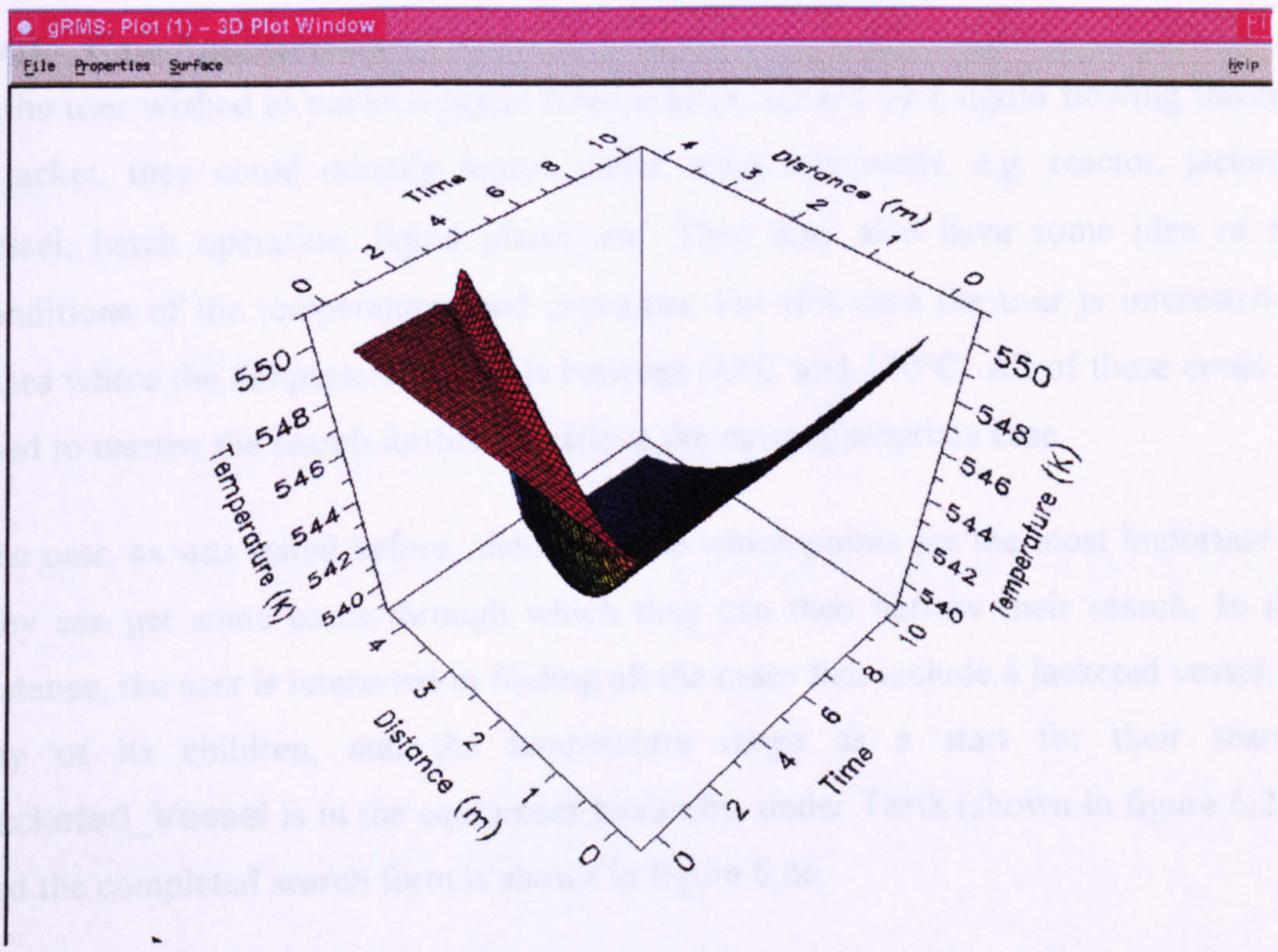


Figure 6.22. gPROMS results plot. Reactor centre temperature variation with time.

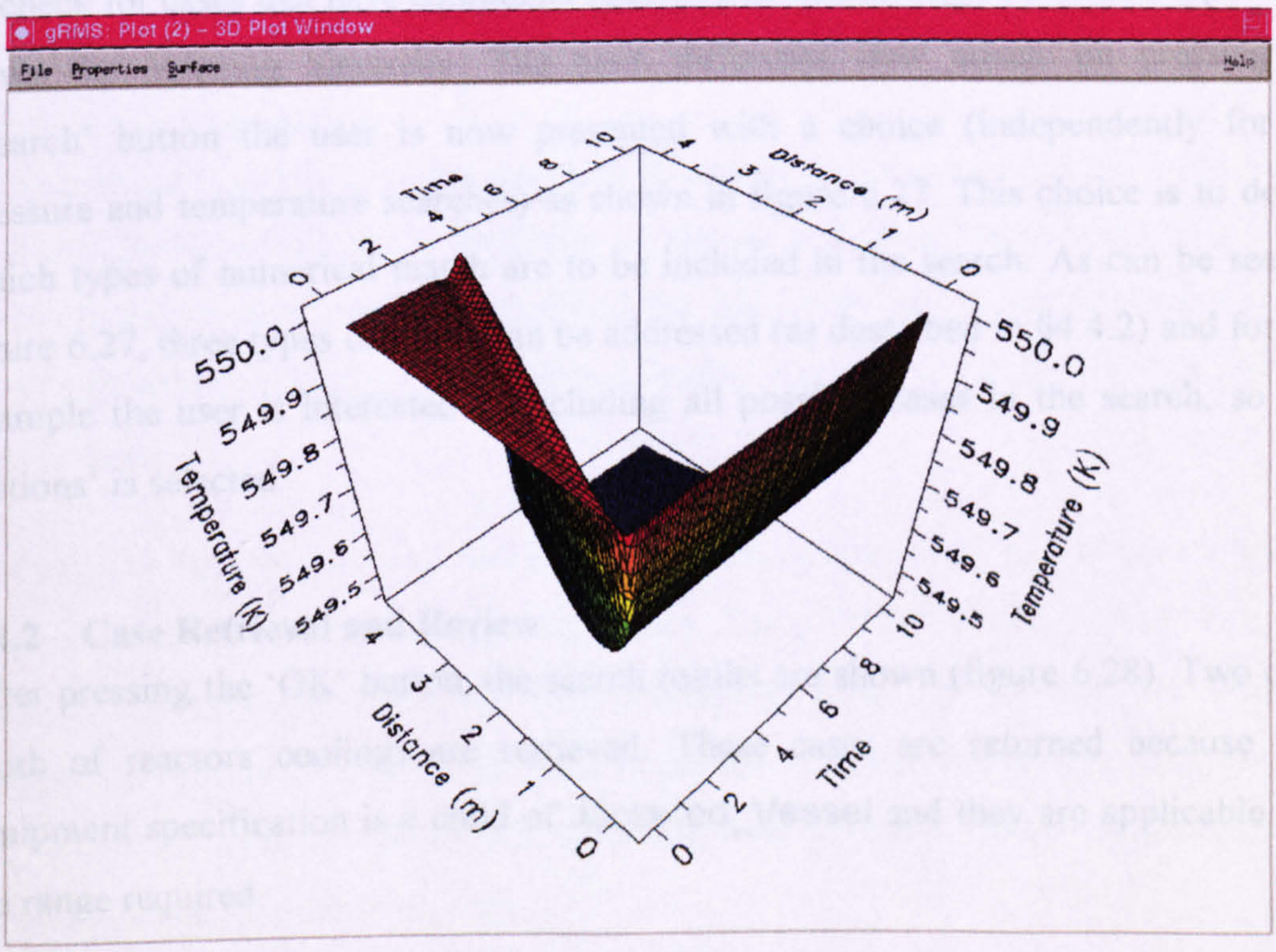


Figure 6.23. gPROMS results plot. Reactor perimeter temperature variation with time.

6.4.1 Case Specification

If the user wished to model a liquid filled reactor, cooled by a liquid flowing through a jacket, they could initially search under many keywords, e.g. reactor, jacketed vessel, batch operation, liquid phase, etc. They may also have some idea of the conditions of the temperatures and pressures. For this case the user is interested in cases where the temperature range is between 90°C and 120°C. All of these could be used to narrow the search further to retrieve the most appropriate case.

The user, as was stated before, should decide which points are the most important so they can get some cases through which they can then narrow their search. In this instance, the user is interested in finding all the cases that include a jacketed vessel, or any of its children, and the temperature range as a start for their search. Jacketed_Vessel is in the equipment hierarchy, under Tank (shown in figure 6.25.) and the completed search form is shown in figure 6.26.

As in case 2, the 'children' box is checked. So, again, the search query will be formed to check for cases that have equipment specified as Jacketed_Vessel or a type lower down the indexing hierarchy. The main difference now arises, on pressing the 'Search' button the user is now presented with a choice (independently for the pressure and temperature searches) as shown in figure 6.27. This choice is to decide which types of numerical match are to be included in the search. As can be seen in figure 6.27, three types of query can be addressed (as described in §4.4.2) and for this example the user is interested in including all possible cases in the search, so 'All Options' is selected.

6.4.2 Case Retrieval and Review

After pressing the 'OK' button, the search results are shown (figure 6.28). Two cases (both of reactors cooling) are retrieved. These cases are returned because their equipment specification is a child of Jacketed_Vessel and they are applicable over the range required.

6.4.3 Case Adaptation

These two cases can be reviewed and the user can then select the one that is most suitable for their needs. As in the second example, all manipulation of the selected case occurs in the database.

If a retrieved case is of use then the user may go back to the database, copy the retrieved case and then amend it for their specific parameters. In this example the second retrieved case is more suitable and will only need little alterations to be of use to solve the problem. The changes made to this case are:

- The reactor dimensions.
- The bulk fluid specific heat capacity.
- The temperatures of the coolant in and the initial temperatures of the coolant and bulk.
- The density of the coolant.

The different values used are shown in §6.4.4.

Once the user is satisfied with the new case (based on the retrieved case) they return to IMIPS, and retrieve their new case.

6.4.4 Case Translation

To ensure the translation of the new case is correct the original case can be translated and compared to the results produced in Simulink from a hand-written file (figure 6.30). If the retrieved case is translated the following graph can be plotted (figure 6.29) and when compared to the original results (figure 6.31) it can be seen that they are the same.

The case is then recalled and, once selected, can be translated into either or both (at present) of the modelling package input files. The gPROMS and Simulink input files created, automatically, from the case description input by the user are shown in Appendices V.4 and V.5. For this case, again, the equations required no changes, only the values of some constants and variables were changed. These amendments are (gPROMS input file code shown):

Originally Retrieved Case

```
SET
  WITHIN A101 DO
    Vol     := 2; # m3
    Cvb     := 4000; # kJ/m3.K
    Cvc     := 4000; # kJ/m3.K
    Volc     := 0.025; # m3
    Area     := 1.7; # m2
    U     := 1; # kW/m2.K
    Tcin     := 20; # C
    q     := 0.3; # m3/min
    rhoc     := 1000; # kg/m3
    rhob     := 1000; # kg/m3
  END # Within
INITIAL
  WITHIN A101 DO
    Tc      = 20; #
    T      = 80; #
  END # Within
```

New Case

```
SET
  WITHIN A101 DO
    Vol     := 0.28; # m3
    Cvb     := 4000; # kJ/m3.K
    Cvc     := 6000; # kJ/m3.K
    Volc     := 0.049; # m3
    Area     := 1.88; # m2
    U     := 1; # kW/m2.K
    Tcin     := 5; # C
    q     := 0.5; # m3/min
    rhoc     := 800; # kg/m3
    rhob     := 1000; # kg/m3
  END # Within
INITIAL
  WITHIN A101 DO
    Tc      = 5; #
    T      = 120; #
  END # Within
```

After running the translated file in gPROMS the following graphs can be plotted. Figure 6.32 shows how the temperatures of the bulk fluid and the coolant vary over time.

The Simulink file as opened in Simulink is shown in figure 6.33 and the results are shown in figure 6.34.

Although these results look different to those produced by gPROMS, they are actually the same. gPROMS plots are drawn at the intervals specified in the user input, and so some fluctuations in the values may not be picked up by the plot. In this case the initial rapid temperature rise (as shown in the Simulink plot) is not picked up in the gPROMS plot, thus the temperature appears to not rise to the same value before declining.

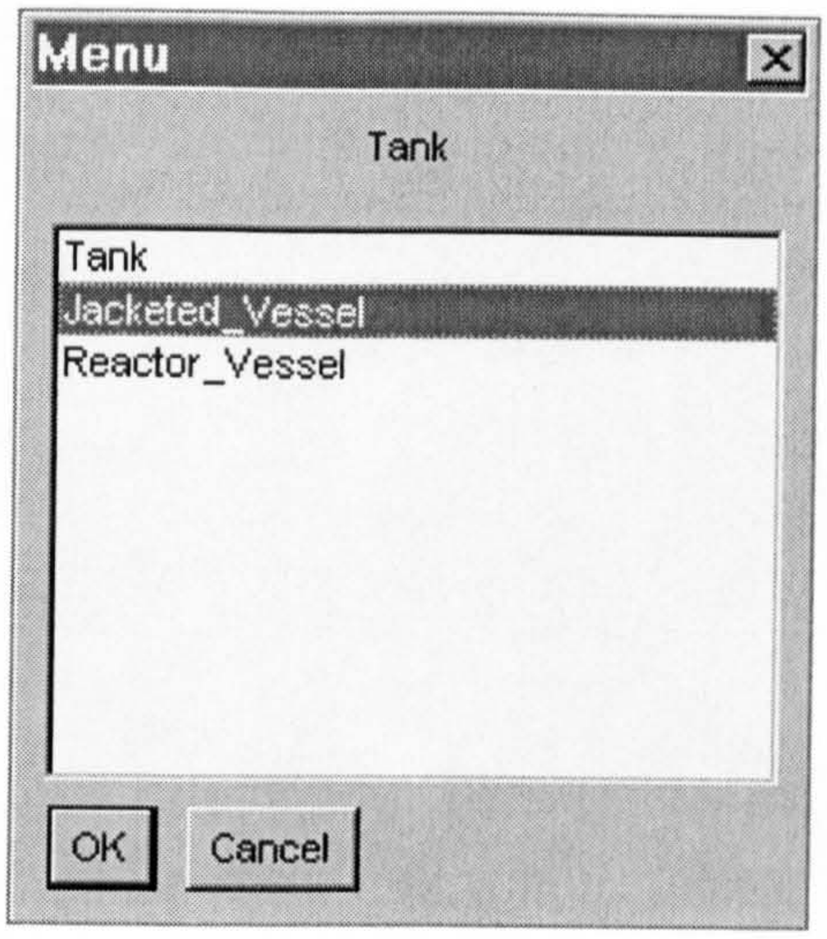


Figure 6.25. Tank hierarchy selection menu. Reactor_Vessel selected.

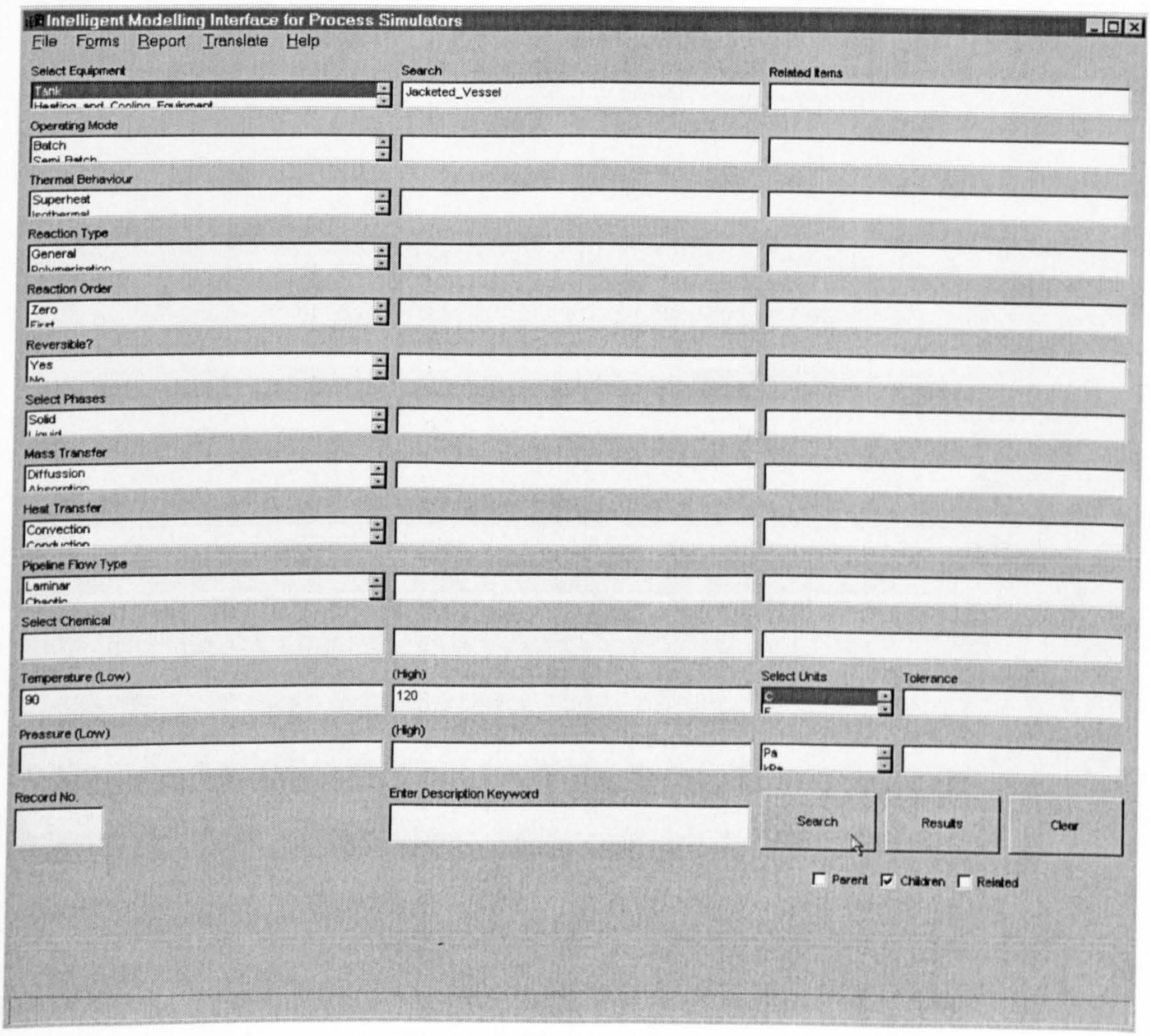


Figure 6.26. Completed search form.

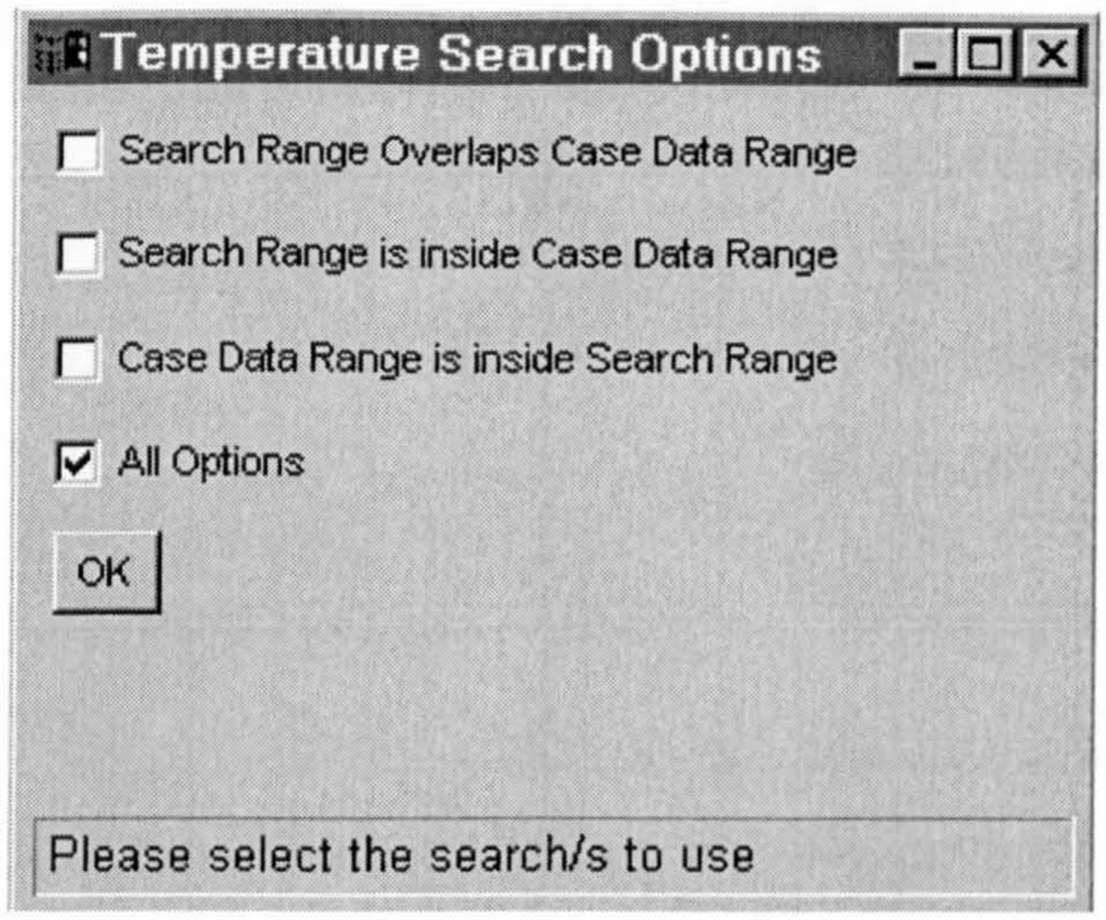


Figure 6.27. Numerical search user options.

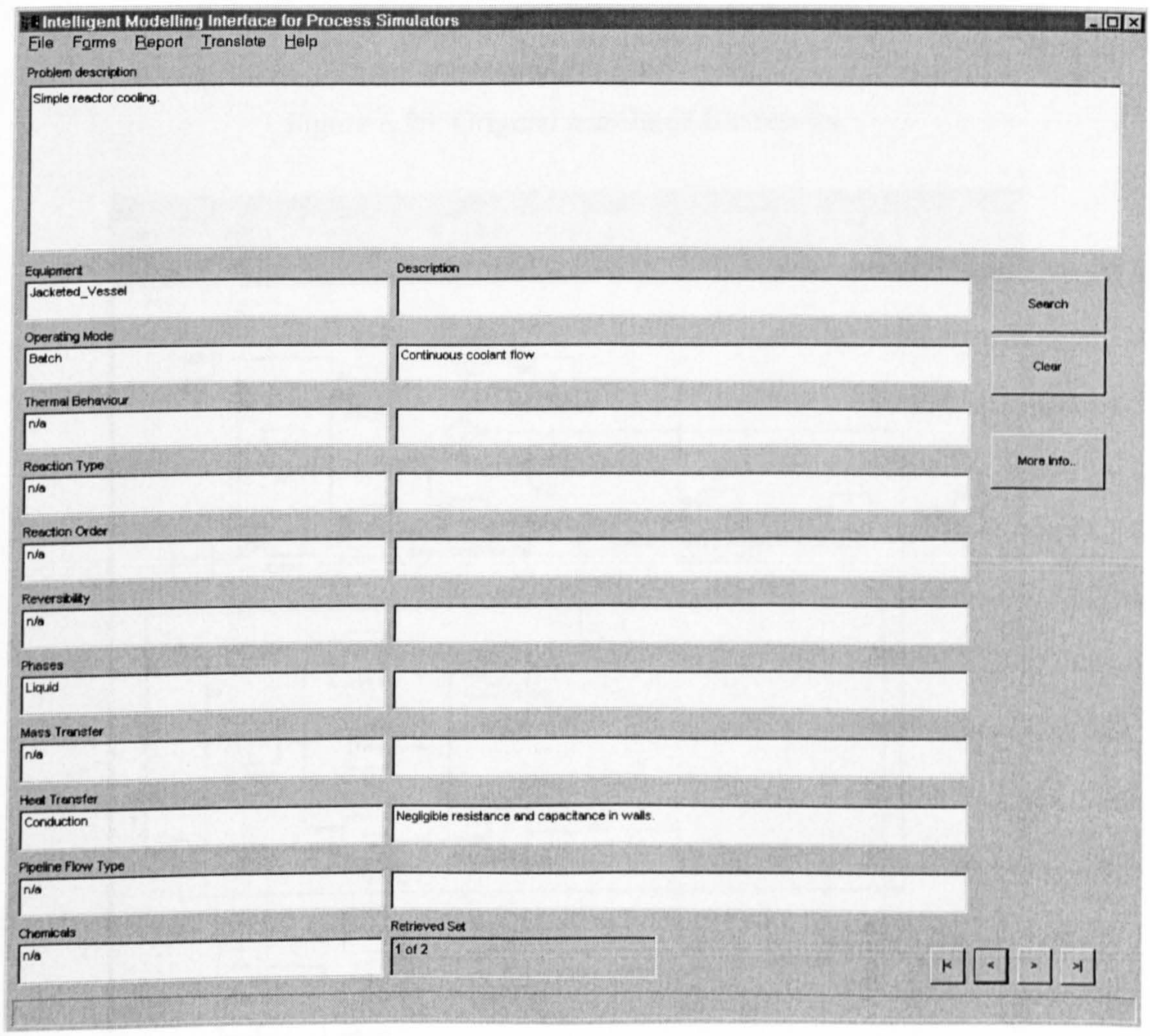


Figure 6.28. Results form.

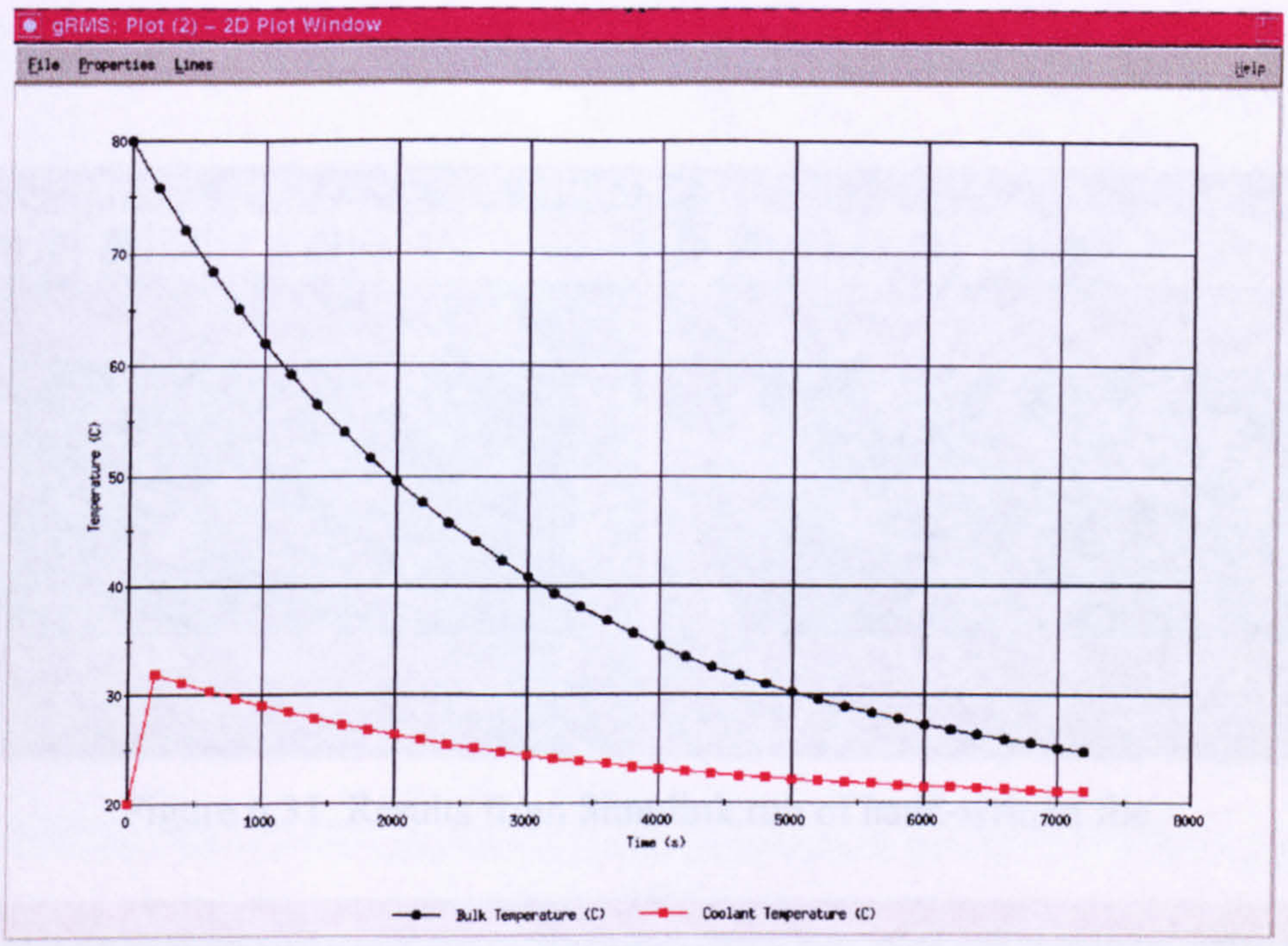


Figure 6.29. Original translated file results.

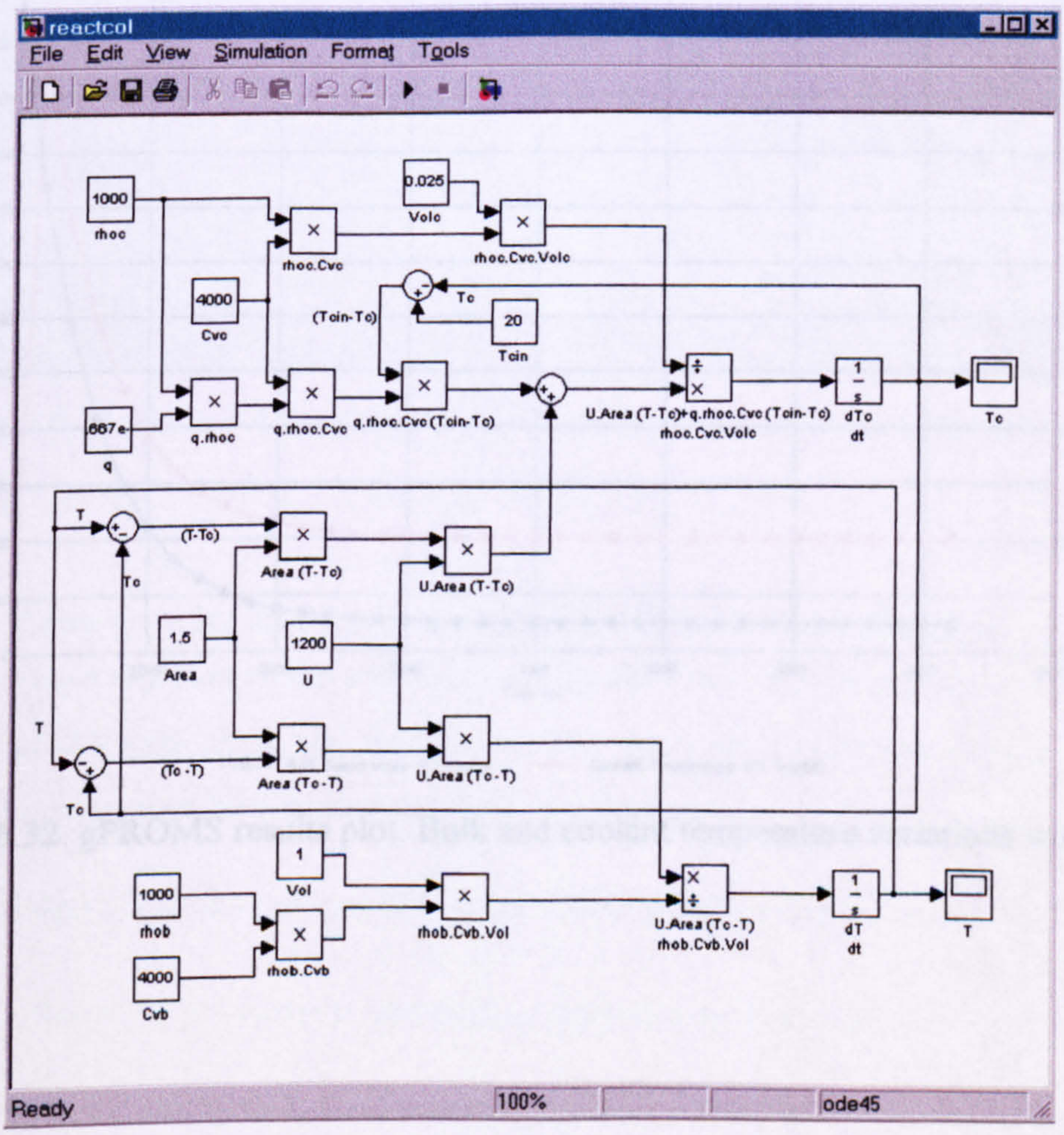


Figure 6.30. Simulink hand-written file.

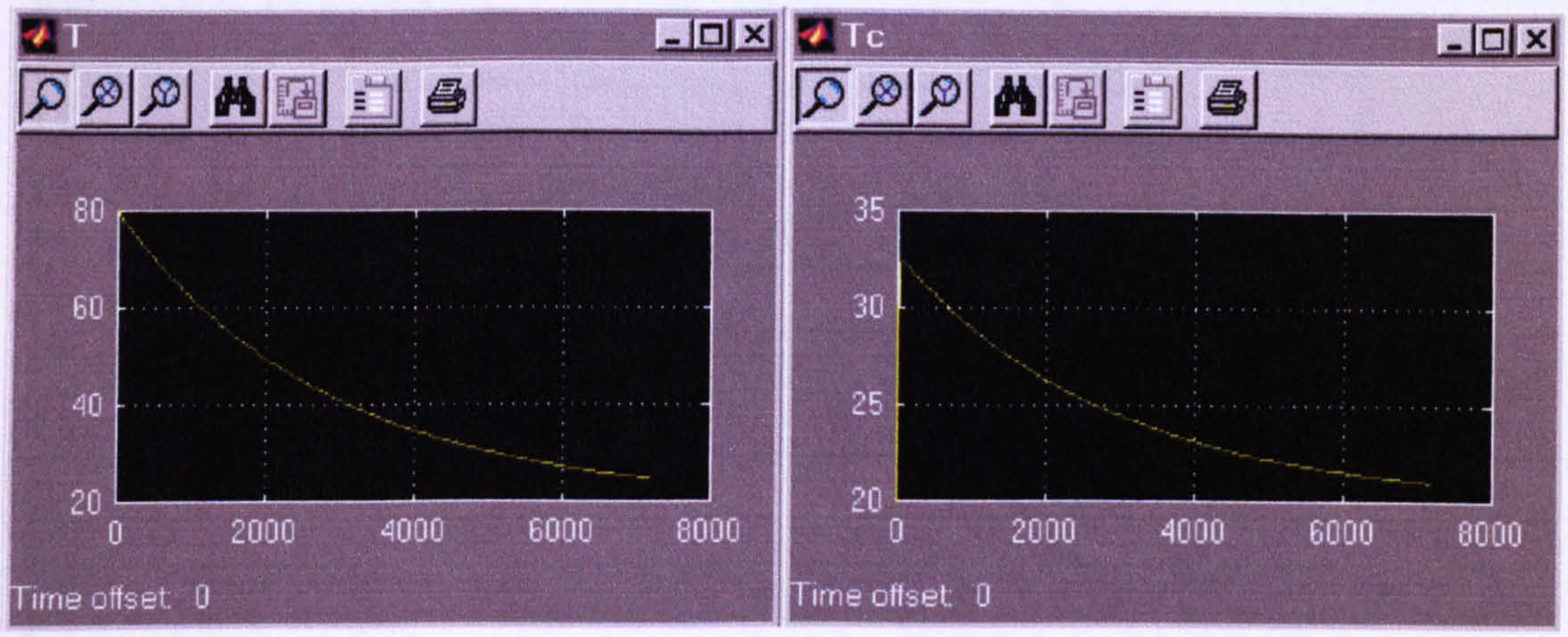


Figure 6.31. Results from Simulink run of hand-written file.

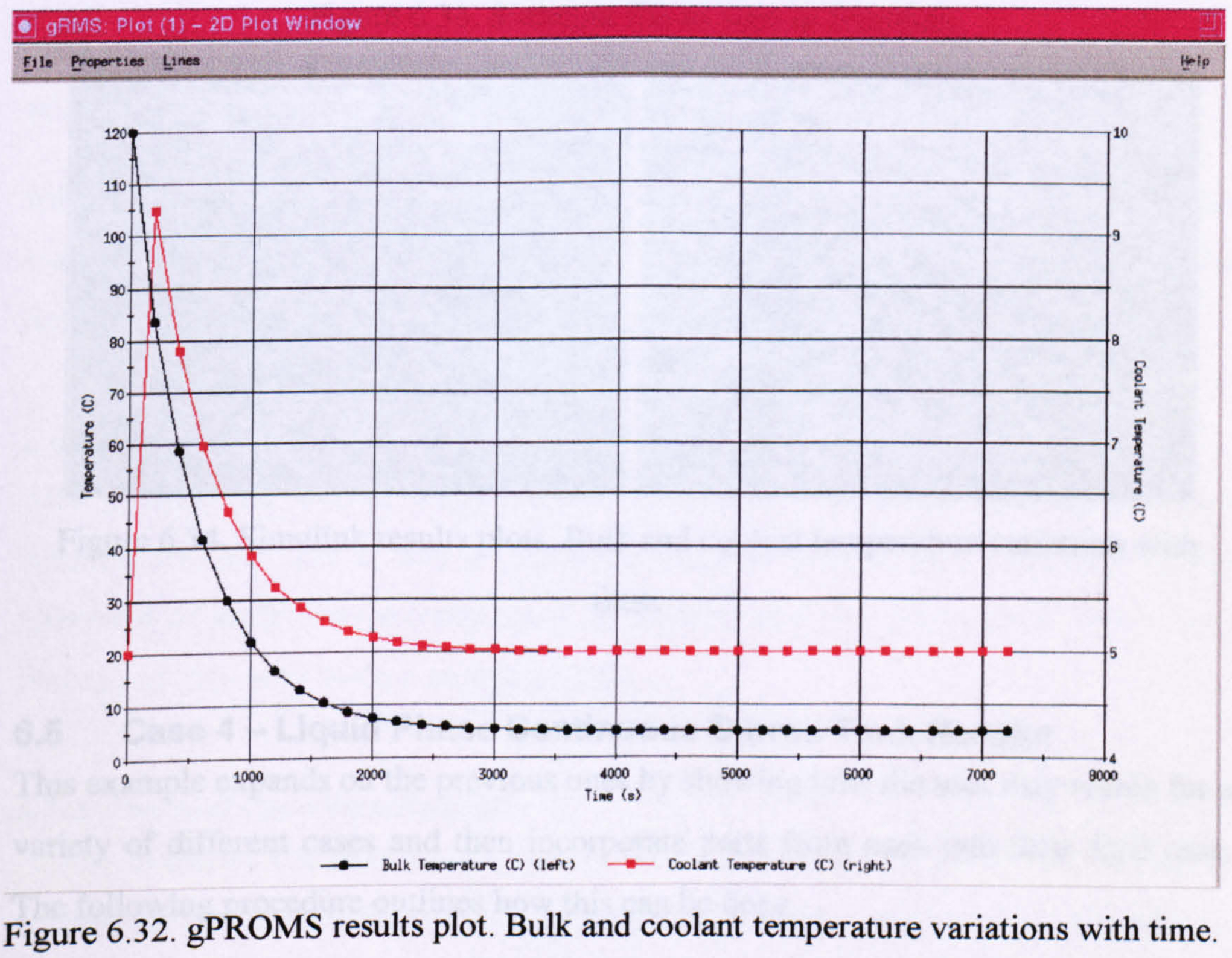


Figure 6.32. gPROMS results plot. Bulk and coolant temperature variations with time.

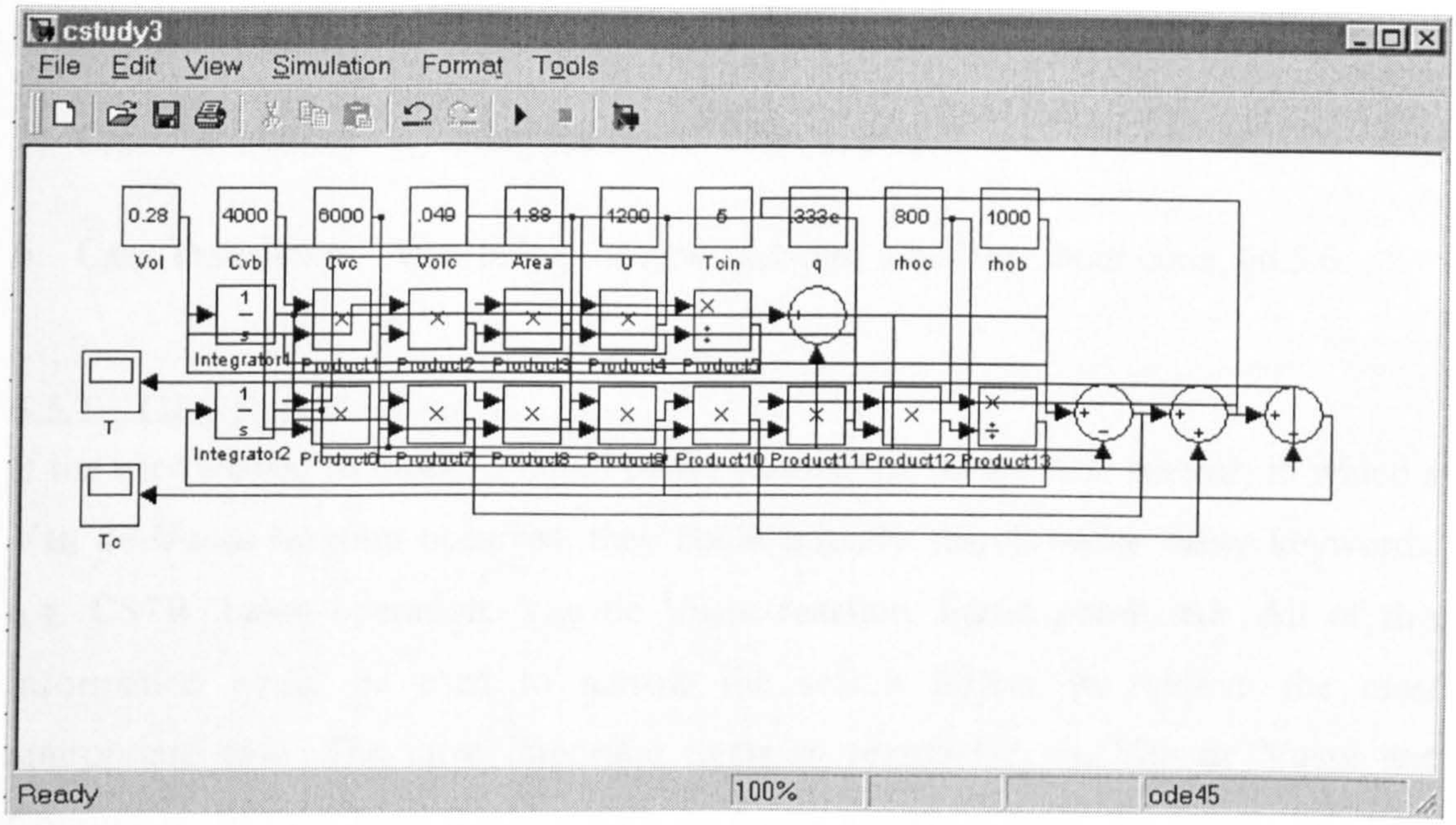


Figure 6.33. Simulink file as seen in Simulink.

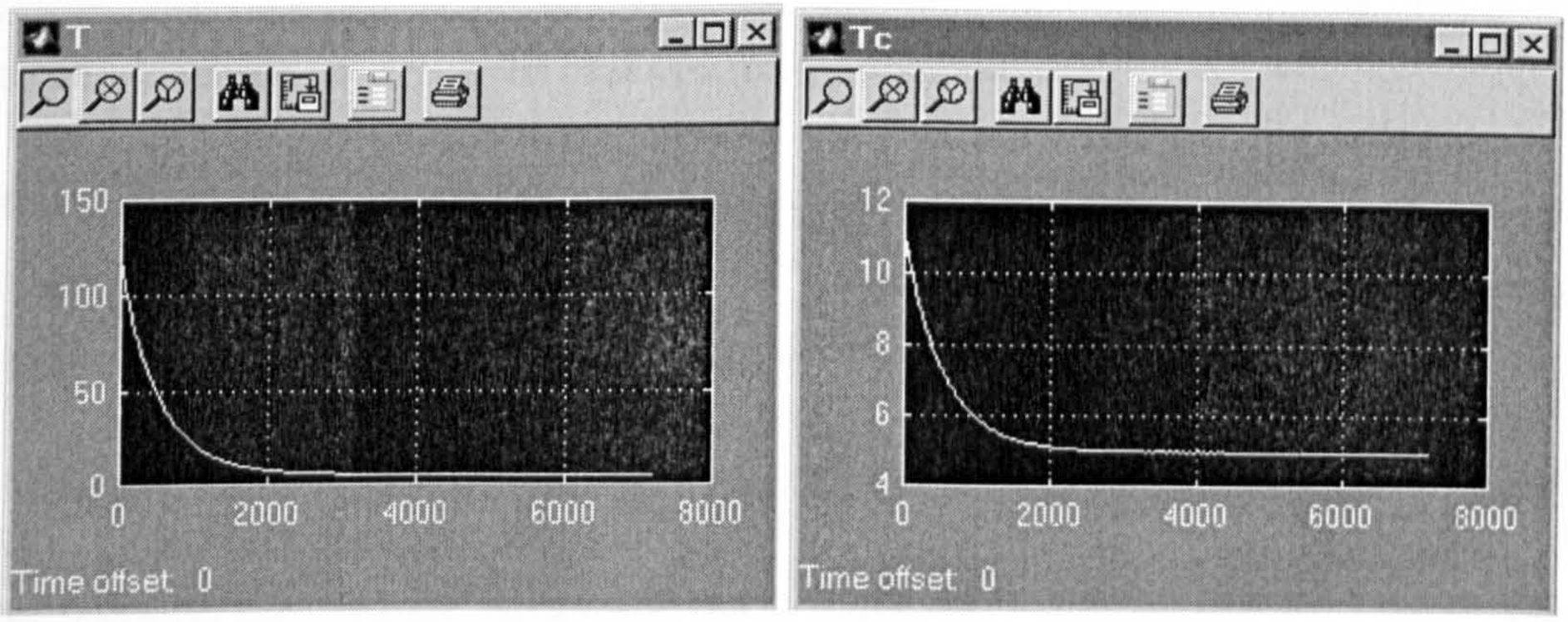


Figure 6.34. Simulink results plots. Bulk and coolant temperature variations with time.

6.5 Case 4 – Liquid Phase Continuous Stirred Tank Reactor

This example expands on the previous ones by showing how the user may search for a variety of different cases and then incorporate parts from each into their final case. The following procedure outlines how this can be done.

1. Case Specification – writing the initial search specification, §6.5.1.
2. Case Retrieval and Review – reviewing the retrieved cases, §6.5.2.
3. Secondary Case Specification – writing the second search specification, §6.5.3.
4. Secondary Case Retrieval and Review – reviewing the retrieved cases, §6.5.4.

4. Secondary Case Retrieval and Review – reviewing the retrieved cases, §6.5.4.

5. Case Adaptation – adapting the selected cases, §6.5.5.

6. Case Translation – translating the new case into simulator input code, §6.5.6.

6.5.1 Case Specification

If the user wished to model a liquid phase continuous stirred tank reactor, in which a Van de Vusse reaction occurred, they could initially search under many keywords, e.g. CSTR, batch operation, Van de Vusse reaction, liquid phase, etc. All of this information could be used to narrow the search further to retrieve the most appropriate case. The most important items to search for are Van de Vusse and Continuous Stirred Tank Reactor with its outlet not at its base. To search for these together may recall some cases, but it may be easier to search for each individually and then combine the retrieved cases to form a solution.

Initially, the user is interested in finding all the cases that include a Van de Vusse reaction as a start for their search. Van de Vusse reaction is in the reaction hierarchy and selecting this creates the following search form (figure 6.35).

6.5.2 Case Retrieval and Review

After pressing the 'Search' button the result is shown, 1 case includes a Van de Vusse reaction and this case is also a Continuous Stirred Tank Reactor. The user is still interested to see if there are any cases of reactors with the outlet off the base of the reactor, so they need to do a second search.

6.5.3 Second Case Specification

In this example, the user is also interested in finding all the cases that include a continuous stirred tank reactor with its outlet off the base of the reactor. Continuous stirred tank reactor is in the equipment hierarchy, under Tank and Reactor_Vessel. It can be seen (figure 6.36) that the 'children' box is not checked, as the user only wishes to see cases that include Continuous Stirred Tank Reactors.

(figure 6.37). This case does not use a Van de Vusse reaction though so the two cases must be combined to solve the problem.

The first of the two suitable cases is from PSE (1998) and so the translation for this can be compared to the original results to ensure the translation procedure is correct. If the retrieved case is translated the following graph can be plotted (figure 6.38) and when compared to the original results (figure 6.39) it can be seen that they are the same.

6.5.5 Case Adaptation

The two cases can now be called up in the database and the relevant equations copied to form the new case description. The basis for this new case is the Continuous Stirred Tank Reactor case with the Van de Vusse reaction. The major change is the inclusion of a conditional equation (from the other case) to allow for the outlet to be off the base of the reactor. This and other changes made to these cases are:

- The equations from each case were placed into the new case.
- The stoichiometric coefficients and reaction orders.
- The dimensions of the reactor vessel.
- The initial concentrations within the reactor.
- The feed stream concentrations.

The different values used are shown in §6.5.6.

Once the user is satisfied with the new case (based on the retrieved cases) they return to IMIPS, and retrieve their new case.

6.5.6 Case Translation

At present, as was outlined towards the end of §5.6.2, Simulink can not deal with multi value (matrix) constant arrays. The translators, also, cannot, at present, convert them into code that Simulink can solve, therefore, this case can only be translated by one of the translators. If the user tried to translate this case into a Simulink file, an error message would pop up indicating this shortcoming (figure 6.40).

The case is then recalled and, once selected, can be translated into either or both (at present) of the modelling package input files.

The gPROMS input file created, automatically, from the case description input by the user is shown in Appendix V.6. For this case the equations retrieved required no amendments to be suitable for the problem, they just needed putting in the same case. Also, the values of some constants and variables were changed. These amendments are (sections from gPROMS input file code shown):

Original CSTR case, tank outlet off base, not Van de Vusse reaction

```
  # Calculation of liquid level
  TotalVolume=CrossSectionalArea*Height;
  # Calculation of flowrate out
  IF Height > Hp  THEN
  Fout = ValveConstant*(Height-Hp) ;
  ELSE
  Fout = 0;
  END # If

SET
  WITHIN R101 DO
    ReactionConstant    := [8E-5,1E-5]; # m3/kmol.s
    Order    := [1,0,1,0,0,1,0,1]; #
    NU    := [-1,1,-1,1,1,-1,1,-1]; #
  END # Within

ASSIGN
  WITHIN R101 DO
    Xin    := [0.5,0.5,0,0]; #
  END # Within

INITIAL
  WITHIN R101 DO
    Xout(2)      = 2*Xout(1); #
    Xout(3)      = 0; #
    Xout(4)      = 0; #
    TotalHoldup      = 10; #
  END # Within
```

Original CSTR case, tank outlet at base, Van de Vusse reaction

```
  # Calculation of liquid level
  TotalVolume=CrossSectionalArea*Height;
  # Calculation of flowrate out
  Fout = ValveConstant*(Height-Hp) ;

SET
  WITHIN R101 DO
    ReactionConstant    := [8E-5,1E-4,1E-2]; # m3/kmol.s
    Order    := [1,0,1,0,1,0,0,0,0,0,0,0]; #
    NU    := [-1,0,-2,1,-1,0,0,1,0,0,0,1]; #
  END # Within

ASSIGN
  WITHIN R101 DO
    Xin    := [0.5,0.1,0.3,0.1]; #
  END # Within

INITIAL
  WITHIN R101 DO
    Xout(2)      = 2*Xout(1); #
    Xout(3)      = 0; #
    Xout(4)      = 0; #
    TotalHoldup      = 10; #
  END # Within
```

New Case

```
PARAMETER
  Pi          AS REAL # Pi

  # Calculation of liquid level
  TotalVolume=Pi*Diameter^2*Height/4;
  # Calculation of flowrate out
  IF Height > Hp  THEN
  Fout = ValveConstant*(Height-Hp) ;
  ELSE
  Fout = 0;
  END # If

SET
  WITHIN R101 DO
    ReactionConstant    := [8E-3,1.3E-2,1E-2]; # m3/kmol.s
    Order    := [1,0,1,0,1,0,0,0,0,0,0,0]; #
    NU    := [-1,0,-2,1,-1,0,0,1,0,0,0,1]; #
    Pi    := 3.1415926; #
  END # Within

ASSIGN
  WITHIN R101 DO
    Xin    := [0.5,0.1,0.3,0.1]; #
  END # Within

INITIAL
  WITHIN CS4 DO
    Xout(1)      = 0.5; #
    Xout(2)      = 0.1; #
    Xout(3)      = 0.3; #
    TotalHoldup      = 10; #
  END # Within
```

After running the translated file in gPROMS the following graphs can be plotted. Figure 6.41 shows the varying concentrations of the four components and the liquid level within the vessel.

This case outlines the other drawback with the use of Simulink. The translator cannot deal with converting multi value constant arrays (matrices) into the relevant Simulink code. As Simulink cannot have constants with values stored in a matrix these problems must be decomposed into arrays (vectors) of constants. This, in turn, means that the equations present that use the matrices must be duplicated for each vector produced from the matrix and so the complexity of the problem is increased. This is an area that is featured in the future work section, §7.2.3.

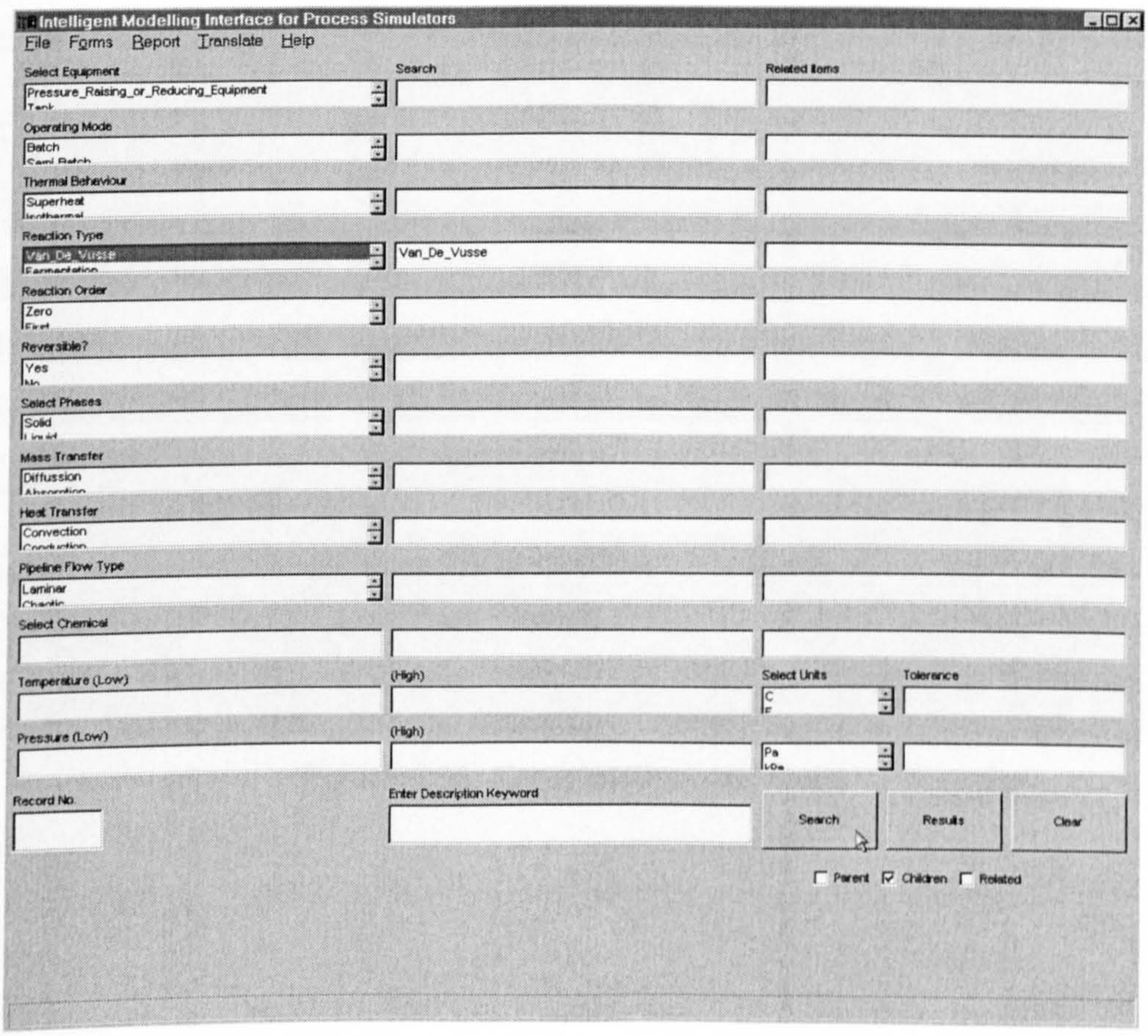


Figure 6.35. Completed search form.

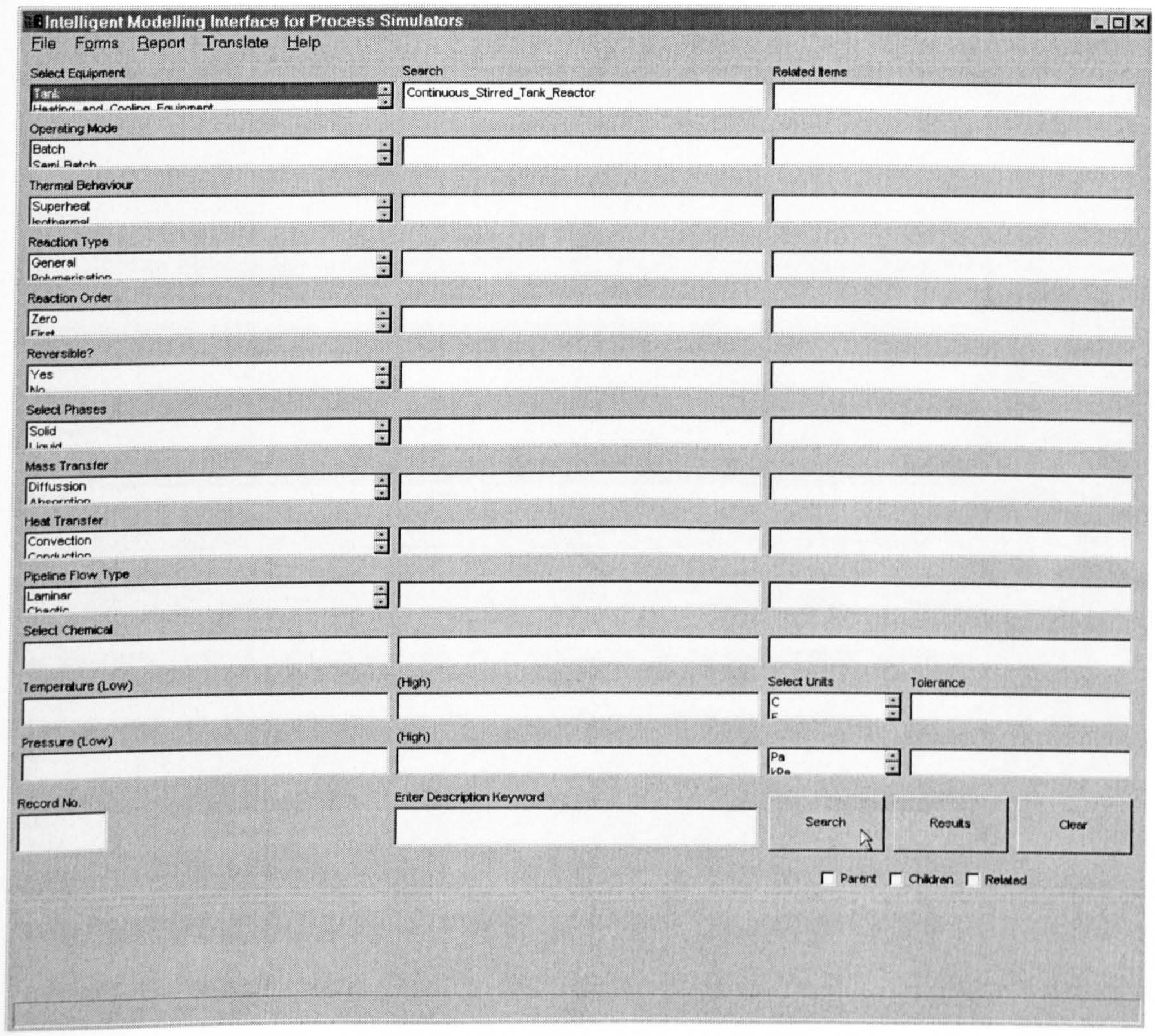


Figure 6.36. Completed search form.

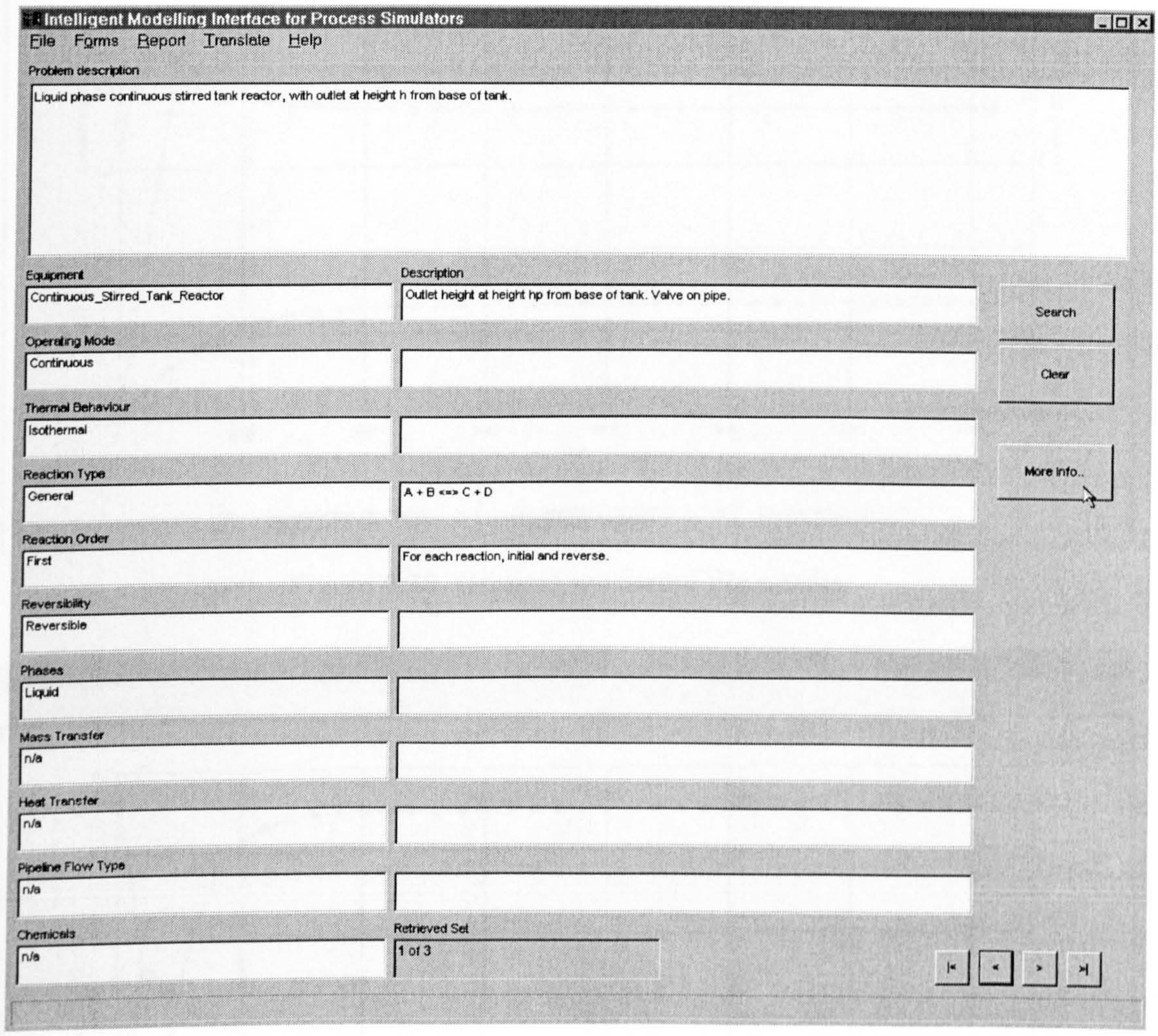


Figure 6.37. Results form.

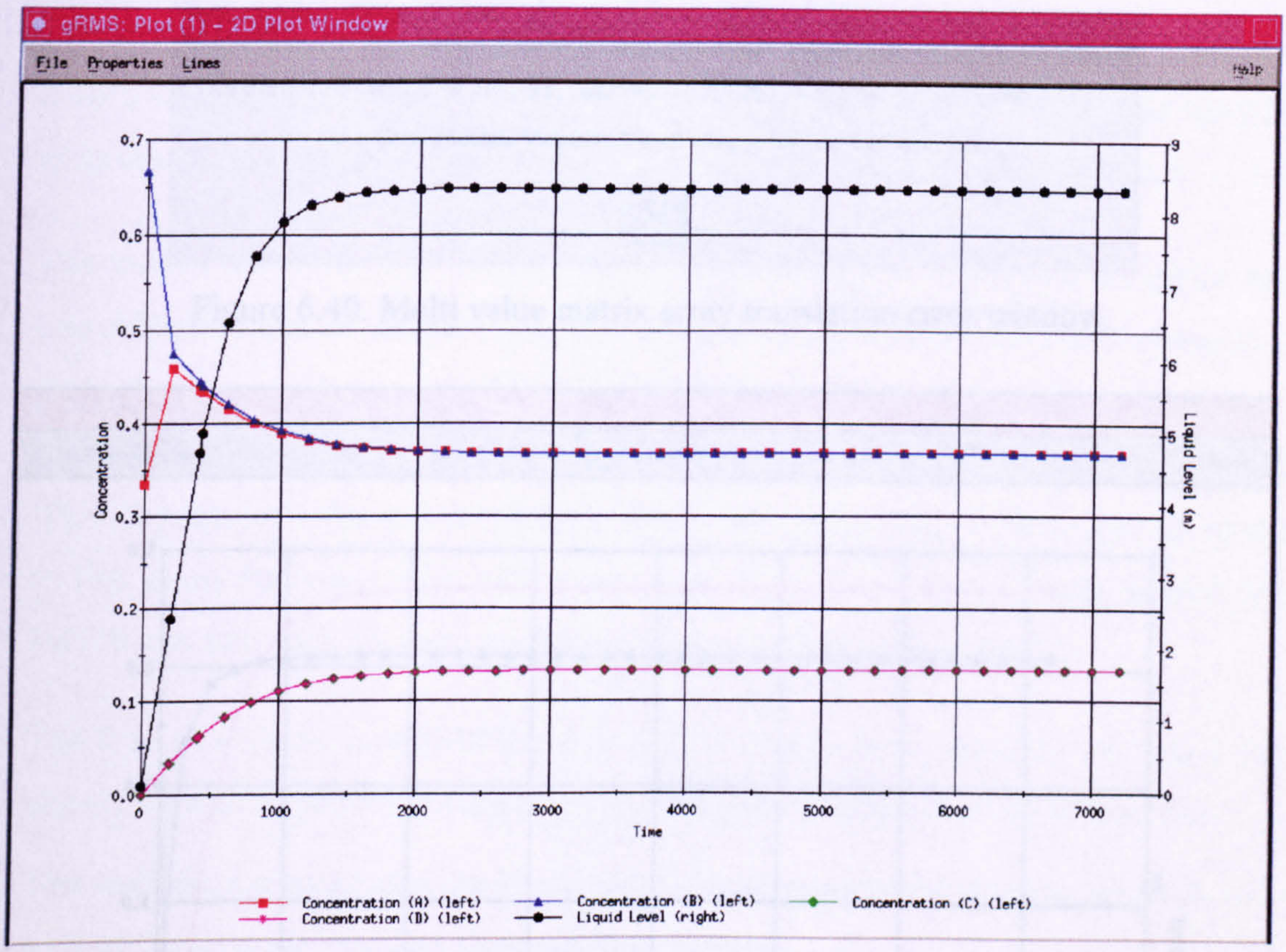


Figure 6.38. Original translated file results.

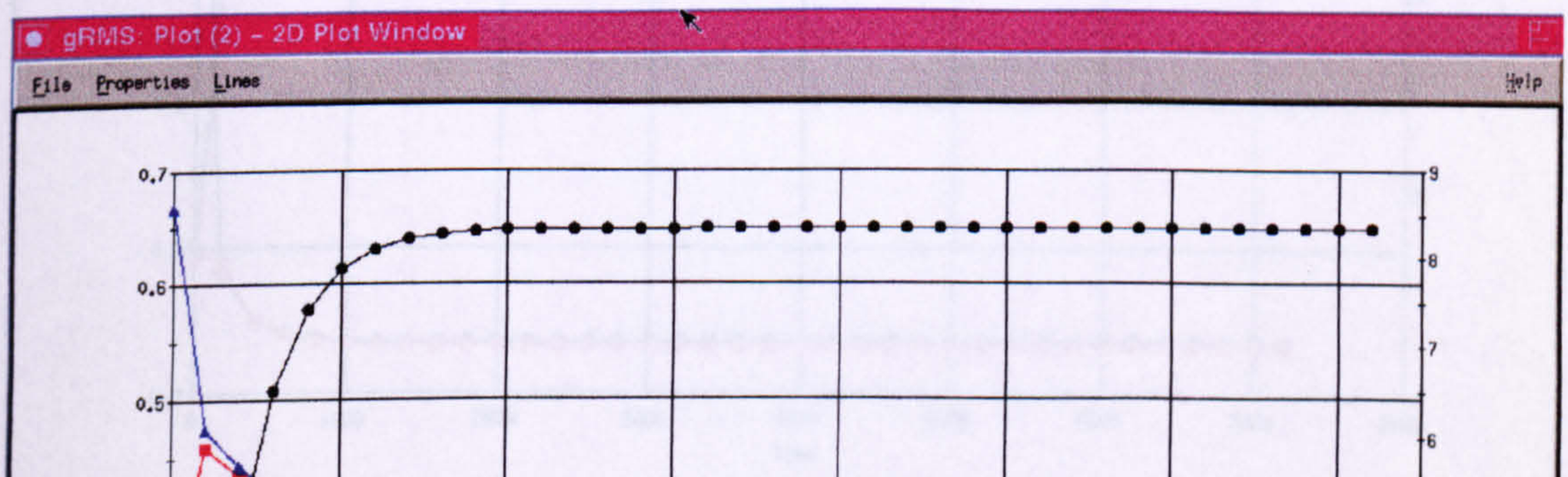


Figure 6.39. Original results from PSE (1998).

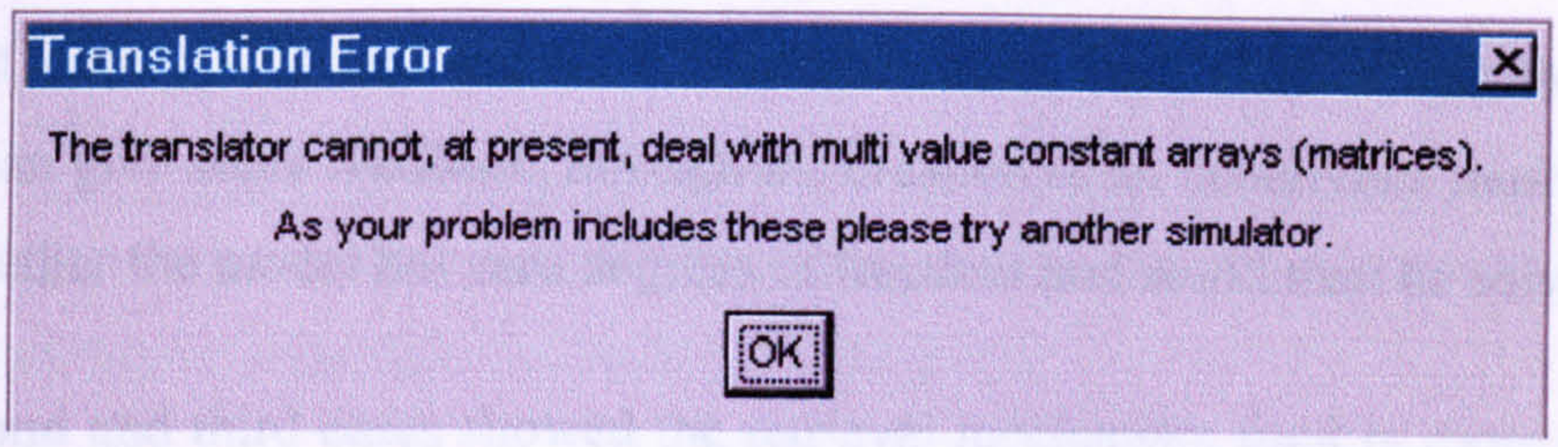


Figure 6.40. Multi value matrix array translation error window.

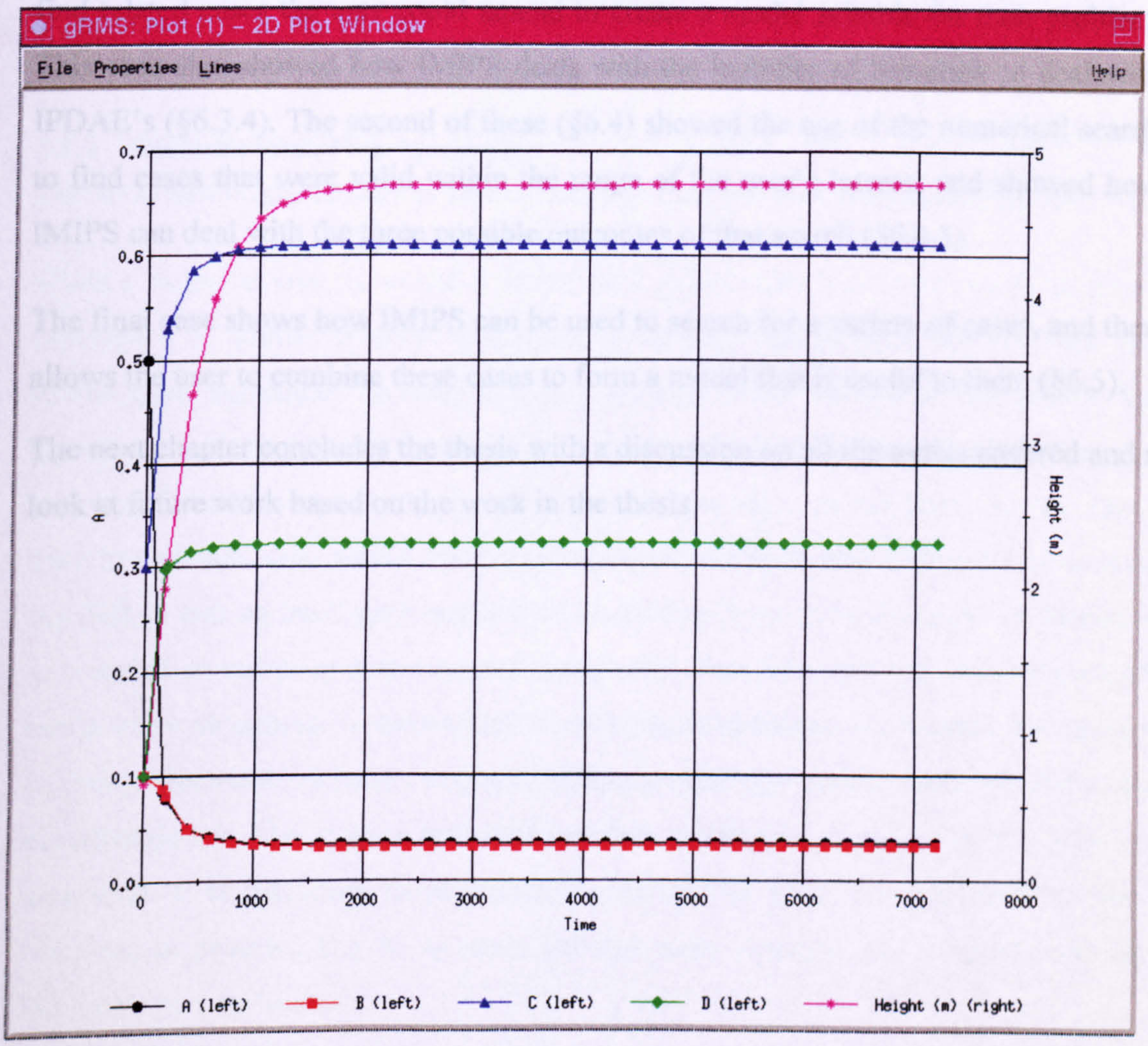


Figure 6.41. gPROMS results plot. Component concentration and height variation with time.

6.6 Summary

This chapter has shown the properties of IMIPS through the use of four case studies and has run through the methods for using IMIPS for the three main scenarios. It has also shown some of the shortcomings of the system and of Simulink. These are expanded on in the future work section, §7.2.

The first case (§6.2) outlined how to write a case from scratch and showed how IMIPS can give some feedback, through the creation of an occurrence matrix (§6.2.2), as to whether the model has zero degrees of freedom and could then be solved.

The second and third cases showed the retrieval mechanism used in more detail with two different types of problem. The first of these (§6.3) used the hierarchical search to find related cases the user could amend to create a model suitable for their problem. This case also showed how IMIPS deals with the inability of Simulink to deal with IPDAE's (§6.3.4). The second of these (§6.4) showed the use of the numerical search to find cases that were valid within the range of the user's interest and showed how IMIPS can deal with the three possible outcomes of that search (§6.4.1).

The final case shows how IMIPS can be used to search for a variety of cases, and then allows the user to combine these cases to form a model that is useful to them (§6.5).

The next chapter concludes the thesis with a discussion on all the topics covered and a look at future work based on the work in the thesis.

7. Conclusions and Future Work

This chapter outlines the conclusions that can be drawn from the thesis and prospective future work that may lead from the thesis.

7.1 Conclusions

The following conclusions can be drawn from the work carried out for the thesis.

7.1.1 Ideas Behind This Approach

The ideas behind the approach we have taken to this problem are outlined below.

When a modeller tries to create a model of a system they base that model on prior experience of similar systems. If they have no prior knowledge they can then get advice from colleagues or from literature to help in the writing of the model. To include some form of prior knowledge in a system would be advantageous to the user and would enable them to review previous relevant cases to aid them in their task. This knowledge base would need to include all relevant information about a case (model) so that an intelligent search may be used to retrieve useful cases. A database is included in our programme to incorporate this idea. The way the cases are stated needs to be structured to ensure that all relevant information is included and so our indexing hierarchy has been developed (based on unit operations) to enable this to be accommodated. The largest drawback of this method is in the way the case is described. It is becoming more common practice that units are described by their function, or product, and so as cases become more complex our original indexing hierarchy may be limited.

The search itself needs to be able to reproduce the processes the user may use when looking through their own experiences. It needs to be able to retrieve all useful cases from the database to give the user the greatest chance of finding a case (or cases) that may be of use to them. To allow this our search methods are based on case-based reasoning techniques. These techniques can allow for good matches to be returned as long as the cases have been stated fully. There is still scope for incorrect cases to be returned, but this can be reduced with the correct use of the indexing scheme and the complete description of each case in the database.

7.1.2 Process Modelling Approaches

There are two main approaches to the modelling and simulation problem. They both have their own advantages and disadvantages, but behind the different interfaces the structure is still equation-oriented in nature. The equation-oriented, textual approach is the most versatile to use, but has little or no help or guidance about the problem for the user. These types of simulator package are mathematical solvers and, generally, do not have an expert system available to guide the user in constructing the model. It is up to the user to supply the necessary equations, operating data, and conditions, in a form the package understands, so that the package can solve the problem. This does allow the user to model complex or novel units more easily than with the black box approach (see below), but it can be far more difficult to debug the model if mistakes are made.

The modular approach, however, has in-built guidance for the units it can be used to model. This guidance is inherent in the black box style that they adopt. As the package already has in-built the equations each unit needs to be modelled, it is just for the user to supply the necessary operating data and conditions and the package can model the process. It is far harder, and sometimes impossible, to model complex or novel units with this approach, as to create a new unit and incorporate the relevant information into it can be very cumbersome. This black box method does incorporate automatic input file code generation, thus eliminating one error prone step from the modelling procedure. Thus, simpler models (where the units are more common) can be quicker to model using these systems.

These characteristics are summarised in table 7.1, along with the characteristics we are aiming for in our new system.

Modelling Approach	Modelling Features	Code Generation	Example Systems
Equation-Oriented Approach	Flexible equation based, but little or no guidance or help.	Manual: Time consuming, easy to make errors.	gPROMS, SpeedUp
Modular Approach	Fixed black box models from library. Choice of models and some guidance.	Automatic.	DIVA, HYSYS
NEW: Case-Based Reasoning Approach	*Flexible equation based. Provides guidance for model.*	*Automatic.*	*IMIPS*

Table 7.1. Brief review of current approaches and proposed approach.

7.1.3 Case Based Reasoning

As the natural way of approaching most engineering problems is to call upon the experience of the modeller or another expert to help with the creation and solving of the problem, an automation of this activity would enable solutions to be produced faster and with fewer errors.

Case indexing is more of an art form than a science. It takes a great deal of knowledge to create a predictive, robust, abstract, and useful index that will cater for all problems encountered. Most indexing schemes involve a great deal of evolution from the initial ideas to a useable index. Whether an automatic or manual indexing system is used is highly dependent on the domain of the study. The translation from a natural language problem description to a description that can be easily indexed is one where the human brain is still far more efficient than other automatic methods. Within our domain of interest, the cases are to be stated in a very structured and formal way, and so these fields can be used as the index for the system.

Cases can be retrieved using different search methods: the usual 'database' field search, and a partial or fuzzy search. As the possibility of a case in the library being exactly the same as the current problem is virtually zero, then a search that draws out all cases that might be slightly relevant is better than one searching for the exact answer. For this reason, a partial or fuzzy search is a far better way of drawing the most appropriate cases from the library. This retrieval procedure should also have a

method of ranking the retrieved case so that the more relevant ones are drawn to the user's attention before those that are less similar. The search should differentiate between numerical and symbolic values and vary its search techniques accordingly.

Automatic case adaptation is a very powerful method of solving problems, but only within a limited domain (such as that in CHEF). To automate the process over our domain seems, at present, virtually impossible with current methods and technologies. It would therefore seem more appropriate to concentrate on manual adaptation of cases and concentrate on the indexing and retrieval side of the case-based reasoning tool.

7.1.4 IMIPS

To enable a package to be flexible and still able to provide guidance means a package that requires some form of built in knowledge. The guidance in our new system is in the form of a knowledge base or case library. This consists of past cases that have been indexed so that they can be recalled at a later date. The indexing procedure allows for all possible problems to be indexed and the retrieval procedure allows for relevant cases to be recalled. As the chances of having an exact match for any problem in the case library is virtually zero the retrieval system is able to draw similar and relevant cases from the library for the user to view.

The problem input needs to be concise enough to include all relevant information and be structured to allow for an indexing procedure to be applied. This input must have all operating conditions, equations, and data to allow the automatic translation into a modelling package's input file. This translation could be carried out for many different packages. In the thesis gPROMS (PSE, 1999) and Simulink (MathWorks, 1999) input codes are automatically generated.

Chapters 4 and 5 outlined the reasoning behind the way IMIPS has been developed and the problems faced. IMIPS has been designed to allow the novice user to easily create a search and then gives them the ability to review and amend the retrieved cases with ease. The user then has the ability to create simulator input files at the touch of a button. These files can be run and the results compared. How a case can be specified, retrieved, reviewed, adapted and translated was discussed, and how the translators for Simulink and gPROMS were created was also described. These

chapters also outline the nuances of the translation procedure for the simulators for Simulink and gPROMS and also outline the three main scenarios that IMIPS can be used for:

- Scenario 1: A new case from scratch.
- Scenario 2: Retrieval and use of an existing case.
- Scenario 3: Retrieval, Adaptation, and use of an existing case.

To demonstrate the properties of IMIPS four case studies were used to show how IMIPS deals with the three main scenarios shown above.

The first case (§6.2) outlined how to write a case from scratch and showed how IMIPS can give some feedback, through the creation of an occurrence matrix (§6.2.2), as to whether the model has zero degrees of freedom and could then be solved.

The second and third cases showed the retrieval mechanism used in more detail with two different types of problem. The first of these (§6.3) used the hierarchical search to find related cases the user could amend to create a model suitable for their problem. This case also showed how IMIPS deals with the inability of Simulink to deal with IPDAE's (§6.3.4). The second of these (§6.4) showed the use of the numerical search to find cases that were valid within the range of the user's interest and showed how IMIPS can deal with the three possible outcomes of that search (§6.4.1).

The final case shows how IMIPS can be used to search for a variety of cases, and then allows the user to combine these cases to form a model that is useful to them (§6.5).

This thesis has shown that, although IMIPS is not a finished product, the ideas behind it are valid and that they are a suitable method of approaching this problem. The system is now at a stage where the ideas behind it have been shown to work and the system should move forward to continue progressing and evolving into a complete, working system. The system is written on packages that were available at the time, but to create a commercially viable product the system would need to be re-written as one package (this would remove one source of error, the ODBC link between the interface and the database) incorporating the database and interface. Other work to be

done on this project to improve and expand on the present state of IMIPS is described in the next section.

7.2 Future Work

As has been outlined within the case studies, the IMIPS system can be improved upon by progressing in a variety of directions. These are outlined below.

7.2.1 Adaptation of the Hierarchy Structure

The hierarchy structure is, at present, based on the unit operation as this is the way the majority of users look at a problem at the present time. There is, however, a move away from this approach as unit operations become more complex and varied. This move is to classing units by their function or the product of the operation.

The incorporation of these other schemes of classification should be included in future versions of IMIPS to allow plant that is difficult to identify with a single operation to be classified. This will also aid the user in their search by being able to search for the function of a unit/plant so that the likelihood of a suitable match is increased.

7.2.2 Improvements in the Search, Retrieval and Adaptation Mechanisms

The search and retrieval mechanism is limited, at present, by the length the SQL query can be. This can not be extended, but the field names used could be shortened to allow for longer queries. Also, there is no specific order given to the retrieved cases. They are returned in the order they are in the database. Some form of ranking system needs to be included in the retrieval so that the more relevant cases are shown first.

In its current state, IMIPS incorporates manual adaptation of the retrieved cases to the users specifications. This task should be automated to allow the user to concentrate on the initial problem specification and not worry about the editing of a retrieved case. At the moment the most likely cause for error (in model specification) is where the user adapts a past case and incorporates an error in this. These errors can be hard to spot and so automating this process would reduce the chance of a badly posed problem for the simulator to solve.

7.2.3 Investigation into the Relationship between Simulator Error Code and Original User Input

At present, if the code produced by the IMIPS translator creates errors at simulation, this information is not linked to the initial user input in any way. Although the error codes produced by both gPROMS and Simulink are very helpful in diagnosing the problem with the input file, it can be hard to relate these errors to the original case (in the database). Therefore, the addition of an expert system to interpret the simulator errors and give meaningful information about the original user input would be needed to improve this error fixing procedure.

7.2.4 Statement of Multi Value (Matrix) Constant Arrays within Simulink.

Within Simulink the statement of multi value constant arrays as matrices is not possible. This could be overcome by the incorporation of some form of FOR loops within Simulink to solve the equations involving these arrays and then combining the results (where necessary) into vector formats to be used in later calculations.

7.2.5 Consideration of Multi-Unit Cases

IMIPS only deals with single unit cases at present and this, although useful, is very limiting. The expansion of IMIPS features to include multi-unit cases needs to be investigated and implemented to improve IMIPS's usefulness and applicability to problems that are more likely to be faced in industry. The main problem to be overcome will be based on the linking of the individual units together to ensure consistency within the problem.

7.2.6 Addition of More Translators

There are many simulators on the market at present and these can all be viable as they all have something to offer that the others do not. Due to this, there will be some problems that are solved better by one simulator than another. The inclusion of more translators for more simulators is, therefore, a logical next step to ensure that the best solution is available for the problems being modelled. The translators needed would depend on the users requirements, but once written could be utilised when needed.

7.2.7 Expansion of the Cases in the Database of IMIPS.

The knowledge base within IMIPS is the case database. This database needs to include as many cases as possible to allow the user to be certain of some case retrieval when searching through the database. The additional cases need to be selected carefully to ensure a wide range of type of case, and to ensure that the indexing scheme can cope fully with the new cases. The indexing scheme, at present, is quite small and should be expanded as more cases are included.

References

AspenTech (1999) '*Aspen Custom Modeler (10.0/10.1-0) User Guides (1998-99)*', Aspen Technology Inc. USA.

Bär, M. & Zeitz, M. (1990) '*A Knowledge-Based Flowsheet-Oriented User Interface for a Dynamic Process Simulator*', **Computers and Chemical Engineering**, 14, pp. 1275-1280

Barber, J.A. (2000) '*Computer Generation of Process Models*', **PhD Thesis**, University of London

Biegler, L. (1989) '*Chemical Process Simulation*', **Chemical Engineering Progress**, October, pp. 50-61

Boston, J.F., Britt, H.I. & Tayyabkhan, M.T. (1993) '*Software: Tackling tougher tasks*', **Chemical Engineering Progress**, November, pp. 38-49

Chung, P.W.H. & Inder, R. (1992) '*Handling Uncertainty in Accessing Petroleum Exploration Data*', **Revue de L'Institute Francais du Petrole**, 47, pp. 305-314

Chung, P.W.H. & Jefferson, M. (1998) '*A Fuzzy Approach to Accessing Accident Databases*', **International Journal of Applied Intelligence**, 9, pp. 129-137

Clark. G. (1998) '*An Intelligent Front-End System for Chemical Engineering Modelling: Background and Proposed Methodology*', Internal Report, Plant Engineering Group, Loughborough University, Department of Chemical Engineering, Loughborough LE11 3TU, UK, 7 September 1998.

Dubois, D., Prade, H. & Testemale, C. (1988) '*Weighted Fuzzy Pattern Matching*', **Fuzzy Sets and Systems**, 28, pp. 313 - 331

Gaines, B.R. (1976) '*Foundations of Fuzzy Reasoning*', **International Journal of Man-Machines Studies**, 8, pp. 623-668

Hammond, K. (1986) '*CHEF: A Model Of Case-Based Planning*', **Proceedings of the AAAI '86**, Philadelphia, PA, pp. 237-258

Holl, P., Marquardt, W. & Gilles, E.D. (1988) '*DIVA - A Powerful Tool for Dynamic Process Simulation*', **Computers and Chemical Engineering**, 12, pp. 421-426

Ingham, J., Dunn, I.J., Heinzle, E. & Prenosil, J.E. (1995) '*Chemical Engineering Dynamics*', VCH, Weinheim

Illiffe, R.E., Chung, P.W.H. & Kletz, T.A. (1998) '*More Effective Permit-To-Work Systems*', **Institution of Chemical Engineers Symposium Series**, 144, pp. 181-194

Jeng, B.C. & Liang, T.P. (1995) '*Fuzzy Indexing and Retrieval in Case-Based Systems*', **Expert Systems With Applications**, 8, pp. 135-142

Koiranen, T., Virkki-Hatakka, T., Kraslawski, A. & Nyström, L. (1998) '*Hybrid, Fuzzy and Neural Adaptation in Case-Based Reasoning System for Process Equipment Selection*', **Computers and Chemical Engineering**, 22, pp. S997-S1000

Kolodner, J. (1983a) '*Maintaining Organization in a Dynamic Long-Term Memory*' **Cognitive Science**, 7 (4)

Kolodner, J. (1983b) '*Reconstructive Memory: A Computer Model*' **Cognitive Science**, 7 (4)

Kolodner, J. & Simpson, R. (1989) '*The MEDIATOR: Analysis of an Early Case-Based Problem Solver*' **Cognitive Science**, 13 (4), pp. 507-549

Kolodner, J. (1993) '*Case-Based Reasoning*', Morgan Kaufmann, CA.

Luyben, W.L. (1990) '*Process Modeling, Simulation and Control for Chemical Engineers*', McGraw-Hill

Maclay, D. (2000) '*Click and Code*', **IEE Review,** May 2000, pp. 25-28
Maher, M.L., Balachandran, M.B. & Zhang, D.M. (1995) '*Case-Based Reasoning in Design*', Lawrence Erlbaum Associates, Mahwah, NJ.
Marlin, T.E. (1995) '*Process Control – Designing Processes and Control Systems for Dynamic Performance*', McGraw-Hill
Marquardt, W. (1994), '*Trends in computer-aided process modeling*' **Proceedings of PSE'94**, 1, pp.1-23, Kyongju, KOREA.
MathWorks, (1999), '*SIMULINK Dynamic System Simulation for MATLAB, Using SIMULINK version 3*', The MathWorks, Inc., 24 prime Park Way, Natick, MA
Microsoft (1997) '*Microsoft Access – Relational Database Management System for Windows – Users Guide*' Microsoft Corporation
Ogata, K. (1992) '*System Dynamics*', Prentice Hall, Inc. p. 6
Pantelides, C.C. & Barton, P.I. (1993) '*Equation-Oriented Dynamic Simulation. Current Status and Future Perspectives*', **Computers and Chemical Engineering,** 17, pp. S263-S285
Pantelides, C.C. (1988) '*SPEEDUP - Recent Advances in Process Simulation*', **Computers and Chemical Engineering,** 12, pp. 745-755
Piela, P.C., Epperly, T.G. Westerberg, K.M. & Westerberg, A.W. (1991) '*ASCEND: An Object Oriented Computer Environment for Modeling and Analysis: The Modeling Language*', **Computers and Chemical Engineering,** 15, pp. 53-72
PSE (1998) '*gPROMS Training Course I*', Process Systems Enterprise Ltd., London, United Kingdom
PSE (1999a) '*gPROMS Advanced User Guide – Release 1.7*', Process Systems Enterprise Ltd., London, United Kingdom
PSE (1999b) '*gPROMS Introductory User Guide – Release 1.7*', Process Systems Enterprise Ltd., London, United Kingdom
ReMind Reference Manual, Cognitive Systems Inc., 1992, Boston
Sargent, R.W.H. (1967) *'Integrated Design and Optimization of Processes'*, **Chemical Engineering Progress,** 63, pp. 71-78
Schmucker, K.J. (1984) '*Fuzzy Sets, Natural Language Computations, and Risk Analysis*', **Computer Science Press**
Schiesser, W.E. (1994) '*Computational Mathematics in Engineering and Applied Science: ODEs, DAEs, and PDEs*', CRC Press, London
Shacham, M., Macchietto, S., Stutzman, L.F. & Babcock, P. (1982) '*Equation Oriented Approach to Process Flowsheeting*', **Computers and Chemical Engineering,** 6, pp. 79-95
Stephanopoulos, G. & Han, C. (1996) '*Intelligent Systems in Process Engineering: A Review*', **Computers and Chemical Engineering,** 20, pp. 743-791
Stephanopoulos, G., Henning, G. & Leone, H. (1990a) '*MODEL.LA. A Modeling Language for Process Engineering - I. The Formal Framework*', **Computers and Chemical Engineering,** 14, pp. 813-846
Stephanopoulos, G., Henning, G. & Leone, H. (1990b) '*MODEL.LA. A Modeling Language for Process Engineering - II. Multifaceted Modeling of Process Systems*', **Computers and Chemical Engineering,** 14, pp. 847-869
Svercek, W.Y., Mahoney, D.P. & Young, B.R. (Eds.) (2000) '*A Real-Time Approach to Process Control*', John Wiley and Sons, Ltd. Chichester. pp. 1-12
Telnes, K. (1992) '*Computer-aided modeling of dynamic processes based on elementary physics*', Doctoral Dissertation, Division of Engineering Cybernetics, Norwegian Institute of Technology, Trondheim.

Tsatsoulis, C. & Alexander, P. (1997) '*Integrating Cases, Subcases, and Generic Prototypes for Design*', in Maher, M.L. & Pu, P. (Eds.) **Issues and Applications of Case-Based Reasoning in Design**, Lawrence Erlbaum Associates, London, pp. 261-299

Ushold, M., Harding, N., Muetzelfeld, R. & Bundy, A. (1984) '*Intelligent Front End for Ecological Modeling*', Research Paper 223, Dept of Artificial Intelligence, University of Edinburgh.

Von Neumann, J. & Morgenstern, O. (1964) '*Theory of Games and Economic Behavior*', John Wiley and Sons, New York

Bibliography

Abufares, A.A. & Douglas, P.L. (1995) '*Mathematical Modelling and Simulation of an MTBE Catalytic Distillation Process Using SpeedUp and Aspen Plus*', **Transactions of the IChem**, 73, pp. 3-12

Acorn, T & Walden, S. (1992) '*SMART: Support Management Cultivated Reasoning Technology For Compaq Customer Service*', **Proceedings of AAAI '92**, Cambridge, MA, AAAI Press/MIT Press

Alterman, R. (1986) '*An Adaptive Planner*', **Proceedings of AAAI '86**, Cambridge, MA, AAAI Press/MIT Press

Bär, M., Schaffner, J., Selg, W. & Zeitz, M. (1993) '*Functionality and Implementation of a Knowledge-Based Flowsheet-Oriented User Interface for the Dynamic Process Simulator DIVA*', **Simulation**, 61, pp. 117-123

Barletta, R. (1992) '*An Introduction to Case-Based Reasoning*', **AI Expert**, 6, pp. 43-49

Barton, P.I. (1992) '*The Modelling and Simulation of Combined Discrete/Continuous Processes*', **PhD Thesis**, University of London

Barton, P.I. & Pantelides, C.C. (1994) '*Modeling of Combined discrete/Continuous Processes*', **AIChE Journal**, 40, pp. 966-979

Carbonell, J.G. (1986) '*Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition*', in Michalski, R.S., Carbonell, J.G. & Mitchell, T.M. (Eds.) **Machine Learning: An Artificial Intelligence Approach Vol. 2**, Morgan Kaufmann

Cho, J.H. (1997) '*Computer Aids for Mathematical Model Building*', **PhD Thesis**, University of London

Chung, P.W.H. & Jefferson, M. (1997) '*Accident Databases - Indexing and Retrieval*', **Ling Seminar**

Clark, G., Rossiter, D. & Chung, P.W.H. (2000) '*Intelligent Modelling Interface for Dynamic Process Simulators*', **Transactions of the IChemE**, 78, Part A, pp. 823-839

Gani, R., Sørensen, E.L. & Perregaard, J. (1992) '*Design and Analysis of Chemical Processes through DYNSIM*', **Industrial Engineering and Chemical Research**, 31, pp. 244-254

Jensen, A.K. & Gani, R. (1999) '*A Computer Aided Modeling System*' **Computers and Chemical Engineering**, 23, pp. S673-S678

Kambhampati, S. & Hendler, J.A. (1992) '*A Validation Structure Based Theory Of Plan Modification And Reuse*' **Artificial Intelligence Journal**, 55, pp. 193-258

Ketler, K. (1993) '*Case-Based Reasoning: An Introduction*', **Expert Systems with Applications**, 6, pp. 3-8

Khan, T & Yip, Y.J. (1995) '*CBT II - Case-Based Computer-Aided Instruction: Survey of Principles, Applications and Issues*', **The Knowledge Engineering Review**, 10, pp. 235-268

Koton, P. (1988) '*Integrating Case-Based and Casual Reasoning*' **Proceedings of the Tenth Annual Conference of Cognitive Science Society**, Hillsdale, NY, Lawrence Erlbaum Associates

Kröner, A., Holl, P., Marquardt, W. & Gilles, E.D. (1990) '*DIVA - An Open Architecture for Dynamic Simulation*', **Computers and Chemical Engineering**, 14, pp. 1289-1295

Maher, M.L. & Pu, P. (Eds.) (1997) '*Issues and Applications of Case-Based Reasoning in Design*', Lawrence Erlbaum Associates, Mahwah, NJ.

Marquardt, W. (1991), '*Dynamic Process Simulation – Recent Trends and Future Challenges*', in Arkun, Y. & Ray, E.D. (Eds), **Chemical Process Control CPC-IV**, CACHE, Austin, AIChE, New York, pp. 131-180

Mott, S. (1993) '*Case-Based Reasoning: Market, Applications, and Fit with Other Technologies*', **Expert Systems with Applications**, 6, pp. 97-104

Näf, U.G. (1994) '*Stochastic Simulation using gPROMS*', **Computers and Chemical Engineering**, 18, pp. S743-S747

Navinchandra, D. (1988) '*Case-Based Reasoning in CYCLOPS, a Design Problem Solver*' **Proceedings of the DARPA Workshop on Case-Based Reasoning**, San Mateo, CA, Morgan Kaufmann, pp. 286-301

Oh, M. (1995) '*Modelling and Simulation of Combined Lumped and Distributed Processes*', **PhD Thesis**, University of London

Oh, M. & Pantelides, C.C. (1996) '*A Modelling and Simulation Language for Combined Lumped and Distributed Parameter Systems*', **Computers and Chemical Engineering**, 20, pp. 611-633

Pantelides, C.C. & Britt, H.I. (1995) '*Multipurpose Process Modelling Environments*', In L.T. Biegler & M.F. Doherty, (Eds.), **Proceedings of the Conference on Foundations of Computer Aided Process Design '94**, CACHE Publications, pp. 128-141

Pantelides, C.C. (1996) '*gPROMS: An Advanced Tool for Process Modelling, Simulation and Optimisation*', Paper presented at **CHEMPUTERS EUROPE III**, Frankfurt, 29-30 October, 1996

Perkins, J.D. & Sargent, R.W.H. (1982) '*SPEEDUP: A Computer Program for Steady-State and Dynamic Simulation and Design of Chemical Processes*', **AIChE Symposium Series**, 78, pp. 1-11

Perregaard, J., Pedersen, B.S. & Gani, R. (1992) '*Steady-State and Dynamic Simulation of Complex Chemical Processes*', **Transactions of the Institution of Chemical Engineers**, 70, pp. 99-109

Ram, A. (1993) '*Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases*', **Machine Learning**, 10, pp. 201-248

Sargent, R.W.H. & Westerberg, A.W. (1964) '*"SPEED-UP" in Chemical Engineering Design*', **Transactions of the Institution of Chemical Engineers**, 42, pp. T190-T197

Sargent, R.W.H., Perkins, J.D., Cho, J.H. & Barber, J. (1997) '*Computer Aids for Process Model Building*', From the **Centre for Process Systems Engineering Annual Report 1997**, pp. 62-63

Schenk, M., Gani, R., Bogle, I.D.L. & Pistikopoulos, E.N. (1999) '*A Hybrid Approach for Reactive Separation Systems*' **Computers and Chemical Engineering**, pp. S419-S422

Seader, J.D. (1985) '*Computer Modelling of Chemical Processes*', **AIChE Monograph Series**, 15 (Volume 81)

Simoudis, E. (1992) '*Using Case-Based Retrieval For Customer Technical Support*', **IEEE Expert**, 7 (5), pp. 7-13

Steward, D.V. (1962) '*On an approach to techniques for the analysis of the structure of large scale systems of equations*', **Soc. Ind. App. Math. Rev.**, 4, pp. 321-342

Sycara, K., Guttal, R., Koning, J., Narasimhan, S. & Navinchandra, D. (1990) '*CADET: A Case-Based Synthesis Tool for Engineering Design*', **International Journal of Expert Systems**

Sycara, K.P. & Navinchandra, D. (1989) '*Integrated Case-Based Reasoning And Qualitative Reasoning In Engineering Design*', in J.S. Gero (Ed.), **Artificial Intelligence in Design,** NY, Springer Verlag, pp. 231-250

Von Watzdorf, R., Näf, U.G., Barton, P.I. & Pantelides, C.C. (1994) '*Deterministic and Stochastic Simulation of Batch/Semi-Continuous Processes*', **Computers and Chemical Engineering,** 18, pp. S343-S347

Watson, I. (1997) '*Applying Case-Based Reasoning: Techniques for Enterprise Systems*', Morgan Kaufmann, San Fransisco, CA

Westerberg, A.W. & Benjamin, D.R. (1985) '*Thoughts on a Future Equation-Oriented Flowsheeting System*', **Computers and Chemical Engineering,** 9, pp. 517-526

Winkel, M.L., Zullo, L.C., Verheijen, P.T.J. & Pantelides, C.C. (1995) '*Modelling and Simulation of the Operation of an Industrial Batch Plant using gPROMS*', **Computers and Chemical Engineering,** 19, pp. S571-S576

Web Sites of Interest

The ASCEND web page: http://www.cs.cmu.edu/~ascend/

The ASPEN PLUS web page: http://www.aspentech.com/

The CLIPS web page: http://web.ukonline.co.uk/julian.smart/wxclips/

The DIVA web page: http://www.isr.uni-stuttgart.de/diva/diva.html

The gPROMS web page: http://www.psenterprise.com/gPROMS/

The HYSIM web page: http://www.hyprotech.com/

The HYSYS web page: http://www.hyprotech.com/

The MATLAB web page: http://www.mathworks.co.uk/

The Simulink web page: http://www.mathworks.co.uk/

The SpeedUp web page: http://www.aspentech.com/

The VisSim web page: http://www.adeptscience.co.uk/products/mathsim/vissim/

Appendices

I. Syntax

Below are some examples of equations and their IMIPS representation.

Natural Language $k = Ae^{-\frac{E}{Rg \cdot T}}$

IMIPS
```
k = A*EXP(-E/(Rg*T))
```

Natural Language $rhoc \cdot Vc \cdot Cpc \frac{d(Tc)}{dt} = Fc \cdot Cpc(Tcin - Tc) + Q$

IMIPS
```
rhoc*Vc*Cpc*$Tc = Fc*Cpc*(Tcin-Tc) + Q
```

Natural Language $-BedVoid \cdot Dz \frac{\partial C}{\partial (ax)} = u(Cfeed - C)$

IMIPS
```
-BedVoid*Dz*PARTIAL(C,ax) = u*(Cfeed-C)
```

Natural Language $Q = OverallU \cdot Area \int_0^{} (T(ax,R) - Tc) d(ax)$

IMIPS
```
Q = OverallU*Area*INTEGRAL(ax:=0:L;T(ax,R)-Tc)
```

Natural Language IF Height > Hp THEN Fout = ValveConst · (Height-Hp) ELSE Fout = 0

IMIPS
```
IF Height > Hp THEN Fout = ValveConstant*(Height-
Hp) ELSE Fout = 0
```

Natural Language WHILE the temperature is too low, heat the vessel

IMIPS
```
WHILE Temp < Temp_fix DO Q = U*A*ΔT
```

Other usual signs apply, e.g. *, /, +, -, COS, SIN, TAN, etc. and these are shown on the next page

The symbols syntax for IMIPS is shown below.

IMIPS	Description
*	Multiplication
+	Addition
-	Subtraction
/	Division
^	Raised to the power
ABS	Absolute value
ACOS	Arc-cosine
ACOSH	Arc-hyperbolic cosine
ASIN	Arc-sine
ASINH	Arc-hyperbolic sine
ATAN	Arc-tangent
ATANH	Arc-hyperbolic tangent
COS	Cosine
COSH	Hyperbolic cosine
ERF	Error function
EXP	Exponential
IF arg THEN arg ELSE arg	IF … THEN … ELSE loop
WHILE arg DO arg	WHILE … DO loop
INT	Truncation of a real argument
LOG	Natural logarithm
LOG10	Logarithm to base 10
MAX	The largest of the arguments
MIN	The smallest of the arguments
PRODUCT	The product of the arguments
SIGN	The sign of the argument
SUM	The sum of the arguments
SIN	Sine
SINH	Hyperbolic sine
SQRT	Square root
TAN	Tangent
TANH	Hyperbolic tangent
$(arg)	Derivative, with respect to time
PARTIAL(x:y,z)	Partial derivative of x with respect to y, and with respect to z
INTEGRAL(x:=a:b;f(x))	Integral of f(x) from x=a to x=b

Table I.1. Equation syntax for IMIPS.

II. Indexing for Cases

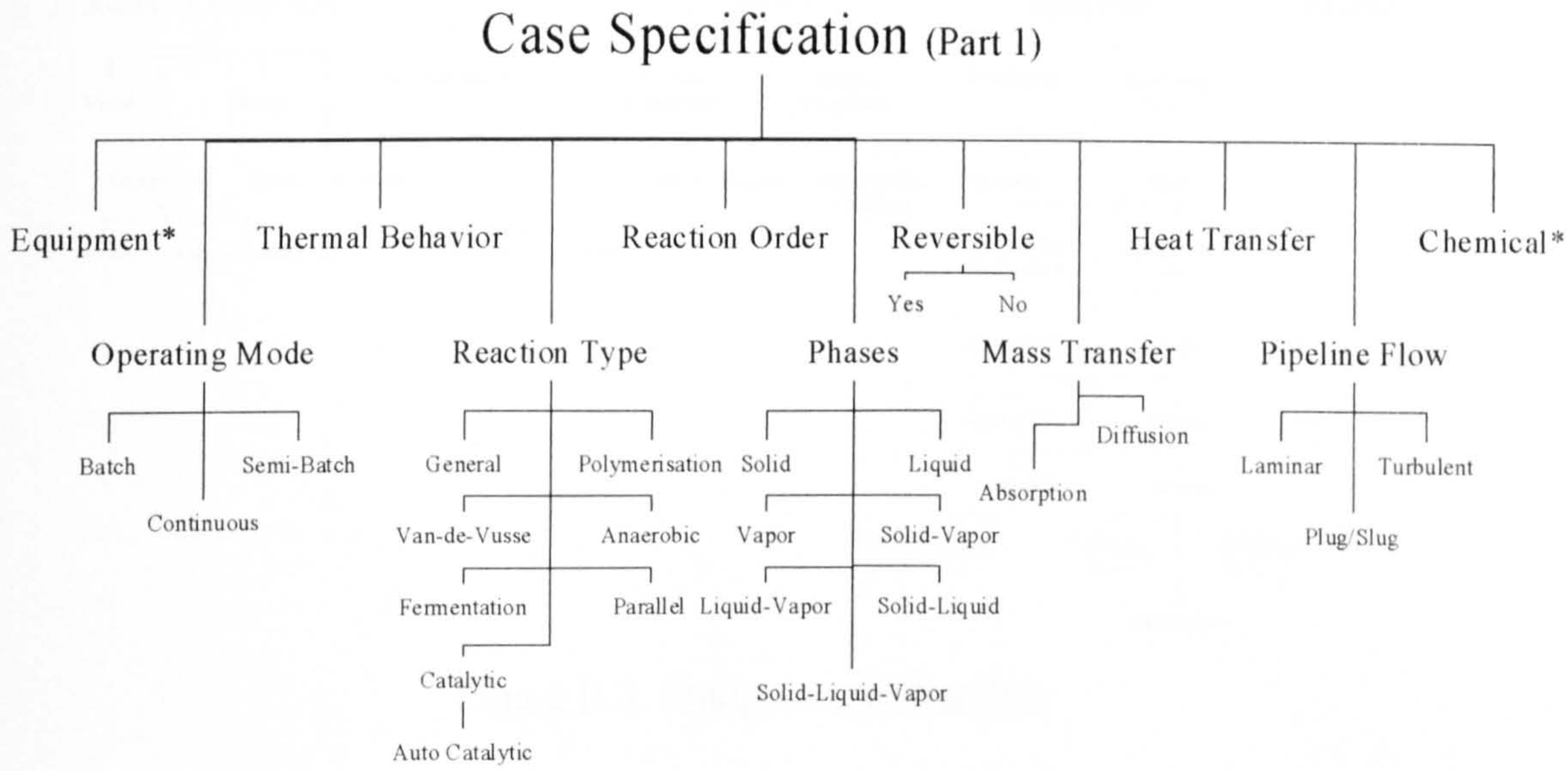


Figure II.1. Case Specification Hierarchy

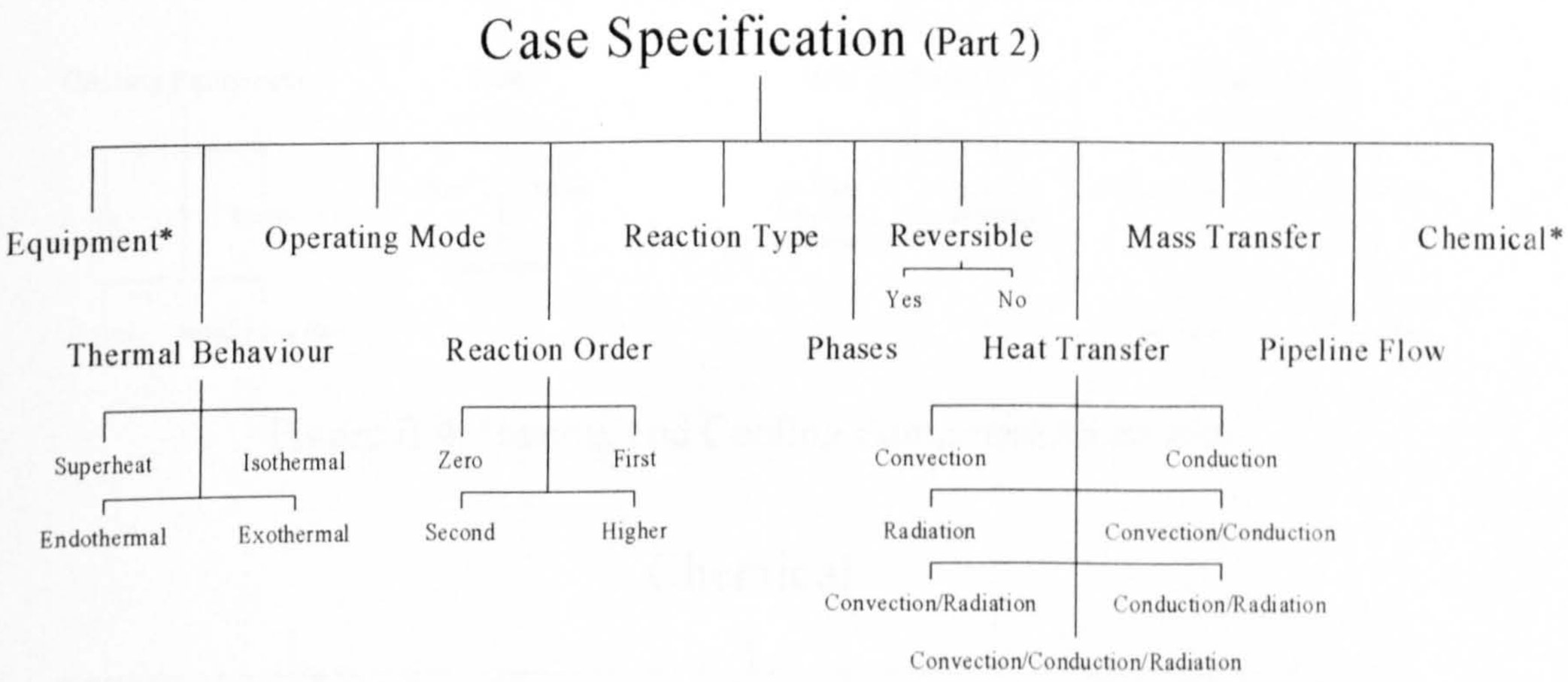


Figure II.2. Case Specification Hierarchy

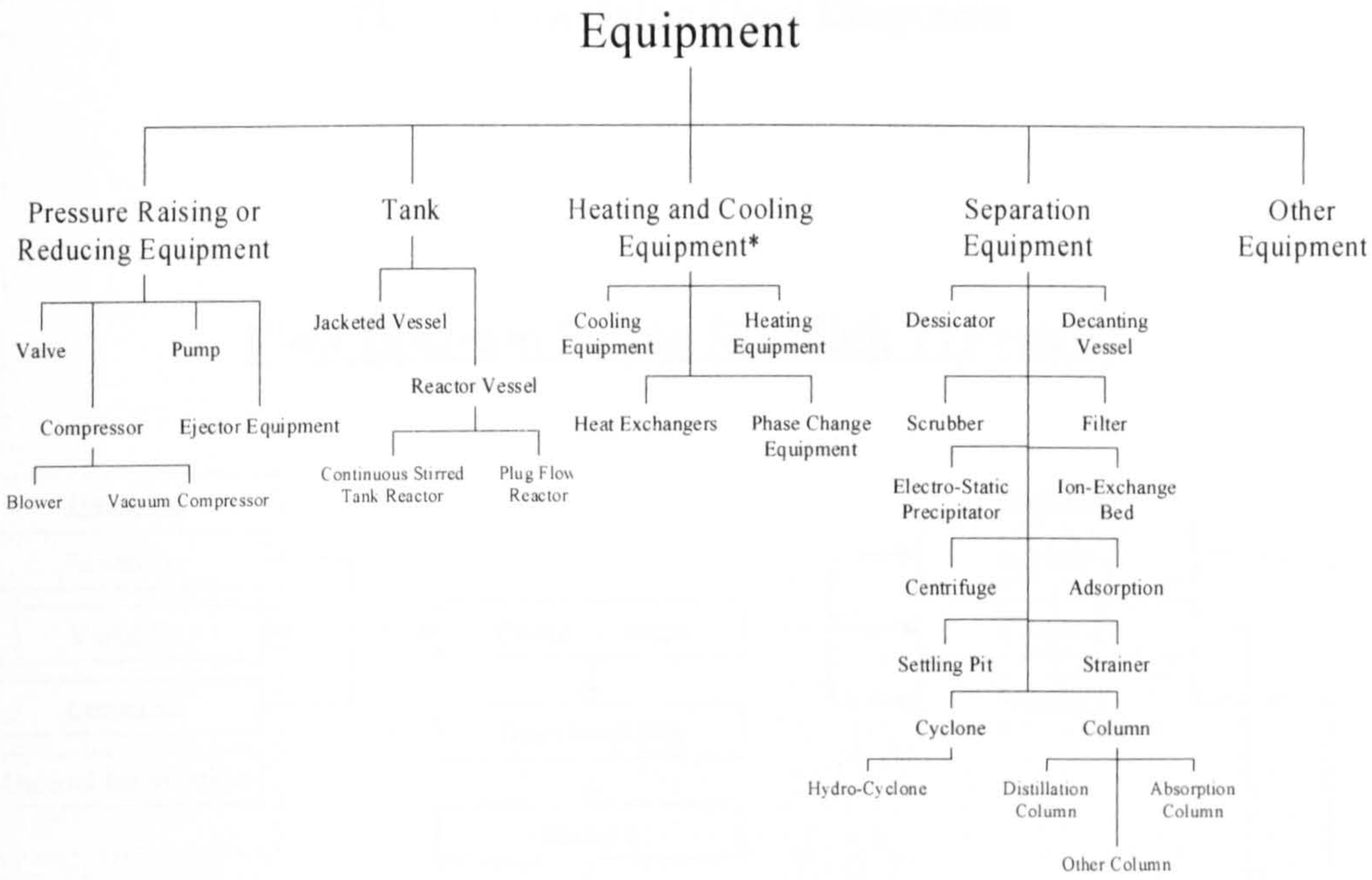


Figure II.3. Equipment Hierarchy

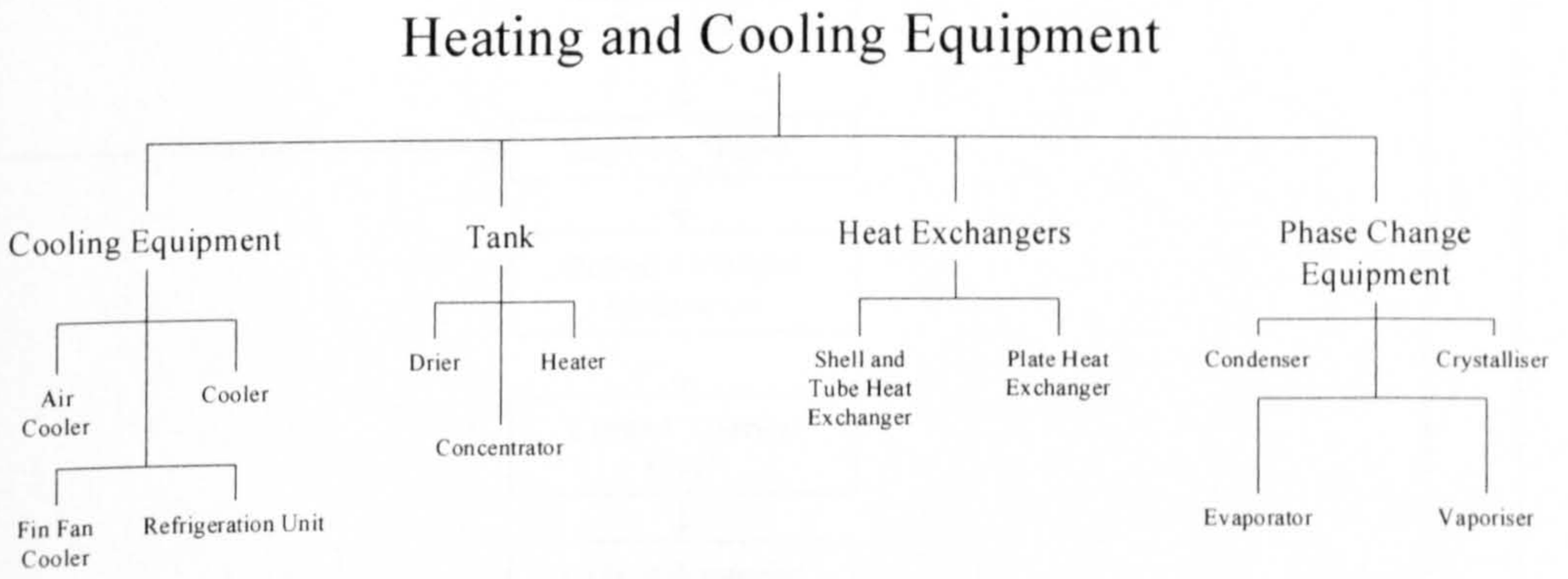


Figure II.4. Heating and Cooling Equipment Hierarchy

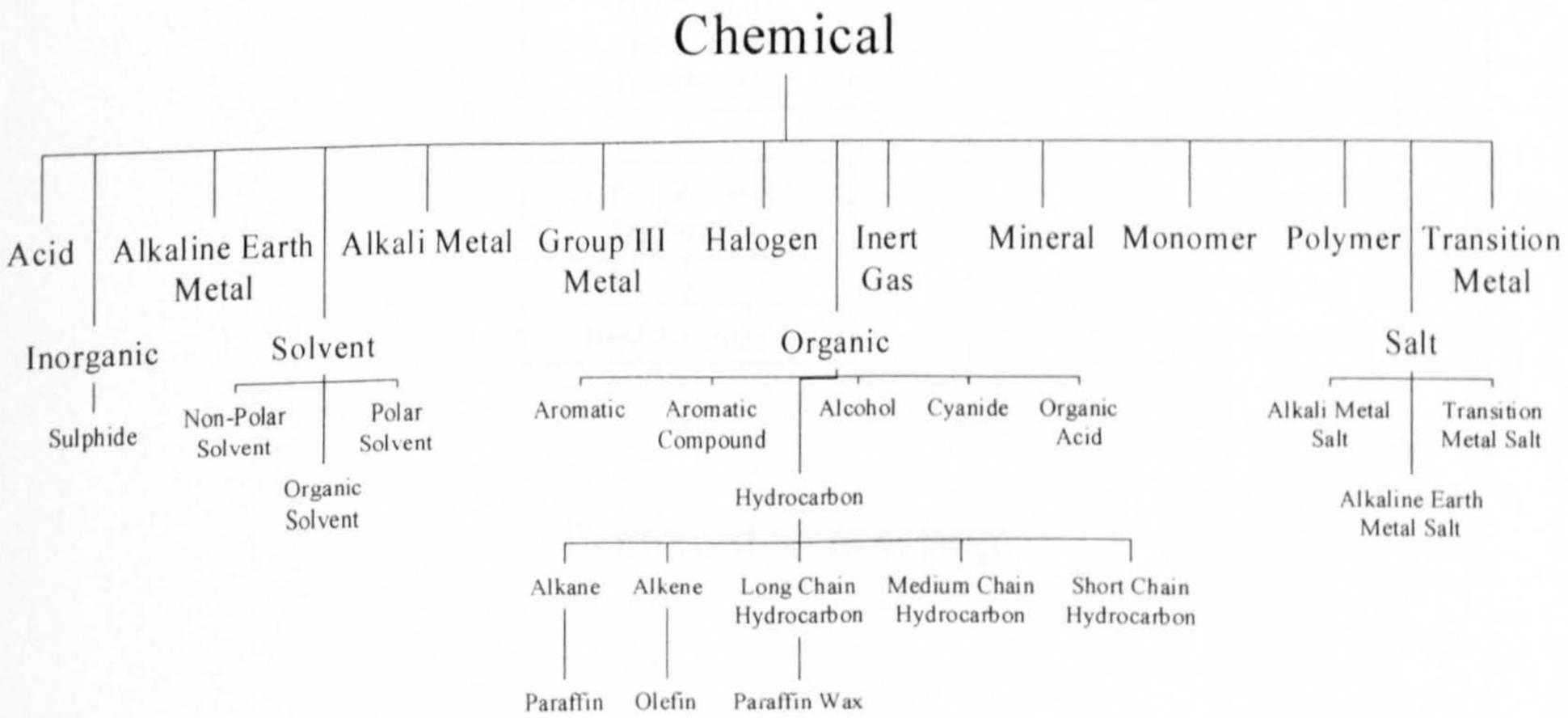
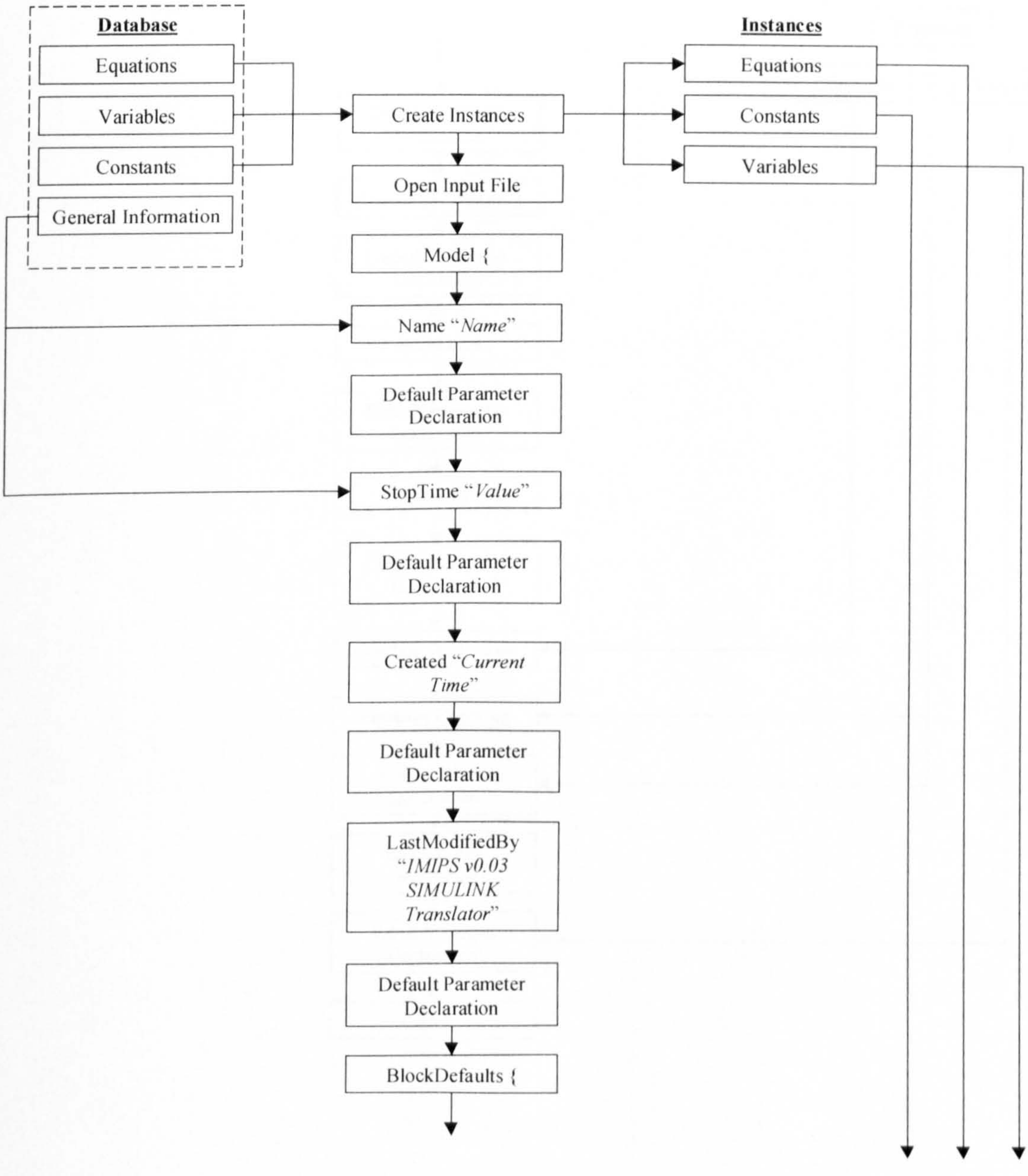


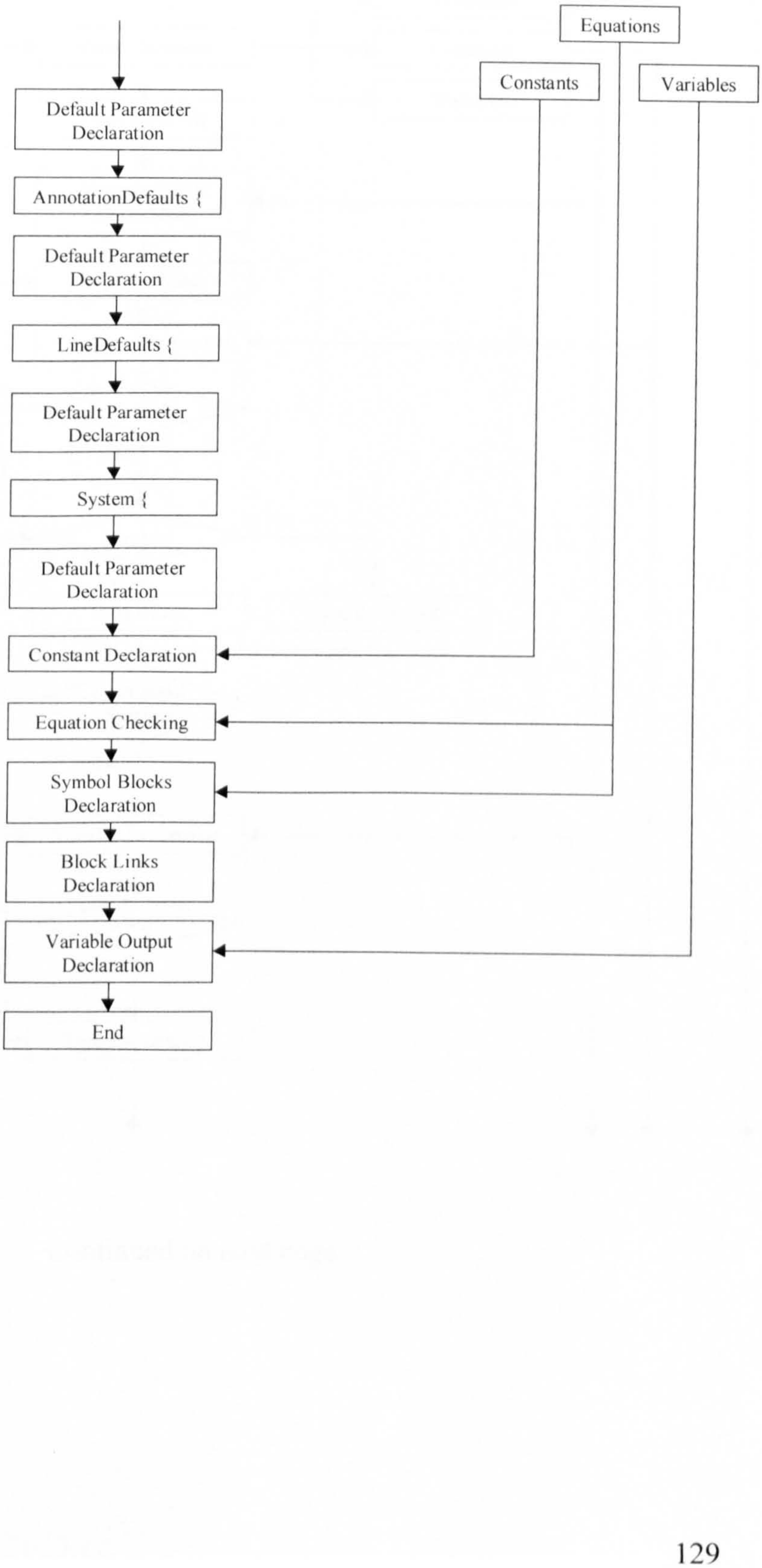
Figure II.5. Chemical Hierarchy

Flow Diagram for the Simulink Translator

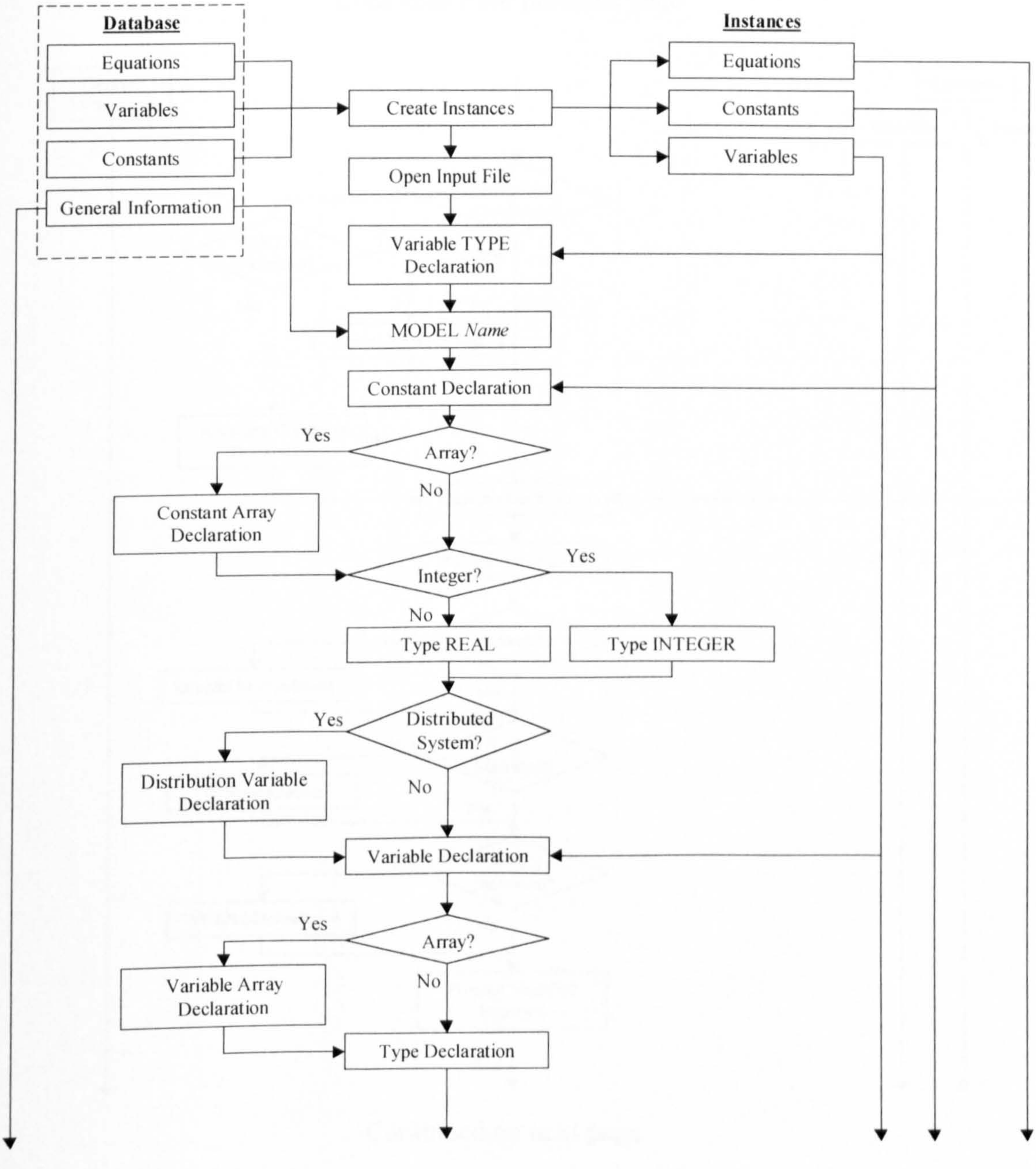


Continued on next page

Continues from previous page

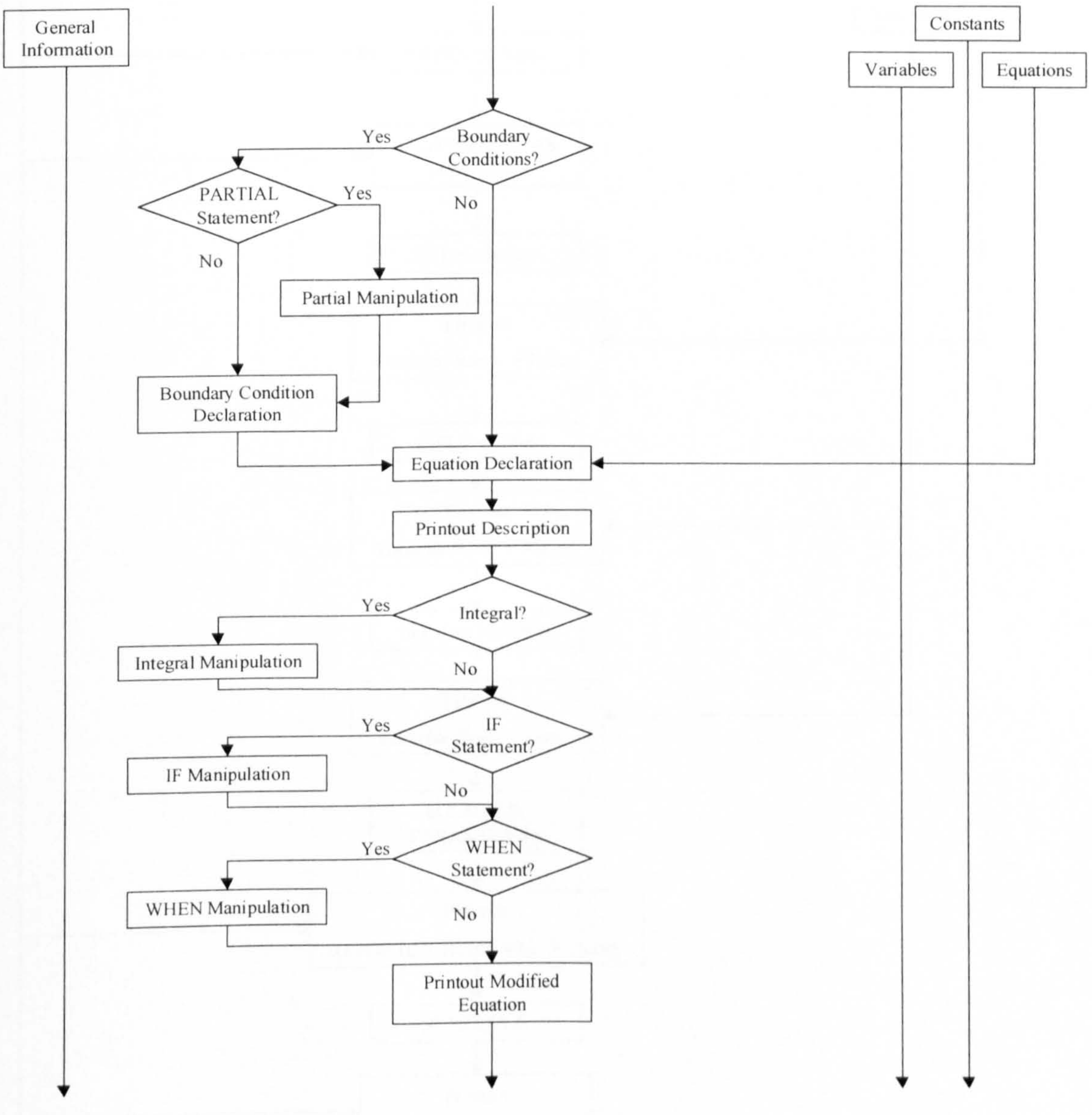


Flow Diagram for the gPROMS Translator



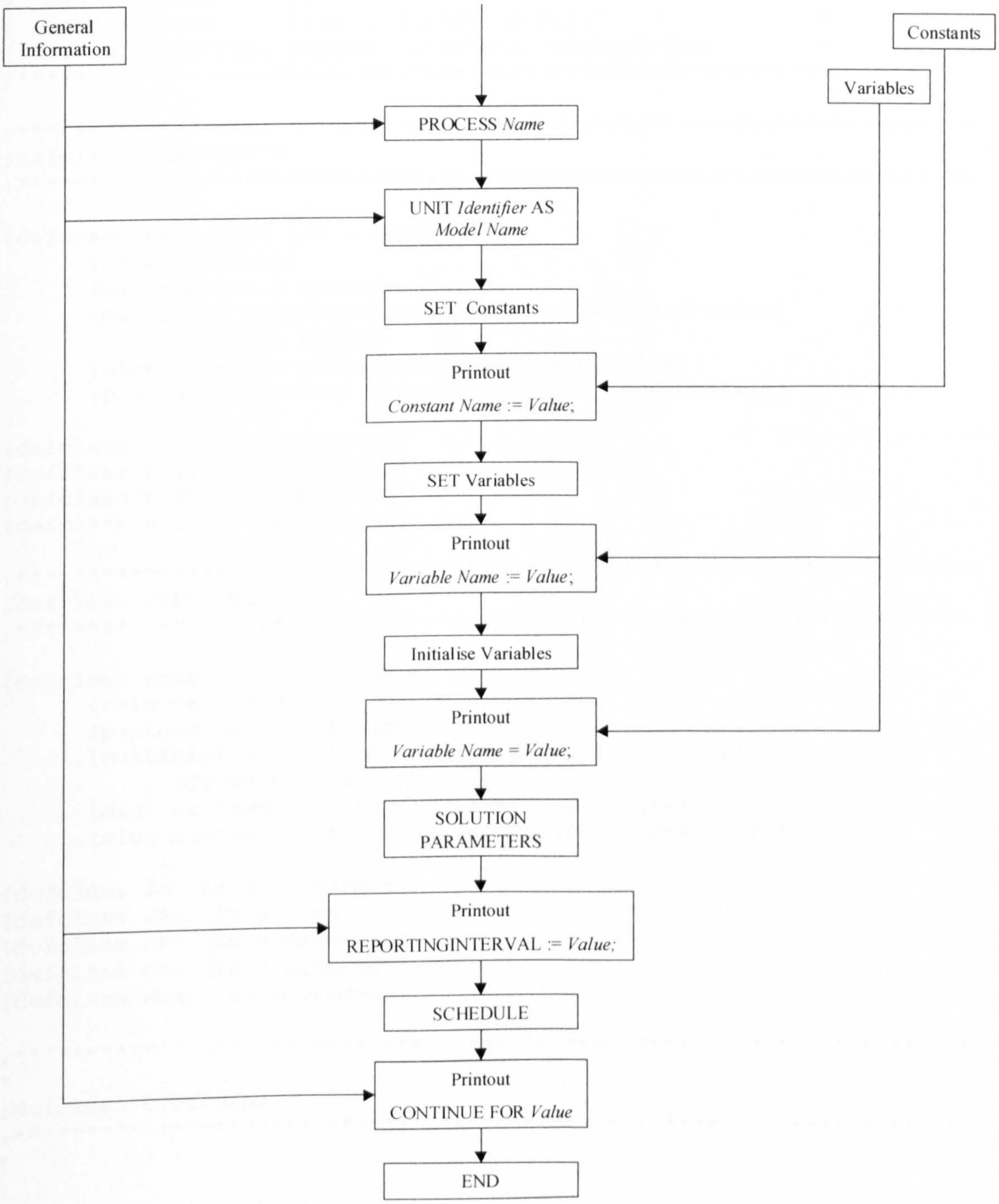
Continued on next page

Continues from previous page



Continued on next page

Continues from previous page



IV. Code for IMIPS programme

IV.1. Selection hierarchies declaration: classes.clp

```
;*****************************************************
;  Author :          Graham Clark                     *
;  File Name:        classes.clp                      *
;  Last Updated:     20/12/00                         *
;  Description:      Class declarations File          *
;       adapted from Richard Illiffe's Original Work  *
;*****************************************************

;*****************************************************
;defclass TEMP-UNITS
;*****************************************************

(defclass temp-unit (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (default none))
      (slot synonym  (create-accessor read-write))
      (slot abbreviation  (create-accessor read-write)))

(defclass C (is-a temp-unit))
(defclass F (is-a temp-unit))
(defclass K (is-a temp-unit))
(defclass R (is-a temp-unit))

;*****************************************************
;defclass PRES-UNITS
;*****************************************************

(defclass pres-unit (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (default none))
      (slot synonym  (create-accessor read-write))
      (slot abbreviation  (create-accessor read-write)))

(defclass Pa (is-a pres-unit))
(defclass kPa (is-a pres-unit))
(defclass mPa (is-a pres-unit))
(defclass bar (is-a pres-unit))
(defclass mbar (is-a pres-unit))

;*****************************************************
*
;defclass EQUIPMENT
;*****************************************************
*

(defclass equipment (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (default none))
      (slot synonym  (create-accessor read-write))
      (slot abbreviation  (create-accessor read-write)))
```

```
(defclass Pressure_Raising_or_Reducing_Equipment (is-a equipment))
  (defclass Compressor (is-a Pressure_Raising_or_Reducing_Equipment))
     (defclass Blower (is-a Compressor))
     (defclass Vacuum_Compressor (is-a Compressor))
  (defclass Ejector_Equipment (is-a
Pressure_Raising_or_Reducing_Equipment))
  (defclass Pump (is-a Pressure_Raising_or_Reducing_Equipment))
  (defclass Valve (is-a Pressure_Raising_or_Reducing_Equipment))

(defclass Tank (is-a equipment))
  (defclass Jacketed_Vessel (is-a Tank))
  (defclass Reactor_Vessel (is-a Tank))
     (defclass Plug_Flow_Reactor (is-a Reactor_Vessel))
     (defclass Continuous_Stirred_Tank_Reactor (is-a Reactor_Vessel))

(defclass Heating_and_Cooling_Equipment (is-a equipment))
  (defclass Cooling_Equipment (is-a Heating_and_Cooling_Equipment))
     (defclass Air_Cooler (is-a Cooling_Equipment))
     (defclass Cooler (is-a Cooling_Equipment))
     (defclass Cooling_Tower (is-a Cooling_Equipment))
     (defclass Fin_Fan_Cooler (is-a Cooling_Equipment))
     (defclass Refrigeration_Unit (is-a Cooling_Equipment))
  (defclass Heating_Equipment (is-a Heating_and_Cooling_Equipment))
     (defclass Concentrator (is-a Heating_Equipment))
     (defclass Drier (is-a Heating_Equipment))
     (defclass Heater (is-a Heating_Equipment))
  (defclass Heat_Exchangers (is-a Heating_and_Cooling_Equipment))
     (defclass Plate_Heat_Exchanger (is-a Heat_Exchangers))
     (defclass Shell_and_Tube_Heat_Exchanger (is-a Heat_Exchangers))
  (defclass Phase_Change_Equipment (is-a
            Heating_and_Cooling_Equipment))
     (defclass Condenser (is-a Phase_Change_Equipment))
     (defclass Crystalliser (is-a Phase_Change_Equipment))
     (defclass Evaporator (is-a Phase_Change_Equipment))
     (defclass Vaporiser (is-a Phase_Change_Equipment))

(defclass Separation_Equipment (is-a equipment))
  (defclass Column (is-a Separation_Equipment))
     (defclass Distillation_Column (is-a Column))
     (defclass Absorption_Column (is-a Column))
     (defclass Other_Column (is-a Column))
  (defclass Decanting_vessel (is-a Separation_Equipment))
  (defclass Scrubber (is-a Separation_Equipment))
  (defclass Cyclone (is-a Separation_Equipment))
  (defclass Electro-Static_Precipitator (is-a Separation_Equipment))
  (defclass Filter (is-a Separation_Equipment))
  (defclass Hydro-Cyclone (is-a Separation_Equipment))
  (defclass Centrifuge (is-a Separation_Equipment))
  (defclass Adsorption (is-a Separation_Equipment))
  (defclass Settling_Pit (is-a Separation_Equipment))
  (defclass Strainer (is-a Separation_Equipment))
  (defclass Ion-Exchange_Bed (is-a Separation_Equipment))
  (defclass Dessicator (is-a Separation_Equipment))

(defclass Other_Equipment (is-a equipment))
  (defclass Pipeline (is-a Other_Equipment))

;*****************************************************************
;defclass CONDITIONS
;*****************************************************************
```

```
(defclass condition (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (default none))
      (slot synonym  (create-accessor read-write))
      (slot abbreviation  (create-accessor read-write)))

(defclass Operating_Mode (is-a condition))
   (defclass Batch (is-a Operating_Mode))
   (defclass Semi-Batch (is-a Operating_Mode))
   (defclass Continuous (is-a Operating_Mode))

(defclass Thermal_Behaviour (is-a condition))
   (defclass Superheat (is-a Thermal_Behaviour))
   (defclass Isothermal (is-a Thermal_Behaviour))
   (defclass Endothermal (is-a Thermal_Behaviour))
   (defclass Exothermal (is-a Thermal_Behaviour))

;*****************************************************************
;defclass REACTION
;*****************************************************************

(defclass reaction (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (default none))
      (slot synonym  (create-accessor read-write))
      (slot abbreviation  (create-accessor read-write)))

(defclass Reaction_Type (is-a reaction))
   (defclass General (is-a Reaction_Type))
   (defclass Polymerisation (is-a Reaction_Type))
   (defclass Catalytic (is-a Reaction_Type))
     (defclass Auto-Catalytic (is-a Catalytic))
   (defclass Anaerobic (is-a Reaction_Type))
   (defclass Van_De_Vusse (is-a Reaction_Type))
   (defclass Fermentation (is-a Reaction_Type))
   (defclass Parallel (is-a Reaction_Type))

(defclass Reaction_Order (is-a reaction))
   (defclass Zero (is-a Reaction_Order))
   (defclass First (is-a Reaction_Order))
   (defclass Second (is-a Reaction_Order))
   (defclass Higher (is-a Reaction_Order))

(defclass Reversibility (is-a reaction))
   (defclass Yes (is-a Reversibility))
   (defclass No (is-a Reversibility))

;*****************************************************************
;defclass PHASES
;*****************************************************************

(defclass phases (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (default none))
```

```
      (slot synonym  (create-accessor read-write))
      (slot abbreviation  (create-accessor read-write)))

(defclass Solid (is-a phases))

(defclass Liquid (is-a phases))

(defclass Vapour (is-a phases))

(defclass Solid-Liquid (is-a phases))

(defclass Solid-Vapour (is-a phases))

(defclass Liquid-Vapour (is-a phases))

(defclass Solid-Liquid-Vapour (is-a phases))

;*****************************************************************
;defclass TRANSFER
;*****************************************************************

(defclass transfer (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (default none))
      (slot synonym  (create-accessor read-write))
      (slot abbreviation  (create-accessor read-write)))

(defclass Mass_Transfer (is-a transfer))
   (defclass Diffussion (is-a Mass_Transfer))
   (defclass Absorption (is-a Mass_Transfer))

(defclass Heat_Transfer (is-a transfer))
   (defclass Convection (is-a Heat_Transfer))
   (defclass Conduction (is-a Heat_Transfer))
   (defclass Radiation (is-a Heat_Transfer))

(defclass Pipeline_Flow (is-a transfer))
   (defclass Laminar (is-a Pipeline_Flow))
   (defclass Chaotic (is-a Pipeline_Flow))
   (defclass Turbulent (is-a Pipeline_Flow))
   (defclass Plug/Slug (is-a Pipeline_Flow))

;*****************************************************************
;defclass CHEMICAL
;*****************************************************************

(defclass chemical (is-a USER)
      (role concrete)
      (pattern-match reactive)
      (multislot properties (create-accessor read-write)
            (visibility public)
            (default nil))
      (slot synonym  (create-accessor read-write) (default nil))
      (slot abbreviation  (create-accessor read-write) (default
nil)))

(definstances CHEMICALS
([acetic_acid_instance] of chemical
   (synonym ethanoicacid)
```

```
    (properties corrosive causes_burns))
)
```

IV.2. Constant, variable and equation object class declaration: classes1.clp

```
;*****************************************************
;  Author :          Graham Clark                     *
;  File Name:        classes1.clp                     *
;  Last Updated:     20/12/00                         *
;  Description:      Class declarations File          *
;*****************************************************

;*****************************************************
; defclass SIM_DECLARATION General declaration classes
;*****************************************************
(defclass SIM_DECLARATION (is-a USER)
 (role abstract)
)
  (defclass EQUATION (is-a SIM_DECLARATION)
   (role concrete)
   (pattern-match reactive)
   (slot Eqn (create-accessor read-write))
   (slot UpperBound (default "All") (create-accessor read-write))
   (slot LowerBound (default "All") (create-accessor read-write))
   (slot FixedBound (default "All") (create-accessor read-write))
   (slot Description (default "") (create-accessor read-write))
  )
  (defclass CONST_VAR_DEC (is-a SIM_DECLARATION)
   (role concrete)
   (pattern-match reactive)
   (slot Identifier (create-accessor read-write))
   (slot Unit (default "") (create-accessor read-write))
   (slot Include (default "Y") (create-accessor read-write))
   (slot Distributed (default "N/A") (create-accessor read-write))
   (slot Description (default "") (create-accessor read-write))
   (multislot ArrayVar (default 0 "none") (create-accessor read-
                  write))
   (slot Port (default 0) (create-accessor read-write))
   (multislot OutPorts (default 0 ) (create-accessor read-write))
 )
    (defclass VARIABLE (is-a CONST_VAR_DEC)
     (slot Set (default "unknown") (create-accessor read-write))
     (slot Value (default "unknown") (create-accessor read-write))
     (slot UpperBound (default "All") (create-accessor read-write))
     (slot LowerBound (default "All") (create-accessor read-write))
     (slot MathMethod (default "N/A") (create-accessor read-write))
     (slot Type (create-accessor read-write))
    )
    (defclass CONSTANT (is-a CONST_VAR_DEC)
     (slot Value (default 0.1) (create-accessor read-write))
    )
 (defclass PARTIALS (is-a SIM_DECLARATION)
    (role concrete)
    (slot Number (default "0") (create-accessor read-write))
    (slot Value (create-accessor read-write))
    (slot ReplacedWith (create-accessor read-write))
  )
```

IV.3. Global function declarations file: function.clp

```
;*****************************************************
;  Author :          Graham Clark                     *
;  File Name:        function.clp                     *
;  Last Updated:     20/12/00                         *
;  Description:      General functions file           *
;*****************************************************

(deffunction str-replace (?string ?old-str ?sub-str)
 (bind ?start (str-index ?old-str ?string))
 (bind ?length (str-length ?old-str))
 (bind ?totallength (str-length ?string))
  (bind ?newstring
   (str-cat (str-cat (sub-string 1 (- ?start 1) ?string) ?sub-str)
   (sub-string (+ ?start ?length) ?totallength ?string)))
  (return ?newstring)
)

(deffunction str-replace-all (?string ?old-str ?sub-str)
 (bind ?oldstring ?string)
 (bind ?newstring ?string)
 (bind ?start (str-index ?old-str ?oldstring))
 (while (= 0 (str-compare TRUE (integerp ?start))) do
  (bind ?totallength (str-length ?oldstring))
  (bind ?length (str-length ?old-str))
  (bind ?newstring
   (str-cat (str-cat (sub-string 1 (- ?start 1) ?oldstring) ?sub-str)
   (sub-string (+ ?start ?length) ?totallength ?oldstring)))
  (bind ?oldstring ?newstring)
  (bind ?start (str-index ?old-str ?oldstring))
  )
  (return ?newstring)
)

(deffunction str-count-all (?string ?count-str)
 (bind ?count 0)
 (bind ?totallength (str-length ?string))
 (bind ?length (str-length ?count-str))
 (bind ?new-str ?string)
 (while (= 0 (str-compare "TRUE" (integerp (str-index ?count-str
?new-str)))) do
  (bind ?count (+ 1 ?count))
  (bind ?start (str-index ?count-str ?new-str))
  (bind ?new-str (sub-string (+ ?start ?length) ?totallength ?new-
                  str))
  (bind ?totallength (str-length ?new-str))
 )
 (return ?count)
)

(deffunction check-eqn (?eqn ?name)
 (bind ?check (sub-string 1 2 ?eqn))
 (switch ?check
  (case "IF" then (print-eqn-if ?eqn))
  (case "WH" then (print-eqn-while ?eqn))
  (default (print-for-loop ?eqn ?name))
 )
 (return ?eqn)
)
```

```
(deffunction check-bdry-eqn (?eqn ?name)
 (bind ?check (sub-string 1 2 ?eqn))
 (switch ?check
  (case "IF" then (print-eqn-if ?eqn))
  (case "WH" then (print-eqn-while ?eqn))
  (default (print-bdry-for-loop ?eqn ?name))
 )
 (return ?eqn)
)
```

IV.4. Occurrence matrix code: occtable.clp

```
;*****************************************************************
;  Author :          Graham Clark                                 *
;  File Name:        occtable.clp                                 *
;  Last Updated:     20/12/00                                     *
;  Description:      Occurence Matrix creation and population     *
;*****************************************************************

(deffunction get-eqn-sql-query ())
(deffunction get-var-sql-query ())
(deffunction get-const-sql-query ())

(deffunction on-close (?frame)
 (format t "Closing frame.%n")
 (bind ?*matrix* 0)
 1)

(deffunction on-activate (?frame ?active)
 (if (> ?*matrix* 0) then (grid-on-activate ?*matrix* ?active))
)

(deffunction menu-command (?frame ?id)
 (switch ?id
; Help
  (case 200 then (message-box "Occurence Matrix
    Intelligent Modelling Interface for Process Simulators v0.03
    by Graham Clark (c) 1999" wxOK 1 0 "About IMIPS Prototype -
                  Occurence Matrix"))
; Quit
  (case 1 then
    (database-close ?*database*)
    (if (on-close ?frame) then (window-delete ?frame)))
 )
)


(deffunction occurence-matrix ()
 (bind ?*occ-matrix-frame* (frame-create 0 "IMIPS - Occurence Matrix"
                  -1 -1 1150 523))

 (window-add-callback ?*occ-matrix-frame* OnClose on-close)
 (window-add-callback ?*occ-matrix-frame* OnMenuCommand menu-command)
 (window-add-callback ?*occ-matrix-frame* OnActivate on-activate)

 (bind ?file-menu (menu-create))
 (menu-append ?file-menu 1 "&Quit")
 (bind ?help-menu (menu-create))
 (menu-append ?help-menu 200 "&About")

 (bind ?menu-bar (menu-bar-create))
 (menu-bar-append ?menu-bar ?file-menu "&File")
 (menu-bar-append ?menu-bar ?help-menu "&Help")

 (frame-set-menu-bar ?*occ-matrix-frame* ?menu-bar)

; Make a grid
 (bind ?*matrix* (grid-create ?*occ-matrix-frame* 0 0 1150 523))
 (grid-set-editable ?*matrix* 0)
 (grid-create-grid ?*matrix* 22 26)
 (grid-set-column-width ?*matrix* 0 300)
```

```
 (grid-set-label-alignment ?*matrix* wxHORIZONTAL wxCENTRE)

;Insert Variable Names
 (get-var-sql-query)
 (bind ?varcount 0)
 (loop-for-count (?var 1 25) do
  (grid-set-cell-alignment ?*matrix* wxCENTRE 0 ?var)
  (if (neq (recordset-get-char-data ?*recordset* (+ ?var 1)) "")
  then
   (if (eq 0 (str-length (recordset-get-char-data ?*recordset* (+ 21
                    (* ?var 8))))) then
     (if (< (str-length (recordset-get-char-data ?*recordset* (+ 20
                    (* ?var 8)))) 1)
     then
      (grid-set-cell-value ?*matrix* (recordset-get-char-data
                    ?*recordset* (+ ?var 1)) 0 (+ 1 ?varcount))
      (bind ?varcount (+ 1 ?varcount))
     )
    )
   )
  )

;Insert Equations
 (get-eqn-sql-query)
 (bind ?eqncount 0)
 (loop-for-count (?eqn 1 20) do
  (if (not (or (eq (recordset-get-char-data ?*recordset* (+ ?eqn 1))
                    "")
               (eq (recordset-get-char-data ?*recordset* (+ ?eqn 1))
                    "END")))
   then
    (if (not (or (> (str-length (recordset-get-char-data ?*recordset*
                    (+ 20 (* ?eqn 4)))) 0)
              (> (str-length (recordset-get-char-data ?*recordset* (+ 19
                    (* ?eqn 4)))) 0)
              (> (str-length (recordset-get-char-data ?*recordset* (+ 18
                    (* ?eqn 4)))) 0)))
    then
     (bind ?eqncount (+ ?eqncount 1))
     (grid-set-cell-value ?*matrix* (recordset-get-char-data
                    ?*recordset* (+ ?eqn 1)) ?eqncount 0)
    )
   )
  )

;Create Occurence Matrix
 (loop-for-count (?var 1 25) do
  (loop-for-count (?eqn 1 20) do
   (grid-set-cell-alignment ?*matrix* wxCENTRE ?eqn ?var)
   (if (> (str-length (grid-get-cell-value ?*matrix* 0 ?var)) 0)
    then
     (if (neq (str-index (grid-get-cell-value ?*matrix* 0 ?var) (grid-
                    get-cell-value ?*matrix* ?eqn 0)) FALSE)
     then
      (grid-set-cell-value ?*matrix* "*" ?eqn ?var)
     )
    )
  )
 )
 (get-const-sql-query)
; Count unknown variables
```

```
(bind ?var-number 0)
(loop-for-count (?var 1 ?varcount) do
 (if (> (str-length (grid-get-cell-value ?*matrix* 0 ?var)) 0)
 then
  (bind ?varcnt 1)
  (loop-for-count (?count 1 35) do
   (if (= 0 (str-compare "TRUE" (integerp (str-index "N"
            (recordset-get-char-data ?*recordset* (+ 34 (* ?count
                 4))))))))
   then
    (if (neq 0 (str-length (recordset-get-char-data ?*recordset* (+
                 1 ?count)))) then
    (if (= 0 (str-compare "TRUE" (integerp (str-index (recordset-
                 get-char-data ?*recordset* (+ 1 ?count))
                (grid-get-cell-value ?*matrix* 0 ?var)))))
        then
         (bind ?value (string-to-float (symbol-to-string
                 (recordset-get-char-data ?*recordset* (+ 33 (*
                 ?count 4))))))
         (bind ?varcnt (* ?varcnt ?value))
        )
        )
        )
    )
   (bind ?var-number (+ ?var-number ?varcnt))
   )
  )
; Count equations
 (bind ?eqn-number 0)
 (loop-for-count (?eqn 1 ?eqncount) do
  (if (> (str-length (grid-get-cell-value ?*matrix* ?eqn 0)) 0)
  then
(bind ?eqncnt 1)
(loop-for-count (?count 1 35) do
 (if (= 0 (str-compare "TRUE" (integerp (str-index "N"
          (recordset-get-char-data ?*recordset* (+ 34 (* ?count
               4))))))))
 then
 (if (neq 0 (str-length (recordset-get-char-data ?*recordset* (+ 1
               ?count)))) then
  (if (= 0 (str-compare "TRUE" (integerp (str-index (recordset-
               get-char-data ?*recordset* (+ 1 ?count))
              (grid-get-cell-value ?*matrix* ?eqn 0)))))
  then
   (bind ?value (string-to-float (symbol-to-string
               (recordset-get-char-data ?*recordset* (+ 33 (*
               ?count 4))))))
   (bind ?eqncnt (* ?eqncnt ?value))
  )
     )
     )
     )
   (bind ?eqn-number (+ ?eqn-number ?eqncnt))
   )
  )

  (get-var-sql-query)
; Count initial value variables
  (loop-for-count (?var 1 25) do
   (if (neq "" (recordset-get-char-data ?*recordset* (+ (* ?var 8)
19)))
```

```
  then
   (bind ?var-number (- ?var-number 1))
   )
 )

;Write number of eqns and variables for comparison
 (grid-set-cell-text-font ?*matrix* (font-create 12 wxROMAN wxNORMAL
            wxBOLD 0) (+ ?eqncount 1) 0)
 (grid-set-cell-value ?*matrix* (sym-cat ?eqn-number (sym-cat "
            Equations: "
   (sym-cat ?var-number " Unknown Variables")))
   (+ ?eqncount 1) 0)
 (grid-update-dimensions ?*matrix*)
 (window-centre ?*occ-matrix-frame* wxBOTH)
  (window-show ?*occ-matrix-frame* 1)
?*occ-matrix-frame*)
```

IV.5. IMIPS interface code: IMIPS.clp

```
;*****************************************************
;  Author :          Graham Clark                     *
;  File Name:        IMIPS.clp                        *
;  Last Updated:     20/12/00                         *
;  Description:      Main Interface File              *
;   Search tools adapted from Richard Illiffe's Work  *
;*****************************************************
(deffunction translate-gPROMS ())
(deffunction translate-SIMULINK ())
(deffunction create-problem-instances ())
(deffunction printout-eqn-sql-query ())
(deffunction printout-var-sql-query ())
(deffunction printout-const-sql-query ())
(deffunction search-button-callback ())
(deffunction construct-sql-query ())
(deffunction printout-sql-query ())
;*****************************************************
;Sizing callback function
;*****************************************************

(deffunction on-size (?id ?w ?h)
    (if (and (neq ?id 0) (neq ?*panel* 0) (neq ?*text-win* 0) (neq
            ?*boris* 0) (neq ?*doris* 0)) then
        (bind ?client-width (window-get-client-width ?id))
        (bind ?client-height (window-get-client-height ?id))
        (window-set-size ?*panel* 0 0 ?client-width ?client-height)
        (window-set-size ?*text-win* 0 0 ?client-width (- (/ ?client-
            height 3) 1))
        (window-set-size ?*boris* 0 (/ ?client-height 3) ?client-
            width (- (/ ?client-height 3) 1))
        (window-set-size ?*doris* 0 (* 2 (/ ?client-height 3))
            ?client-width (/ ?client-height 3))
    )
)

;*****************************************************
;Window closing callback function
;*****************************************************

(deffunction on-close (?frame)
    (format t "Closing frame.%n")
    (window-delete ?*start-frame*)
    (bind ?*panel* 0)
    (bind ?*text-win* 0)
    (bind ?*boris* 0)
    (bind ?*doris* 0)
    1)

;*****************************************************
;Function changes between panel 1 and panel 2 on button-press
;*****************************************************

(deffunction on-panel-change-button (?id) ""
    (bind ?event-id (panel-item-get-command-event))
    (if (eq "wxEVENT_TYPE_BUTTON_COMMAND" (event-get-event-type
            ?event-id)) then
        (bind ?name (window-get-name ?id))
        (bind ?parent (window-get-parent ?id))
        (window-show (window-get-parent ?id) 0)
```

```
            (switch ?name
                (case "next" then
                    (window-show ?*panel2* 1))
                (case "prev" then
                    (window-show ?*panel1* 1))
                (case "more" then
                    (window-show ?*panel3* 1))
            )
        )
    )

    (deffunction on-panel-more-button (?id) ""
        (bind ?event-id (panel-item-get-command-event))
        (if (eq "wxEVENT_TYPE_BUTTON_COMMAND" (event-get-event-type
            ?event-id)) then
            (bind ?name (window-get-name ?id))
            (bind ?parent (window-get-parent ?id))
            (printout-eqn-sql-query)
            (printout-const-sql-query)
            (printout-var-sql-query)
        )
    )

    ;*****************************************************************
    ;The next two functions pop-up choice menus for the hierarchies and
    ;store the ultimate selection in appropriate globals.
    ;*****************************************************************

    (deffunction popup (?choice1 ?name ?choice) ""
        (if (and (neq (length (class-subclasses (string-to-symbol
                ?choice1))) 0) (neq ?choice1 ?choice)) then
            (bind ?children (create$ (class-subclasses (string-to-symbol
                ?choice1))))
            (bind ?children (insert$ ?children 1 ?choice1))
            (bind ?choice2 (get-choice ?choice1 ?children ?*panel1*))
        else
            (bind ?*related-term* "")
            (assert (find-related ?choice1))
            (run)
            (switch ?name
                (case "equipment-list" then
                    (bind ?*equipment* ?choice1)
                    (text-set-value ?*equipment-searchbox* ?*equipment*)
                    (if (neq ?*related-term* "") then
                        (if (eq ?*match-related-terms* TRUE) then
                            (text-set-value ?*related-equipment-
                                searchbox* ?*related-term*)
                            (bind ?*related-equipment* ?*related-term*)
                        )
                    )
                    (return))
                (case "operating-mode-list" then
                    (bind ?*operating-mode* ?choice1)
                    (text-set-value ?*operating-mode-searchbox*
                        ?*operating-mode*)
                    (if (neq ?*related-term* "") then
                        (if (eq ?*match-related-terms* TRUE) then
                            (text-set-value ?*related-operating-mode-
                                searchbox* ?*related-term*)
                            (bind ?*related-operating-mode* ?*related-
                                term*)
```

```
                    )
                )
                (return))
            (case "thermal-behaviour-list" then
                (bind ?*thermal-behaviour* ?choice1)
                (text-set-value ?*thermal-behaviour-searchbox*
                        ?*thermal-behaviour*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-thermal-behaviour-
                                searchbox* ?*related-term*)
                        (bind ?*related-thermal-behaviour* ?*related-
                                term*)
                    )
                )
                (return))
            (case "reaction-type-list" then
                (bind ?*reaction-type* ?choice1)
                (text-set-value ?*reaction-type-searchbox*
                        ?*reaction-type*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-reaction-type-
                                searchbox* ?*related-term*)
                        (bind ?*related-reaction-type* ?*related-
                                term*)
                    )
                )
                (return))
            (case "reaction-order-list" then
                (bind ?*reaction-order* ?choice1)
                (text-set-value ?*reaction-order-searchbox*
                        ?*reaction-order*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-reaction-order-
                                searchbox* ?*related-term*)
                        (bind ?*related-reaction-order* ?*related-
                                term*)
                    )
                )
                (return))
            (case "reversible-list" then
                (bind ?*reversible* ?choice1)
                (text-set-value ?*reversible-searchbox*
                        ?*reversible*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-reversible-
                                searchbox* ?*related-term*)
                        (bind ?*related-reversible* ?*related-term*)
                    )
                )
                (return))
            (case "phase-list" then
                (bind ?*phases* ?choice1)
                (text-set-value ?*phases-searchbox* ?*phases*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-phases-searchbox*
                                ?*related-term*)
```

```
                (bind ?*related-phases* ?*related-term*)
            )
        )
        (return))
    (case "mass-transfer-list" then
        (bind ?*mass-transfer* ?choice1)
        (text-set-value ?*mass-transfer-searchbox* ?*mass-
                transfer*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-mass-transfer-
                        searchbox* ?*related-term*)
                (bind ?*related-mass-transfer* ?*related-
                        term*)
            )
        )
        (return))
    (case "heat-transfer-list" then
        (bind ?*heat-transfer* ?choice1)
        (text-set-value ?*heat-transfer-searchbox* ?*heat-
                transfer*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-heat-transfer-
                        searchbox* ?*related-term*)
                (bind ?*related-heat-transfer* ?*related-
                        term*)
            )
        )
        (return))
    (case "pipeline-flow-list" then
        (bind ?*pipeline-flow* ?choice1)
        (text-set-value ?*pipeline-flow-searchbox*
                ?*pipeline-flow*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-pipeline-flow-
                        searchbox* ?*related-term*)
                (bind ?*related-pipeline-flow* ?*related-
                        term*)
            )
        )
        (return))
    (case "chemical-list" then
        (bind ?*chemical* ?choice1)
        (text-set-value ?*chemical-searchbox* ?*chemical*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-chemical-searchbox*
                        ?*related-term*)
                (bind ?*related-chemical* ?*related-term*)
            )
        )
        (return))
    )
    (popup ?choice2 ?name ?choice1)
)

(deffunction on-list-choice (?id)
    (bind ?name (window-get-name ?id))
```

```
        (bind ?choice (list-box-get-string-selection ?id))
        (if (neq (length (class-subclasses (string-to-symbol ?choice)))
                0) then
            (bind ?children (create$ (class-subclasses (string-to-symbol
                ?choice))))
            (bind ?children (insert$ ?children 1 (string-to-symbol
                ?choice)))
            (bind ?choice1 (get-choice ?choice ?children ?*panel*))
    else
            (bind ?*related-term* "")
            (assert (find-related ?choice))
            (run)
            (switch ?name
               (case "equipment-list" then
                (bind ?*equipment* ?choice)
                (text-set-value ?*equipment-searchbox* ?*equipment*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-equipment-
                            searchbox* ?*related-term*)
                        (bind ?*related-equipment* ?*related-term*)
                    )
                )
                (return))
            (case "operating-mode-list" then
                (bind ?*operating-mode* ?choice)
                (text-set-value ?*operating-mode-searchbox*
                        ?*operating-mode*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-operating-mode-
                            searchbox* ?*related-term*)
                        (bind ?*related-operating-mode* ?*related-
                            term*)
                    )
                )
                (return))
            (case "thermal-behaviour-list" then
                (bind ?*thermal-behaviour* ?choice)
                (text-set-value ?*thermal-behaviour-searchbox*
                        ?*thermal-behaviour*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-thermal-behaviour-
                            searchbox* ?*related-term*)
                        (bind ?*related-thermal-behaviour* ?*related-
                            term*)
                    )
                )
                (return))
            (case "reaction-type-list" then
                (bind ?*reaction-type* ?choice)
                (text-set-value ?*reaction-type-searchbox*
                        ?*reaction-type*)
                (if (neq ?*related-term* "") then
                    (if (eq ?*match-related-terms* TRUE) then
                        (text-set-value ?*related-reaction-type-
                            searchbox* ?*related-term*)
                        (bind ?*related-reaction-type* ?*related-
                            term*)
                    )
```

```
        )
        (return))
    (case "reaction-order-list" then
        (bind ?*reaction-order* ?choice)
        (text-set-value ?*reaction-order-searchbox*
                ?*reaction-order*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-reaction-order-
                      searchbox* ?*related-term*)
                (bind ?*related-reaction-order* ?*related-
                      term*)
            )
        )
        (return))
    (case "reversible-list" then
        (bind ?*reversible* ?choice)
        (text-set-value ?*reversible-searchbox*
                ?*reversible*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-reversible-
                      searchbox* ?*related-term*)
                (bind ?*related-reversible* ?*related-term*)
            )
        )
        (return))
    (case "phase-list" then
        (bind ?*phases* ?choice)
        (text-set-value ?*phases-searchbox* ?*phases*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-phases-searchbox*
                      ?*related-term*)
                (bind ?*related-phases* ?*related-term*)
            )
        )
        (return))
    (case "mass-transfer-list" then
        (bind ?*mass-transfer* ?choice)
        (text-set-value ?*mass-transfer-searchbox* ?*mass-
                transfer*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-mass-transfer-
                      searchbox* ?*related-term*)
                (bind ?*related-mass-transfer* ?*related-
                      term*)
            )
        )
        (return))
    (case "heat-transfer-list" then
        (bind ?*heat-transfer* ?choice)
        (text-set-value ?*heat-transfer-searchbox* ?*heat-
                transfer*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-heat-transfer-
                      searchbox* ?*related-term*)
                (bind ?*related-heat-transfer* ?*related-
                      term*)
```

```
            )
        )
        (return))
    (case "pipeline-flow-list" then
        (bind ?*pipeline-flow* ?choice)
        (text-set-value ?*pipeline-flow-searchbox*
                ?*pipeline-flow*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-pipeline-flow-
                        searchbox* ?*related-term*)
                (bind ?*related-pipeline-flow* ?*related-
                        term*)
            )
        )
        (return))
    (case "chemical-list" then
        (bind ?*chemical* ?choice)
        (text-set-value ?*chemical-searchbox* ?*chemical*)
        (if (neq ?*related-term* "") then
            (if (eq ?*match-related-terms* TRUE) then
                (text-set-value ?*related-chemical-searchbox*
                        ?*related-term*)
                (bind ?*related-chemical* ?*related-term*)
            )
        )
        (return))
    (case "temp-list" then
        (bind ?*t-unit* ?choice)
        (return))
    (case "pres-list" then
        (bind ?*p-unit* ?choice)
                (return))
            )
        )
        (popup ?choice1 ?name ?choice)
)

;*****************************************************************
;Generic function to append string values to listbox
;*****************************************************************

(deffunction append-to-list (?local ?global)
    (bind ?num (length ?global))
    (loop-for-count (?count 1 ?num) do
        (list-box-append ?local (str-cat (nth$ ?count ?global)))
    )
)

;*****************************************************************
;Initialises database
;*****************************************************************

(deffunction initialise-database () ""
  (bind ?*database* (database-create))
  (if (eq 0 (database-open ?*database* ?*database-name*)) then
   (bind ?msg (database-get-error-message ?*database*))
   (printout t ?msg crlf)
   (return 0)
  )
 (bind ?*recordset* (recordset-create ?*database*
```

```
        "wxOPEN_TYPE_SNAPSHOT"))
)

;*****************************************************************
; The following two functions are for constructing a SQL query based
; on the user selected keywords and options. get-include-terms
; retrieves any of multiple descriptives, divided by slashes.
;*****************************************************************

(deffunction get-include-terms (?key ?term)
   (bind ?symbol-term (string-to-symbol ?term))
   (bind ?key-string (str-cat ?key " like '%/" ?term "/%'"))
   (if (class-existp ?symbol-term) then
       (if (and (eq ?*match-children* TRUE) (neq (implode$ (class-
        subclasses ?symbol-term)) "")) then
           (bind ?term-subclasses (class-subclasses ?symbol-term
                 inherit))
           (bind ?cont 1)
           (loop-for-count (length$ ?term-subclasses)
              (bind ?key-string (str-cat ?key-string " OR " ?key "
                  like '%/" (nth$ ?cont ?term-subclasses)
                 "/%'"))
            (bind ?cont (+ ?cont 1))
            )
       )
       (if (eq ?*match-parent* TRUE) then
           (bind ?term-superclasses (class-superclasses ?symbol-term
                 inherit))
           (bind ?key-string (str-cat ?key-string " OR " ?key " like
                 '%/" (nth$ 1 ?term-superclasses) "/%'"))
       )
   )
 (bind ?key-string (str-cat "(" ?key-string ")"))
 (return ?key-string)
)

;*****************************************************************
;Function retrieves single terms which are not divided by slashes
;*****************************************************************

(deffunction get-include-terms2 (?key ?term)
   (bind ?symbol-term (string-to-symbol ?term))
  (printout t ?term crlf)
   (bind ?key-string (str-cat ?key " like '%" ?term "%'"))
  (printout t ?key-string crlf)
   (if (class-existp ?symbol-term) then
       (if (and (eq ?*match-children* TRUE) (neq (implode$ (class-
        subclasses ?symbol-term)) "")) then
           (bind ?term-subclasses (class-subclasses ?symbol-term
                 inherit))
           (bind ?cont 1)
           (loop-for-count (length$ ?term-subclasses)
              (bind ?key-string (str-cat ?key-string " OR " ?key "
                  like '%" (nth$ ?cont ?term-subclasses)
                 "%'"))
            (bind ?cont (+ ?cont 1))
            )
   )
   (if (eq ?*match-parent* TRUE) then
       (bind ?term-superclasses (class-superclasses ?symbol-term
             inherit))
```

```
            (bind ?key-string (str-cat ?key-string " OR " ?key " like
                  '%" (nth$ 1 ?term-superclasses) "%'"))
        )
    )
    (bind ?key-string (str-cat "(" ?key-string ")"))
  (printout t ?key-string crlf)
  (return ?key-string)
)

;*****************************************************************

(deffunction related-sql (?key ?term ?related) ""
    (bind ?tmp1 (get-include-terms ?key ?term))
    (bind ?tmp2 (get-include-terms ?key ?related))
    (bind ?tmp3 (str-cat (sub-string 1 (- (str-length  ?tmp1) 1)
          ?tmp1) " OR "
               (sub-string 2 (str-length ?tmp2) ?tmp2)))
    (if (eq ?*match-related-terms* TRUE) then
        (if (neq ?*sql* "") then
            (bind ?*sql* (str-cat ?*sql* " AND " ?tmp3))
        else
            (bind ?*sql* ?tmp3)
        )
    else
        (if (neq ?term "") then
            (if (neq ?*sql* "") then
                (bind ?*sql* (str-cat ?*sql* " AND " ?tmp1))
            else
                (bind ?*sql* ?tmp1)
            )
        )
    )
)

;*****************************************************************

;Numerical Search Type - SR overlap, SR internal, CR internal, All
;Temperature
(deffunction Temp-SRO-check-callback (?id)
  (if (eq ?*Temp-SRO-search* FALSE) then
      (bind ?*Temp-SRO-search* TRUE)
     else
      (bind ?*Temp-SRO-search* FALSE)
  )
  (if (eq ?*Temp-SRO-search* TRUE) then
    (bind ?*Temp-ALL-search* FALSE)
    (check-box-set-value ?*Temp-All-check* 0)
  )
)

(deffunction Temp-SRI-check-callback (?id)
  (if (eq ?*Temp-SRI-search* FALSE) then
      (bind ?*Temp-SRI-search* TRUE)
     else
      (bind ?*Temp-SRI-search* FALSE)
  )
  (if (eq ?*Temp-SRI-search* TRUE) then
    (bind ?*Temp-ALL-search* FALSE)
    (check-box-set-value ?*Temp-All-check* 0)
  )
)
```

```
(deffunction Temp-CRI-check-callback (?id)
  (if (eq ?*Temp-CRI-search* FALSE) then
      (bind ?*Temp-CRI-search* TRUE)
     else
      (bind ?*Temp-CRI-search* FALSE)
  )
  (if (eq ?*Temp-CRI-search* TRUE) then
    (bind ?*Temp-ALL-search* FALSE)
    (check-box-set-value ?*Temp-All-check* 0)
  )
)

(deffunction Temp-ALL-check-callback (?id)
  (if (eq ?*Temp-ALL-search* FALSE) then
      (bind ?*Temp-ALL-search* TRUE)
     else
      (bind ?*Temp-ALL-search* FALSE)
  )
  (if (eq ?*Temp-ALL-search* TRUE) then
    (bind ?*Temp-SRO-search* FALSE)
    (check-box-set-value ?*Temp-SRo-check* 0)
    (bind ?*Temp-SRI-search* FALSE)
    (check-box-set-value ?*Temp-SRi-check* 0)
    (bind ?*Temp-CRI-search* FALSE)
    (check-box-set-value ?*Temp-CRi-check* 0)
  )
)

(deffunction Temp-OK-search-button-callback (?id)
 (window-delete ?*temp-search-frame*)
 (initialise-database)
 (construct-sql-query)
 (printout-sql-query)
)

(deffunction Pres-OK-search-button-callback (?id)
 (window-delete ?*pres-search-frame*)
 (initialise-database)
 (construct-sql-query)
 (printout-sql-query)
)

(deffunction Temp-Range-Search () ""
  (bind ?*temp-low* (integer (string-to-float ?*temp-low*)))
  (bind ?*temp-high* (integer (string-to-float ?*temp-high*)))
  (bind ?*temp-search-frame* (frame-create 0 "Temperature Search
     Options" 220 200 300 250))
  (frame-create-status-line ?*temp-search-frame*)
  (frame-set-status-text ?*temp-search-frame* "Please select the
     search/s to use")
  (bind ?*temp-search-panel* (panel-create ?*temp-search-frame* 0 0
     400 400))
  (panel-set-label-position ?*temp-search-panel* wxVERTICAL)
  (bind ?*Temp-SRo-check* (check-box-create ?*temp-search-panel*
     Temp-SRO-check-callback "Search Range Overlaps Case Data Range"
     -1 -1 -1 -1 "" ""))
    (check-box-set-value ?*Temp-SRo-check* 0)
  (panel-new-line ?*temp-search-panel*)
  (bind ?*Temp-SRi-check* (check-box-create ?*temp-search-panel*
     Temp-SRI-check-callback "Search Range is inside Case Data
```

```
    Range" -1 -1 -1 -1 "" ""))
  (check-box-set-value ?*Temp-SRi-check* 0)
(panel-new-line ?*temp-search-panel*)
(bind ?*Temp-CRi-check* (check-box-create ?*temp-search-panel*
    Temp-CRI-check-callback "Case Data Range is inside Search
    Range" -1 -1 -1 -1 "" ""))
  (check-box-set-value ?*Temp-CRi-check* 0)
(panel-new-line ?*temp-search-panel*)
(bind ?*Temp-All-check* (check-box-create ?*temp-search-panel*
    Temp-ALL-check-callback "All Options" -1 -1 -1 -1 "" ""))
  (check-box-set-value ?*Temp-All-check* 1)
(panel-new-line ?*temp-search-panel*)
(bind ?OK-search-button (button-create ?*temp-search-panel* Temp-
    OK-search-button-callback "OK" -1 -1 -1 -1 "" "OK"))
  (window-show ?*temp-search-frame* 1)
)

;Pressure
(deffunction Pres-SRO-check-callback (?id)
  (if (eq ?*Pres-SRO-search* FALSE) then
      (bind ?*Pres-SRO-search* TRUE)
     else
      (bind ?*Pres-SRO-search* FALSE)
  )
  (if (eq ?*Pres-SRO-search* TRUE) then
    (bind ?*Pres-ALL-search* FALSE)
    (check-box-set-value ?*Pres-All-check* 0)
  )
)

(deffunction Pres-SRI-check-callback (?id)
  (if (eq ?*Pres-SRI-search* FALSE) then
      (bind ?*Pres-SRI-search* TRUE)
     else
      (bind ?*Pres-SRI-search* FALSE)
  )
  (if (eq ?*Pres-SRI-search* TRUE) then
    (bind ?*Pres-ALL-search* FALSE)
    (check-box-set-value ?*Pres-All-check* 0)
  )
)

(deffunction Pres-CRI-check-callback (?id)
  (if (eq ?*Pres-CRI-search* FALSE) then
      (bind ?*Pres-CRI-search* TRUE)
     else
      (bind ?*Pres-CRI-search* FALSE)
  )
  (if (eq ?*Pres-CRI-search* TRUE) then
    (bind ?*Pres-ALL-search* FALSE)
    (check-box-set-value ?*Pres-All-check* 0)
  )
)

(deffunction Pres-ALL-check-callback (?id)
  (if (eq ?*Pres-ALL-search* FALSE) then
      (bind ?*Pres-ALL-search* TRUE)
     else
      (bind ?*Pres-ALL-search* FALSE)
  )
  (if (eq ?*Pres-ALL-search* TRUE) then
```

```
      (bind ?*Pres-SRO-search* FALSE)
      (check-box-set-value ?*Pres-SRo-check* 0)
      (bind ?*Pres-SRI-search* FALSE)
      (check-box-set-value ?*Pres-SRi-check* 0)
      (bind ?*Pres-CRI-search* FALSE)
      (check-box-set-value ?*Pres-CRi-check* 0)
    )
)

(deffunction Pres-Range-Search () ""
  (bind ?*pres-low* (integer (string-to-float ?*pres-low*)))
  (bind ?*pres-high* (integer (string-to-float ?*pres-high*)))
  (bind ?*pres-search-frame* (frame-create 0 "Pressure Search
      Options" 220 200 300 250))
  (frame-create-status-line ?*pres-search-frame*)
  (frame-set-status-text ?*pres-search-frame* "Please select the
      search/s to use")
  (bind ?*pres-search-panel* (panel-create ?*pres-search-frame* 0 0
      400 400))
  (panel-set-label-position ?*pres-search-panel* wxVERTICAL)
  (bind ?*Pres-SRo-check* (check-box-create ?*pres-search-panel*
      Pres-SRO-check-callback "Search Range Overlaps Case Data Range"
      -1 -1 -1 -1 "" ""))
    (check-box-set-value ?*Pres-SRo-check* 0)
  (panel-new-line ?*pres-search-panel*)
  (bind ?*Pres-SRi-check* (check-box-create ?*pres-search-panel*
      Pres-SRI-check-callback "Search Range is inside Case Data
      Range" -1 -1 -1 -1 "" ""))
    (check-box-set-value ?*Pres-SRi-check* 0)
  (panel-new-line ?*pres-search-panel*)
  (bind ?*Pres-CRi-check* (check-box-create ?*pres-search-panel*
      Pres-CRI-check-callback "Case Data Range is inside Search
        Range" -1 -1 -1 -1 "" ""))
      (check-box-set-value ?*Pres-CRi-check* 0)
    (panel-new-line ?*pres-search-panel*)
    (bind ?*Pres-All-check* (check-box-create ?*pres-search-panel*
        Pres-ALL-check-callback "All Options" -1 -1 -1 -1 "" ""))
      (check-box-set-value ?*Pres-All-check* 1)
    (panel-new-line ?*pres-search-panel*)
    (bind ?range-search-button (button-create ?*pres-search-panel*
        Pres-OK-search-button-callback
      "OK" -1 -1 -1 -1 "" "OK"))
    (window-show ?*pres-search-frame* 1)
)

(deffunction Numerical-Search () ""
  (if (or (neq ?*temp-low* "") (neq ?*temp-high* "")) then
    (Temp-Range-Search))
  (if (or (neq ?*pres-low* "") (neq ?*pres-high* "")) then
      (Pres-Range-Search))
    (bind ?*Num-Search* 1)
)

;*****************************************************************

(deffunction construct-sql-query () ""
    (bind ?*record-number* (text-get-value ?*record-number-
      searchbox*))
    (bind ?*description-name* (text-get-value ?*description-
      searchbox*))
    (bind ?*temp-tolerance* (text-get-value ?*temp-tolerance-
```

```
  searchbox*))
(bind ?*pres-tolerance* (text-get-value ?*pres-tolerance-
  searchbox*))
(bind ?*sql* "")
(if (neq ?*equipment* "") then
    (related-sql equipment ?*equipment* ?*related-equipment*)
)
(if (neq ?*operating-mode* "") then
    (related-sql operating_mode ?*operating-mode* ?*related-
       operating-mode*)
)
(if (neq ?*thermal-behaviour* "") then
    (related-sql thermal_behaviour ?*thermal-behaviour*
       ?*related-thermal-behaviour*)
)
(if (neq ?*reaction-type* "") then
    (related-sql reaction_type ?*reaction-type* ?*related-
       reaction-type*)
)
(if (neq ?*reaction-order* "") then
    (related-sql reaction_order ?*reaction-order* ?*related-
       reaction-order*)
)
(if (neq ?*reversible* "") then
    (related-sql reversible ?*reversible* ?*related-reversible*)
)
(if (neq ?*phases* "") then
    (related-sql phases ?*phases* ?*related-phases*)
)
(if (neq ?*mass-transfer* "") then
    (related-sql mass_transfer ?*mass-transfer* ?*related-mass-
       transfer*)
)
(if (neq ?*heat-transfer* "") then
    (related-sql heat_transfer ?*heat-transfer* ?*related-heat-
       transfer*)
)
(if (neq ?*pipeline-flow* "") then
    (related-sql pipeline_flow ?*pipeline-flow* ?*related-
       pipeline-flow*)
)
(if (neq ?*chemical* "") then
    (related-sql chemical ?*chemical* ?*related-chemical*)
)
(if (neq ?*record-number* "") then
    (if (neq ?*sql* "") then
        (bind ?*sql* (str-cat ?*sql* " AND IDsearch = " ?*record-
             number*))
    else
        (bind ?*sql* (str-cat "IDsearch = " ?*record-number*))
    )
)
(if  (neq ?*description-name* "") then
    (if (neq ?*sql* "") then
        (bind ?*sql* (str-cat ?*sql* " AND (Description1 like '%"
             ?*description-name*
             "%' OR Description2 like '%" ?*description-name*
             "%' OR Description3 like '%" ?*description-name*
             "%' OR Description4 like '%" ?*description-name*
             "%'))"))
    else
```

```
                (bind ?*sql* (str-cat "(Description1 like '%"
                      ?*description-name*
                      "%' OR Description2 like '%" ?*description-name*
                      "%' OR Description3 like '%" ?*description-name*
                      "%' OR Description4 like '%" ?*description-name*
                      "%')"))
            )
        )
        (bind ?*temp-tolerance* (/ (string-to-float ?*temp-tolerance*)
          100))
        (bind ?*pres-tolerance* (/ (string-to-float ?*pres-tolerance*)
          100))
    ;Temperature
        (if (and (neq ?*temp-low* "") (neq ?*temp-high* "")) then
          (if (eq ?*Temp-ALL-search* TRUE) then
            (if (neq ?*sql* "") then
              (bind ?*sql* (str-cat ?*sql* " AND "
                           "((Temperature_Low <= " ?*temp-low*
                           " <= Temperature_High) OR (Temperature_Low
                           <= " ?*temp-high*
                           " <= Temperature_High) OR (Temperature_Low >
                           " ?*temp-low*
                           " AND Temperature_High < " ?*temp-high* "))"
              ))
            else
              (bind ?*sql* (str-cat ""
                           "((Temperature_Low <= " ?*temp-low*
                           " <= Temperature_High) OR (Temperature_Low <=
                           " ?*temp-high*
                           " <= Temperature_High) OR (Temperature_Low >
                           " ?*temp-low*
                           " AND Temperature_High < " ?*temp-high* "))"
              ))
            )
            (bind ?*Temp-SRO-search* FALSE)
            (bind ?*Temp-SRI-search* FALSE)
            (bind ?*Temp-CRI-search* FALSE)
          )
          (if (eq ?*Temp-SRO-search* TRUE) then
            (if (neq ?*sql* "") then
              (bind ?*sql* (str-cat ?*sql* " AND "
                           "((Temperature_Low <= " ?*temp-low*
                           " <= Temperature_High) OR (Temperature_Low
                           <= " ?*temp-high*
                           " <= Temperature_High) AND NOT
                           ((Temperature_Low <= " ?*temp-low*
                           ") AND (" ?*temp-high* " <=
                           Temperature_High)))"
              ))
            else
              (bind ?*sql* (str-cat ""
                           "((Temperature_Low <= " ?*temp-low*
                           " <= Temperature_High) OR (Temperature_Low <=
                           " ?*temp-high*
                           " <= Temperature_High) AND NOT
                           ((Temperature_Low <= " ?*temp-low*
                           ") AND (" ?*temp-high* " <=
                           Temperature_High)))"
              ))
            )
          )
```

```
(if (eq ?*Temp-SRI-search* TRUE) then
  (if (neq ?*sql* "") then
    (bind ?*sql* (str-cat ?*sql* " AND "
                  "((Temperature_Low <= " ?*temp-low*
                  ") AND (" ?*temp-high* " <=
                  Temperature_High))"
    ))
  else
    (bind ?*sql* (str-cat ""
                  "((Temperature_Low <= " ?*temp-low*
                  ") AND (" ?*temp-high* " <=
                  Temperature_High))"
    ))
  )
  )
  (if (eq ?*Temp-CRI-search* TRUE) then
    (if (neq ?*sql* "") then
      (bind ?*sql* (str-cat ?*sql* " AND "
                    "((" ?*temp-low* " <= Temperature_Low)"
                    " AND (Temperature_High <= " ?*temp-high*
                    "))"
      ))
    else
      (bind ?*sql* (str-cat ""
                    "((" ?*temp-low* " <= Temperature_Low)"
                    " AND (Temperature_High <= " ?*temp-high*
                    "))"
      ))
    )
  )
else
  (if (neq ?*temp-low* "") then
    (bind ?*temp-low* (integer (string-to-float ?*temp-
low*)))
    (if (neq ?*sql* "") then
      (bind ?*sql* (str-cat ?*sql* " AND "
                    "((Temperature_Low <= "
                    ?*temp-low*
                    " <= Temperature_High) OR (Temperature_Low <= "
                    (* ?*temp-low* (- 1 ?*temp-tolerance*))
                    " <= Temperature_High) OR (Temperature_Low <= "
                    (* ?*temp-low* (+ 1 ?*temp-tolerance*))
                    " <= Temperature_High))"
      ))
    else
      (bind ?*sql* (str-cat ""
                    "((Temperature_Low <= "
                    ?*temp-low*
                    " <= Temperature_High) OR (Temperature_Low <= "
                    (* ?*temp-low* (- 1 ?*temp-tolerance*))
                    " <= Temperature_High) OR (Temperature_Low <= "
                    (* ?*temp-low* (+ 1 ?*temp-tolerance*))
                    " <= Temperature_High))"
      ))
    )
  )
else
  (if (neq ?*temp-high* "") then
    (bind ?*temp-high* (integer (string-to-float ?*temp-
          high*)))
    (if (neq ?*sql* "") then
      (bind ?*sql* (str-cat ?*sql* " AND "
```

```
                "((Temperature_Low <= "
                ?*temp-high*
                " <= Temperature_High) OR (Temperature_Low <= "
                (* ?*temp-high* (- 1 ?*temp-tolerance*))
                " <= Temperature_High) OR (Temperature_Low <= "
                (* ?*temp-high* (+ 1 ?*temp-tolerance*))
                " <= Temperature_High))"
        ))
    else
        (bind ?*sql* (str-cat ""
                "((Temperature_Low <= "
                ?*temp-high*
                " <= Temperature_High) OR (Temperature_Low <= "
                (* ?*temp-high* (- 1 ?*temp-tolerance*))
                " <= Temperature_High) OR (Temperature_Low <= "
                (* ?*temp-high* (+ 1 ?*temp-tolerance*))
                " <= Temperature_High))"
        ))
    )
  )
;Pressure
  (if (and (neq ?*pres-low* "") (neq ?*pres-high* "")) then
      (if (eq ?*Pres-ALL-search* TRUE) then
        (if (neq ?*sql* "") then
          (bind ?*sql* (str-cat ?*sql* " AND "
                        "((Pressure_Low <= " ?*pres-low*
                        " <= Pressure_High) OR (Pressure_Low <= "
                        ?*pres-high*
                        " <= Pressure_High) OR (Pressure_Low > "
                        ?*pres-low*
                        " AND Pressure_High < " ?*pres-high* "))"
          ))
        else
          (bind ?*sql* (str-cat ""
                        "((Pressure_Low <= " ?*pres-low*
                        " <= Pressure_High) OR (Pressure_Low <= "
                        ?*pres-high*
                        " <= Pressure_High) OR (Pressure_Low > "
                        ?*pres-low*
                        " AND Pressure_High < " ?*pres-high* "))"
          ))
        )
        (bind ?*Pres-SRO-search* FALSE)
        (bind ?*Pres-SRI-search* FALSE)
        (bind ?*Pres-CRI-search* FALSE)
    )
    (if (eq ?*Pres-SRO-search* TRUE) then
      (if (neq ?*sql* "") then
        (bind ?*sql* (str-cat ?*sql* " AND "
                        "((Pressure_Low <= " ?*pres-low*
                        " <= Pressure_High) OR (Pressure_Low <= "
                        ?*pres-high*
                        " <= Pressure_High) AND NOT ((Pressure_Low <=
                        " ?*pres-low*
                        ") AND (" ?*pres-high* " <= Pressure_High)))"
      ))
    else
      (bind ?*sql* (str-cat ""
                        "((Pressure_Low <= " ?*pres-low*
```

```
                                    " <= Pressure_High) OR (Pressure_Low <= "
                                    ?*pres-high*
                                    " <= Pressure_High) AND NOT ((Pressure_Low <=
                                    " ?*pres-low*
                                    ") AND (" ?*pres-high* " <= Pressure_High)))"
                    ))
                  )
                )
                (if (eq ?*Pres-SRI-search* TRUE) then
                  (if (neq ?*sql* "") then
                    (bind ?*sql* (str-cat ?*sql* " AND "
                                    "((Pressure_Low <= " ?*pres-low*
                                    ") AND (" ?*pres-high* " <= Pressure_High))"
                    ))
                    else
                      (bind ?*sql* (str-cat ""
                                    "((Pressure_Low <= " ?*pres-low*
                                    ") AND (" ?*pres-high* " <= Pressure_High))"
                      ))
                  )
                )
                (if (eq ?*Pres-CRI-search* TRUE) then
                  (if (neq ?*sql* "") then
                    (bind ?*sql* (str-cat ?*sql* " AND "
                                    "((" ?*pres-low* " <= Pressure_Low)"
                                    " AND (Pressure_High <= " ?*pres-high* "))"
                    ))
                  else
                    (bind ?*sql* (str-cat ""
                                    "((" ?*pres-low* " <= Pressure_Low)"
                                    " AND (Pressure_High <= " ?*pres-high* "))"
                    ))
                  )
                )
            else
                (if (neq ?*pres-low* "") then
                    (bind ?*pres-low* (integer (string-to-float ?*pres-
                          low*)))
                    (if (neq ?*sql* "") then
                      (bind ?*sql* (str-cat ?*sql* " AND "
                                    "((Pressure_Low <= "
                                    ?*pres-low*
                                    " <= Pressure_High) OR (Pressure_Low <= "
                                    (* ?*pres-low* (- 1 ?*pres-tolerance*))
                                    " <= Pressure_High) OR (Pressure_Low <= "
                                    (* ?*pres-low* (+ 1 ?*pres-tolerance*))
                                    " <= Pressure_High))"
                      ))
                    else
                      (bind ?*sql* (str-cat ""
                                    "((Pressure_Low <= "
                                    ?*pres-low*
                                    " <= Pressure_High) OR (Pressure_Low <= "
                                    (* ?*pres-low* (- 1 ?*pres-tolerance*))
                                    " <= Pressure_High) OR (Pressure_Low <= "
                                    (* ?*pres-low* (+ 1 ?*pres-tolerance*))
                                    " <= Pressure_High))"
                      ))
                )
            else
                (if (neq ?*pres-high* "") then
```

```
(bind ?*pres-high* (integer (string-to-float ?*pres-
        high*)))
(if (neq ?*sql* "") then
  (bind ?*sql* (str-cat ?*sql* " AND "
        "((Pressure_Low <= "
        ?*pres-high*
        " <= Pressure_High) OR (Pressure_Low <= "
        (* ?*pres-high* (- 1 ?*pres-tolerance*))
        " <= Pressure_High) OR (Pressure_Low <= "
        (* ?*pres-high* (+ 1 ?*pres-tolerance*))
        " <= Pressure_High))"
  ))
else
  (bind ?*sql* (str-cat ""
        "((Pressure_Low <= "
        ?*pres-high*
        " <= Pressure_High) OR (Pressure_Low <= "
        (* ?*pres-high* (- 1 ?*pres-tolerance*))
        " <= Pressure_High) OR (Pressure_Low <= "
        (* ?*pres-high* (+ 1 ?*pres-tolerance*))
        " <= Pressure_High))"
                    ))
                )
            )
        )
    )
(printout t ?*sql* crlf)
)

;*****************************************************************
; display the current record set values on to the various window
boxes
;*****************************************************************

(deffunction get-display-string (?recordset ?field)
    (bind ?record-string (recordset-get-char-data ?recordset ?field))
    (bind ?string-length (length ?record-string))
    (if (> ?string-length 2) then
        (bind ?record-string (sub-string 2 (- ?string-length 1)
            ?record-string))
     else
        (bind ?record-string "")
    )
)

;*****************************************************************
**

(deffunction display-record () ""
    (menu-enable ?*data-menu* 11 1)
    (menu-enable ?*data-menu* 12 1)
    (menu-enable ?*report-menu* 21 1)
    (menu-enable ?*report-menu* 22 1)
    (menu-enable ?*report-menu* 23 1)
    (menu-enable ?*report-menu* 24 1)
    (menu-enable ?*translate-menu* 30 1)
    (menu-enable ?*translate-menu* 31 1)
    (if (neq ?*retrieved-number* 0) then
        (text-set-value ?*number-textbox*  (recordset-get-char-data
            ?*recordset* 0))
        (bind ?*main-ID* (recordset-get-int-data ?*recordset* 1))
```

```
        (bind ?*problem-title* (recordset-get-char-data ?*recordset*
2))
        (text-set-value ?*equipment-textbox* (get-display-string
            ?*recordset* 7))
        (text-set-value ?*operating-mode-textbox* (get-display-string
            ?*recordset* 9))
        (text-set-value ?*thermal-behaviour-textbox* (get-display-
            string ?*recordset* 11))
        (text-set-value ?*reaction-type-textbox* (get-display-string
            ?*recordset* 13))
        (text-set-value ?*reaction-order-textbox* (get-display-string
            ?*recordset* 15))
        (text-set-value ?*reversible-textbox* (get-display-string
            ?*recordset* 17))
        (text-set-value ?*phases-textbox* (get-display-string
            ?*recordset* 19))
        (text-set-value ?*mass-transfer-textbox* (get-display-string
            ?*recordset* 21))
        (text-set-value ?*heat-transfer-textbox* (get-display-string
            ?*recordset* 23))
        (text-set-value ?*pipeline-flow-textbox* (get-display-string
            ?*recordset* 25))
        (text-set-value ?*equipment-desc-textbox* (recordset-get-
            char-data ?*recordset* 8))
        (text-set-value ?*operating-mode-desc-textbox* (recordset-
            get-char-data ?*recordset* 10))
        (text-set-value ?*thermal-behaviour-desc-textbox* (recordset-
            get-char-data ?*recordset* 12))
        (text-set-value ?*reaction-type-desc-textbox* (recordset-get-
            char-data ?*recordset* 14))
        (text-set-value ?*reaction-order-desc-textbox* (recordset-
            get-char-data ?*recordset* 16))
        (text-set-value ?*reversible-desc-textbox* (recordset-get-
            char-data ?*recordset* 18))
        (text-set-value ?*phases-desc-textbox* (recordset-get-char-
            data ?*recordset* 20))
        (text-set-value ?*mass-transfer-desc-textbox* (recordset-get-
            char-data ?*recordset* 22))
        (text-set-value ?*heat-transfer-desc-textbox* (recordset-get-
            char-data ?*recordset* 24))
        (text-set-value ?*pipeline-flow-desc-textbox* (recordset-get-
            char-data ?*recordset* 26))
        (text-set-value ?*chemical-textbox* (get-display-string
            ?*recordset* 27))
        (text-set-value ?*record-number-textbox* (float-to-string
            (recordset-get-int-data ?*recordset* 1)))
        (multi-text-set-value ?*description-multi-textbox*
          (str-cat (recordset-get-char-data ?*recordset* 3) " "
            (recordset-get-char-data ?*recordset* 4) " "
            (recordset-get-char-data ?*recordset* 5) " "
            (recordset-get-char-data ?*recordset* 6) ))
        (text-set-value ?*message-box*
          (str-cat ?*current-record-number* " of " ?*retrieved-
            number*))
        (bind ?*prob-name* (recordset-get-char-data ?*recordset* 29))
        (bind ?*unit-name* (recordset-get-char-data ?*recordset* 30))
        (bind ?*model-time* (recordset-get-char-data ?*recordset*
            31))
        (bind ?*rep-int* (recordset-get-char-data ?*recordset* 32))
        (bind ?*temp-low* (recordset-get-char-data ?*recordset* 34))
        (bind ?*temp-high* (recordset-get-char-data ?*recordset* 35))
```

```
            (bind ?*pres-low* (recordset-get-char-data ?*recordset* 36))
            (bind ?*pres-high* (recordset-get-char-data ?*recordset* 37))
            (bind ?*temp-unit* (recordset-get-char-data ?*recordset* 38))
            (bind ?*pres-unit* (recordset-get-char-data ?*recordset* 39))
        else
            (text-set-value ?*number-textbox*  "")
            (text-set-value ?*equipment-textbox* "")
            (text-set-value ?*operating-mode-textbox* "")
            (text-set-value ?*thermal-behaviour-textbox* "")
            (text-set-value ?*reaction-type-textbox* "")
            (text-set-value ?*reaction-order-textbox* "")
            (text-set-value ?*reversible-textbox* "")
            (text-set-value ?*phases-textbox* "")
            (text-set-value ?*mass-transfer-textbox* "")
            (text-set-value ?*heat-transfer-textbox* "")
            (text-set-value ?*pipeline-flow-textbox* "")
            (text-set-value ?*chemical-textbox* "")
            (multi-text-set-value ?*description-multi-textbox* "")
        )
        (window-show ?*panel* 0)
        (window-show ?*panel2* 1)
        (window-show ?*panel3* 0)
    )

    (deffunction large-crlf ()

    )

    (deffunction display-more-eqn () ""
        (open "eqn-file.tmp" ?*temp-file* "w")
        (printout ?*temp-file* "EQUATIONS" crlf crlf)
      (loop-for-count (?cnt 1 20) do
        (if (not (or (eq (recordset-get-char-data ?*recordset* (+ ?cnt
            1)) "")
              (eq (recordset-get-char-data ?*recordset* (+ ?cnt 1))
            "END"))) then
         (printout ?*temp-file* (recordset-get-char-data ?*recordset*
            (+ ?cnt 1)) crlf)
         (if (neq (recordset-get-char-data ?*recordset* (+ 20 (* ?cnt
            4))) "") then
           (printout ?*temp-file* "UB " (recordset-get-char-data
              ?*recordset* (+ 20 (* ?cnt 4))))
         )
         (if (neq (recordset-get-char-data ?*recordset* (+ 19 (* ?cnt
            4))) "") then
           (printout ?*temp-file* " ;LB " (recordset-get-char-data
              ?*recordset* (+ 19 (* ?cnt 4))))
         )
         (if (neq (recordset-get-char-data ?*recordset* (+ 18 (* ?cnt
              4))) "") then
            (printout ?*temp-file* " ;FB " (recordset-get-char-data
              ?*recordset* (+ 18 (* ?cnt 4))))
          )
          (printout ?*temp-file* " ; " (recordset-get-char-data
              ?*recordset* (+ 21 (* ?cnt 4))) crlf crlf)
        )
      )
      (close)
      (text-window-load-file ?*text-win* "eqn-file.tmp")
    )
    (deffunction display-more-const () ""
```

```
(open "const-file.tmp" ?*temp-file* "w")
(printout ?*temp-file* "CONSTANTS" crlf crlf)
(loop-for-count (?cnt 1 35) do
  (if (neq (recordset-get-char-data ?*recordset* (+ ?cnt 1)) "")
  then
   (printout ?*temp-file* (recordset-get-char-data ?*recordset*
        (+ ?cnt 1)) " ") ;Name
   (printout ?*temp-file* (recordset-get-char-data ?*recordset*
        (+ 33 (* ?cnt 4))) " ") ;Value
   (printout ?*temp-file* (recordset-get-char-data ?*recordset*
        (+ 35 (* ?cnt 4))) " ; ") ;Units
   (printout ?*temp-file* (recordset-get-char-data ?*recordset*
        (+ 36 (* ?cnt 4))) crlf crlf) ;Description
  )
     )
     (close)
     (text-window-load-file ?*boris* "const-file.tmp")
)
(deffunction display-more-var () ""
     (open "var-file.tmp" ?*temp-file* "w")
     (printout ?*temp-file* "VARIABLES (Mathematical Method)" crlf
           crlf)
     (loop-for-count (?cnt 1 25) do
       (if (neq (recordset-get-char-data ?*recordset* (+ ?cnt 1)) "")
       then
        (printout ?*temp-file* (recordset-get-char-data ?*recordset*
             (+ ?cnt 1)) " ")
        (printout ?*temp-file* (recordset-get-char-data ?*recordset*
             (+ 19 (* ?cnt 8))) "" )
        (printout ?*temp-file* (recordset-get-char-data ?*recordset*
             (+ 20 (* ?cnt 8))) "")
        (if (not (and (eq (recordset-get-char-data ?*recordset* (+ 19
        (* ?cnt 8))) "")
             (eq (recordset-get-char-data ?*recordset* (+ 20 (*
        ?cnt 8))) "")))) then
          (printout ?*temp-file* " " (recordset-get-char-data
          ?*recordset* (+ 25 (* ?cnt 8))))
     )
     (if (neq (recordset-get-char-data ?*recordset* (+ 23 (* ?cnt
          8))) "") then
       (printout ?*temp-file* " ;UB " (recordset-get-char-data
          ?*recordset* (+ 23 (* ?cnt 8))))
     )
     (if (neq (recordset-get-char-data ?*recordset* (+ 22 (* ?cnt
          8))) "") then
       (printout ?*temp-file* " ;LB " (recordset-get-char-data
          ?*recordset* (+ 22 (* ?cnt 8))))
     )
     (printout ?*temp-file* " ; " (recordset-get-char-data
          ?*recordset* (+ 26 (* ?cnt 8))))
        (if (neq (recordset-get-char-data ?*recordset* (+ 24 (* ?cnt
             8))) "") then
          (printout ?*temp-file* " (; "
                    (recordset-get-char-data ?*recordset* (+ 24 (*
                   ?cnt 8))) ")")
        )
        (printout ?*temp-file* crlf crlf)
      )
   )
   (close)
   (text-window-load-file ?*doris* "var-file.tmp")
```

```
      (window-show ?*panel* 0)
      (window-show ?*panel2* 0)
      (window-show ?*panel3* 1)
)

;*****************************************************************
;Button and checkbox callbacks
;*****************************************************************

; when the children check-box is checked
(deffunction children-check-callback (?id)
   (if (eq ?*match-children* FALSE)
       then (bind ?*match-children* TRUE)
       else (bind ?*match-children* FALSE))
)
; when the parent check-box is checked
(deffunction parent-check-callback (?id)
   (if (eq ?*match-parent* FALSE)
       then (bind ?*match-parent* TRUE)
       else (bind ?*match-parent* FALSE))
)
; when the related check-box is checked
(deffunction related-check-callback (?id)
   (if (eq ?*match-related-terms* FALSE)
       then (bind ?*match-related-terms* TRUE)
       else (bind ?*match-related-terms* FALSE))
)
; when the first button is pressed, set pointer to first record
(deffunction first-button-callback (?id)
  (if (neq ?*retrieved-number* 0)
      then (bind ?*current-record-number* 1)
           (recordset-move-first ?*recordset*)
           (display-record)
  )
)
; when previous button is pressed, move pointer back one place
(deffunction previous-button-callback (?id)
  (if (neq ?*retrieved-number* 0)
      then (if (neq ?*current-record-number* 1)
             then (bind ?*current-record-number* (- ?*current-record-
                  number* 1))
                  (recordset-move-prev ?*recordset*)
                  (display-record))
  )
)
; when next button is pressed, move pointer one place forward
(deffunction next-button-callback (?id)
  (if (neq ?*retrieved-number* 0)
      then (if (neq ?*current-record-number* ?*retrieved-number*)
             then (bind ?*current-record-number* (+ ?*current-record-
                  number* 1))
                  (recordset-move-next ?*recordset*)
                  (display-record))
  )
)
; when last button is pressed, move pointer to last record
(deffunction last-button-callback (?id)
  (if (neq ?*retrieved-number* 0)
      then (bind ?*current-record-number* ?*retrieved-number*)
           (recordset-move-last ?*recordset*)
           (display-record)
```

```
   )
)
;when clear button is pressed, clear panel and reset globals
(deffunction on-panel-clear-button (?id) ""
    (set-reset-globals TRUE)
    (window-delete ?*main-frame*)
    (reset)
    (eval "(startup)")
)

(deffunction printout-sql-query () ""
    (if (neq ?*sql* "") then
        (bind ?*sql* (str-cat "SELECT * FROM " ?*database-data-table-
            name* " WHERE (" ?*sql* ")" ))
        (if  (eq 0 (recordset-execute-sql ?*recordset* ?*sql*)) then
            (bind ?msg (database-get-error-message ?*database*))
            (printout t ?msg crlf)
            (return 0)
        )
 (printout t "Records: " (recordset-get-number-records
            ?*recordset*) crlf)
 )
 (bind ?*retrieved-number* (recordset-get-number-records
            ?*recordset*))
 (if (neq ?*retrieved-number* 0) then
     (bind ?*current-record-number* 1)
     (display-record)
 else
     (text-set-value ?*message-box* (str-cat "No records
         retrieved." ))
     (display-record)
 )
 )
 (deffunction get-eqn-sql-query () ""
  (initialise-database)
   (bind ?*sql* (str-cat ?*sqlall* ?*main-ID* ))
     (if (neq ?*sql* "") then
         (bind ?*sql* (str-cat "SELECT * FROM " ?*database-equation-
             table-name* " WHERE (" ?*sql* ")" ))
         (if  (eq 0 (recordset-execute-sql ?*recordset* ?*sql*)) then
             (bind ?msg (database-get-error-message ?*database*))
             (printout t ?msg crlf)
             (return 0)
         )
     )
     (bind ?*retrieved-number* (recordset-get-number-records
             ?*recordset*))
 )
 (deffunction printout-eqn-sql-query () ""
   (get-eqn-sql-query)
     (if (neq ?*retrieved-number* 0) then
         (bind ?*current-record-number* 1)
         (display-more-eqn)
     )
 )
 (deffunction get-const-sql-query ()
  (initialise-database)
   (bind ?*sql* (str-cat ?*sqlall* ?*main-ID* ))
     (if (neq ?*sql* "") then
         (bind ?*sql* (str-cat "SELECT * FROM " ?*database-constant-
             table-name* " WHERE (" ?*sql* ")" ))
```

```
            (if  (eq 0 (recordset-execute-sql ?*recordset* ?*sql*)) then
                (bind ?msg (database-get-error-message ?*database*))
                (printout t ?msg crlf)
                (return 0)
            )
        )
        (bind ?*retrieved-number* (recordset-get-number-records
          ?*recordset*))
    )
    (deffunction printout-const-sql-query () ""
      (get-const-sql-query)
        (if (neq ?*retrieved-number* 0) then
            (bind ?*current-record-number* 1)
            (display-more-const)
        )
    )
    (deffunction get-var-sql-query ()
     (initialise-database)
      (bind ?*sql* (str-cat ?*sqlall* ?*main-ID* ))
        (if (neq ?*sql* "") then
            (bind ?*sql* (str-cat "SELECT * FROM " ?*database-variable-
                table-name* " WHERE (" ?*sql* ")" ))
            (if  (eq 0 (recordset-execute-sql ?*recordset* ?*sql*)) then
                (bind ?msg (database-get-error-message ?*database*))
                (printout t ?msg crlf)
                (return 0)
            )
        )
        (bind ?*retrieved-number* (recordset-get-number-records
                ?*recordset*))
    )
    (deffunction printout-var-sql-query () ""
      (get-var-sql-query)
        (if (neq ?*retrieved-number* 0) then
            (bind ?*current-record-number* 1)
            (display-more-var)
        )
    )

    ;*****************************************************************

    (deffunction not-exist (?term ?multi-term) ""
         (while (> (length ?multi-term) 0)
            (if (eq (str-compare ?term
                        (sub-string 2
                        (- (length (implode$ (first$ ?multi-term))) 1)
                        (implode$ (first$ ?multi-term)))) 0) then
                (return FALSE)
            )
            (bind ?multi-term (rest$ ?multi-term))
        )
        (return TRUE)
    )

    ;*****************************************************************

    ; when search button is pressed, set up and call the SQL query
    (deffunction search-button-callback (?id)
        (bind ?*temp-low* (text-get-value ?*temp-low-searchbox*))
        (bind ?*temp-high* (text-get-value ?*temp-high-searchbox*))
        (bind ?*pres-low* (text-get-value ?*pres-low-searchbox*))
```

```
      (bind ?*pres-high* (text-get-value ?*pres-high-searchbox*))
  (printout t ?*Num-Search* crlf)
      (if (neq ?*Num-Search* 1) then
        (Numerical-Search)
      else
        (initialise-database)
        (construct-sql-query)
        (printout-sql-query)
      )
)

;*****************************************************************
; When More Info button pressed, show equation screen
(deffunction more-button-callback (?id)
    (window-show ?*panel2* 0)
    (menu-enable ?*data-menu* 11 0)
    (menu-enable ?*report-menu* 21 0)
    (menu-enable ?*report-menu* 22 0)
    (menu-enable ?*report-menu* 23 0)
    (menu-enable ?*report-menu* 24 0)
    (printout-eqn-sql-query)
    (printout-const-sql-query)
    (printout-var-sql-query)
)

;*****************************************************************
;startup creates a frame, panels and panel items
;*****************************************************************

(deffunction on-menu-command (?frame ?id)
 (switch ?id
; Quit
   (case 1 then
     (database-close ?*database*)
     (if (on-close ?frame) then (window-delete ?frame)))

;More Info
   (case 11 then (more-button-callback ?id)
   )
;Occurence Matrix
   (case 12 then (occurence-matrix))
; Search
   (case 13 then (search-button-callback ?id))
; Clear
   (case 14 then (set-reset-globals TRUE)
     (database-close ?*database*)
     (window-delete ?*main-frame*)
     (reset)
     (eval "(startup)"))

; First
   (case 21 then (first-button-callback ?id))
; Previous
   (case 22 then (previous-button-callback ?id))
; Next
   (case 23 then (next-button-callback ?id))
; Last
   (case 24 then (last-button-callback ?id))

; gPROMS translator
   (case 30 then (translate-gPROMS))
```

```
; Simulink translator
  (case 31 then (translate-SIMULINK))

; Help
  (case 200 then (message-box "Intelligent Modelling Interface for
      Process Simulators v0.04
    by Graham Clark (c) 2000" wxOK 1 0 "About IMIPS Prototype"))
 )
)

(deffunction startup () ""
    (bind ?*main-frame* (frame-create 0 "Intelligent Modelling
      Interface for Process Simulators"
       -1 -1 ?*main-frame-x* ?*main-frame-y* wxDEFAULT_FRAME))
    (frame-create-status-line ?*main-frame*)
    (frame-set-status-text ?*main-frame* "")


;add callbacks
    (window-add-callback ?*main-frame* OnSize on-size)
    (window-add-callback ?*main-frame* OnClose on-close)
    (window-add-callback ?*main-frame* OnMenuCommand on-menu-command)

;Populate panel1
    (bind ?*panel* (panel-create ?*main-frame* 0 0 ?*main-frame-x*
      ?*main-frame-y*))
    (panel-set-label-position ?*panel* wxVERTICAL)

    (bind ?file-menu (menu-create))
    (menu-append ?file-menu 1 "&Quit")
    (bind ?*data-menu* (menu-create))
    (menu-append ?*data-menu* 11 "&More Info...")
    (menu-append ?*data-menu* 12 "&Occurence Matrix")
    (menu-enable ?*data-menu* 11 0)
    (menu-enable ?*data-menu* 12 0)
    (menu-append-separator ?*data-menu*)
    (menu-append ?*data-menu* 13 "&Search")
    (menu-append-separator ?*data-menu*)
    (menu-append ?*data-menu* 14 "&Clear")
    (bind ?*report-menu* (menu-create))
    (menu-append ?*report-menu* 21 "&First")
    (menu-append ?*report-menu* 22 "&Previous")
    (menu-append ?*report-menu* 23 "&Next")
    (menu-append ?*report-menu* 24 "&Last")
    (menu-enable ?*report-menu* 21 0)
    (menu-enable ?*report-menu* 22 0)
    (menu-enable ?*report-menu* 23 0)
    (menu-enable ?*report-menu* 24 0)
    (bind ?*translate-menu* (menu-create))
    (menu-append ?*translate-menu* 30 "&gPROMS")
    (menu-append ?*translate-menu* 31 "&SIMULINK")
    (menu-enable ?*translate-menu* 30 0)
    (menu-enable ?*translate-menu* 31 0)
    (bind ?help-menu (menu-create))
    (menu-append ?help-menu 200 "&About")
    (bind ?menu-bar (menu-bar-create))
    (menu-bar-append ?menu-bar ?file-menu "&File")
    (menu-bar-append ?menu-bar ?*data-menu* "F&orms")
    (menu-bar-append ?menu-bar ?*report-menu* "&Report")
    (menu-bar-append ?menu-bar ?*translate-menu* "&Translate")
    (menu-bar-append ?menu-bar ?help-menu "&Help")
```

```
(frame-set-menu-bar ?*main-frame* ?menu-bar)

(bind ?listequip (list-box-create ?*panel* on-list-choice "Select
  Equipment" 0 -1 -1 ?*box-s-x* ?*list-s-y* "" "equipment-list"))
(append-to-list ?listequip (class-subclasses equipment))
(bind ?*equipment-searchbox* (text-create ?*panel* "" "Search" ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-equipment-searchbox* (text-create ?*panel* ""
  "Related Items" "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listopmode (list-box-create ?*panel* on-list-choice
  "Operating Mode"  0 -1 -1 ?*box-s-x* ?*list-s-y* "" "operating-
  mode-list"))
(append-to-list ?listopmode (class-subclasses Operating_Mode))
(bind ?*operating-mode-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-operating-mode-searchbox* (text-create ?*panel*
  "" " " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listthermal (list-box-create ?*panel* on-list-choice
  "Thermal Behaviour"  0 -1 -1 ?*box-s-x* ?*list-s-y* ""
  "thermal-behaviour-list"))
(append-to-list ?listthermal (class-subclasses
  Thermal_Behaviour))
(bind ?*thermal-behaviour-searchbox* (text-create ?*panel* "" " "
  "" -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-thermal-behaviour-searchbox* (text-create
  ?*panel* "" " " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listreactype (list-box-create ?*panel* on-list-choice
  "Reaction Type" 0 -1 -1 ?*box-s-x* ?*list-s-y* "" "reaction-
  type-list"))
(append-to-list ?listreactype (class-subclasses Reaction_Type))
(bind ?*reaction-type-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-reaction-type-searchbox* (text-create ?*panel* ""
  " " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listreacorder (list-box-create ?*panel* on-list-choice
  "Reaction Order"  0 -1 -1 ?*box-s-x* ?*list-s-y* "" "reaction-
  order-list"))
(append-to-list ?listreacorder (class-subclasses Reaction_Order))
(bind ?*reaction-order-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-reaction-order-searchbox* (text-create ?*panel*
  "" " " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listreversible (list-box-create ?*panel* on-list-choice
  "Reversible?"  0 -1 -1 ?*box-s-x* ?*list-s-y* "" "reversible-
  list"))
(append-to-list ?listreversible (class-subclasses Reversibility))
(bind ?*reversible-searchbox* (text-create ?*panel* "" " " "" -1
  -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-reversible-searchbox* (text-create ?*panel* "" "
  " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listphase (list-box-create ?*panel* on-list-choice "Select
  Phases" 0 -1 -1 ?*box-s-x* ?*list-s-y* "" "phase-list"))
(append-to-list ?listphase (class-subclasses phases))
(bind ?*phases-searchbox* (text-create ?*panel* "" " " "" -1 -1
  ?*text-s-x* ?*text-s-y*))
```

```
(bind ?*related-phases-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listmasstrans (list-box-create ?*panel* on-list-choice
  "Mass Transfer" 0 -1 -1 ?*box-s-x* ?*list-s-y* "" "mass-
  transfer-list"))
(append-to-list ?listmasstrans (class-subclasses Mass_Transfer))
(bind ?*mass-transfer-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-mass-transfer-searchbox* (text-create ?*panel* ""
  " " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listheattrans (list-box-create ?*panel* on-list-choice
  "Heat Transfer"  0 -1 -1 ?*box-s-x* ?*list-s-y* "" "heat-
  transfer-list"))
(append-to-list ?listheattrans (class-subclasses Heat_Transfer))
(bind ?*heat-transfer-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-heat-transfer-searchbox* (text-create ?*panel* ""
  " " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?listpipeline (list-box-create ?*panel* on-list-choice
  "Pipeline Flow Type"  0 -1 -1 ?*box-s-x* ?*list-s-y* ""
  "pipeline-flow-list"))
(append-to-list ?listpipeline (class-subclasses Pipeline_Flow))
(bind ?*pipeline-flow-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*related-pipeline-flow-searchbox* (text-create ?*panel* ""
  " " "" -1 -1 ?*text-s-x* ?*text-s-y*))
(panel-new-line ?*panel*)
(bind ?list-chemicals (list-box-create ?*panel* on-list-choice
  "Select Chemical" 0 -1 -1 ?*box-s-x* ?*list-s-y* "" "chemical-
  list"))
(append-to-list ?list-chemicals (class-subclasses chemical))
(bind ?*chemical-searchbox* (text-create ?*panel* "" " " "" -1 -1
  ?*text-s-x* ?*text-s-y*))
(bind ?*related-chemical-searchbox* (text-create ?*panel* "" " "
  "" -1 -1 ?*text-s-x* ?*text-s-y*))

(panel-new-line ?*panel*)
(bind ?*temp-low-searchbox* (text-create ?*panel* "" "Temperature
  (Low)" "" -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*temp-high-searchbox* (text-create ?*panel* "" "(High)" ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?list-temp (list-box-create ?*panel* on-list-choice "Select
  Units" 0 -1 -1 (- ?*box-s-x* 200) ?*list-s-y* "" "temp-list"))
(append-to-list ?list-temp (class-subclasses temp-unit))
(bind ?*temp-tolerance-searchbox* (text-create ?*panel* ""
  "Tolerance" "" -1 -1 (- ?*text-s-x* 123) ?*text-s-y*))

(panel-new-line ?*panel*)
(bind ?*pres-low-searchbox* (text-create ?*panel* "" "Pressure
  (Low)" "" -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?*pres-high-searchbox* (text-create ?*panel* "" "(High)" ""
  -1 -1 ?*text-s-x* ?*text-s-y*))
(bind ?list-pres (list-box-create ?*panel* on-list-choice " " 0 -
  1 -1 (- ?*box-s-x* 200) ?*list-s-y* "" "pres-list"))
(append-to-list ?list-pres (class-subclasses pres-unit))
(bind ?*pres-tolerance-searchbox* (text-create ?*panel* "" " " ""
  -1 -1 (- ?*text-s-x* 123) ?*text-s-y*))
```

```
(bind ?search-button (button-create ?*panel* search-button-
  callback "Search" ?*button-p-x* ?*button-p-y* ?*but-s-x* ?*but-
  s-y* "" "search"))
(bind ?swap-panel (button-create ?*panel* on-panel-change-button
  "Results" -1 -1 ?*but-s-x* ?*but-s-y* "" "next"))
(bind ?clear-panel (button-create ?*panel* on-panel-clear-button
  "Clear" -1 -1 ?*but-s-x* ?*but-s-y* "" "clear"))

(bind ?*description-searchbox* (text-create ?*panel* "" "Enter
  Description Keyword" "" ?*supp-search-p-x* ?*freetext-desc-y*
  ?*box-s-x* ?*key-y*))
(bind ?*record-number-searchbox* (text-create ?*panel* "" "Record
  No." "" ?*rec-num-p-x* ?*rec-num-p-y* 75 ?*key-y*))
```

```
(bind ?parent-check (check-box-create ?*panel* parent-check-
  callback "Parent" ?*check-p-x* ?*check-p-y* -1 -1 "" ""))
(bind ?children-check (check-box-create ?*panel* children-check-
  callback "Children" -1 -1 -1 -1 "" ""))
    (check-box-set-value ?children-check 1)
    (bind ?*match-children* TRUE)
(bind ?related-check (check-box-create ?*panel* related-check-
  callback "Related" -1 -1 -1 -1 "" ""))

;populate panel2
(bind ?*panel2* (panel-create ?*main-frame* 0 0 ?*frame-x*
  ?*frame-y*))
(panel-set-label-position ?*panel2* wxVERTICAL)
(bind ?*description-multi-textbox* (multi-text-create ?*panel2*
  "" "Problem description"
  "" -1 -1 ?*descrip-s-x* ?*descrip-s-y* "wxREADONLY"))
(panel-new-line ?*panel2*)
(bind ?*equipment-textbox* (text-create ?*panel2* "" "Equipment"
  "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*equipment-desc-textbox* (text-create ?*panel2* ""
  "Description" "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*operating-mode-textbox* (text-create ?*panel2* ""
  "Operating Mode" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*operating-mode-desc-textbox* (text-create ?*panel2* "" "
  " "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*thermal-behaviour-textbox* (text-create ?*panel2* ""
  "Thermal Behaviour" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*thermal-behaviour-desc-textbox* (text-create ?*panel2* ""
  " " "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*reaction-type-textbox* (text-create ?*panel2* ""
  "Reaction Type" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*reaction-type-desc-textbox* (text-create ?*panel2* "" " "
  "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*reaction-order-textbox* (text-create ?*panel2* ""
  "Reaction Order" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*reaction-order-desc-textbox* (text-create ?*panel2* "" "
  " "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*reversible-textbox* (text-create ?*panel2* ""
  "Reversibility" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*reversible-desc-textbox* (text-create ?*panel2* "" " " ""
  -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
```

```
(bind ?*phases-textbox* (text-create ?*panel2* "" "Phases" "" -1
  -1 ?*box-s-x* ?*key-y*))
(bind ?*phases-desc-textbox* (text-create ?*panel2* "" " " "" -1
  -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*mass-transfer-textbox* (text-create ?*panel2* "" "Mass
  Transfer" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*mass-transfer-desc-textbox* (text-create ?*panel2* "" " "
  "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*heat-transfer-textbox* (text-create ?*panel2* "" "Heat
  Transfer" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*heat-transfer-desc-textbox* (text-create ?*panel2* "" " "
  "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*pipeline-flow-textbox* (text-create ?*panel2* ""
  "Pipeline Flow Type" "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*pipeline-flow-desc-textbox* (text-create ?*panel2* "" " "
  "" -1 -1 ?*box-s-xl* ?*key-y*))
(panel-new-line ?*panel2*)
(bind ?*chemical-textbox* (text-create ?*panel2* "" "Chemicals"
  "" -1 -1 ?*box-s-x* ?*key-y*))
(bind ?*message-box* (text-create ?*panel2* "" "Retrieved Set"
   "" -1 -1 ?*rec-set-s-x* ?*key-y* "wxREADONLY"))
(bind ?*record-number-textbox* (text-create ?*panel2* "" "Record
  no." "" -1 -1 200 ?*key-y*))

(bind ?first-button (button-create ?*panel2* first-button-
  callback "|<" ?*first-but-p-x* ?*first-but-p-y* 30 30))
(bind ?previous-button (button-create ?*panel2* previous-button-
  callback "<" -1 -1 30 30))
(bind ?next-button (button-create ?*panel2* next-button-callback
  ">" -1 -1 30 30))
(bind ?last-button (button-create ?*panel2* last-button-callback
  ">|" -1 -1 30 30))

(bind ?swap-panel (button-create ?*panel2* on-panel-change-button
  "Search" ?*pan2-button-p-x* ?*pan2-button-p-y1* ?*but-s-x*
  ?*but-s-y* "" "prev"))
(bind ?clear-panel2 (button-create ?*panel2* on-panel-clear-
  button "Clear" ?*pan2-button-p-x* ?*pan2-button-p-y2* ?*but-s-
  x* ?*but-s-y* "" "clear"))
(bind ?more-panel (button-create ?*panel2* on-panel-more-button
  "More Info.." ?*pan2-button-p-x* ?*pan2-button-p-y3* ?*but-s-x*
  ?*but-s-y* "" "more"))

;populate panel3
    (bind ?*panel3* (panel-create ?*main-frame* 0 0 ?*frame-x*
      ?*frame-y*))
    (panel-set-label-position ?*panel3* wxVERTICAL)
    (bind ?*text-win* (text-window-create ?*panel3*))
    (bind ?*boris* (text-window-create ?*panel3* ))
    (bind ?*doris* (text-window-create ?*panel3* ))
    (text-window-set-editable ?*text-win* 0)
    (text-window-set-editable ?*boris* 0)
    (text-window-set-editable ?*doris* 0)

    (window-centre ?*main-frame* wxBOTH)
    (window-show ?*main-frame* 1)
    (window-show ?*panel* 1)
    (window-show ?*panel2* 0)
```

```
    (window-show ?*panel3* 0)
    ?*main-frame*)

;*****************************************************************

(deffunction on-start-choice (?id) ""
    (bind ?start-selection (list-box-get-string-selection ?id))
    (switch ?start-selection
        (case "Large Display (17+ inch Screen)" then
            (load "lgscreen.clp")
        )
        (case "Medium Display (15 inch Screen)" then
            (load "mdscreen.clp")
        )
        (case "Small Display (14 inch Screen)" then
            (load "smscreen.clp")
        )

    )
    (window-show ?*start-frame* 0)
    (window-show ?*start-panel* 0)
    (startup)
)

;*****************************************************************
; Function to create problem eqn, var and const instances
;*****************************************************************

(deffunction create-problem-instances ()
;Create equations
  (text-window-write ?*trans* "Creating problem equation instances ")
  (get-eqn-sql-query)
(bind ?*no-of-eqn* 0)
(make-instance [eqn] of EQUATION
 (Eqn "") (Description "Equation used for initialising loops"))
(loop-for-count (?cnt 1 20) do
 (if (not (or (eq (recordset-get-char-data ?*recordset* (+ ?cnt 1))
    "") (eq (recordset-get-char-data ?*recordset* (+ ?cnt 1))
    "END"))) then
  (bind ?*no-of-eqn* (+ ?*no-of-eqn* 1))
  (make-instance (symbol-to-instance-name (sym-cat eqn ?cnt)) of
    EQUATION
    (Eqn (recordset-get-char-data ?*recordset* (+ ?cnt 1)))
      (UpperBound
       (if (neq (recordset-get-char-data ?*recordset* (+ 20 (* ?cnt
          4))) "") then
        (recordset-get-char-data ?*recordset* (+ 20 (* ?cnt 4)))
        else "All"))
      (LowerBound (if (neq (recordset-get-char-data ?*recordset* (+
          19 (* ?cnt 4))) "") then
        (recordset-get-char-data ?*recordset* (+ 19 (* ?cnt 4)))
        else "All"))
      (FixedBound (if (neq (recordset-get-char-data ?*recordset* (+
          18 (* ?cnt 4))) "") then
        (recordset-get-char-data ?*recordset* (+ 18 (* ?cnt 4)))
        else "All"))
      (Description (recordset-get-char-data ?*recordset* (+ 21 (*
          ?cnt 4))))
  )
 )
)
```

```
  (text-window-write ?*trans* "Completed ")

;Create variables
  (text-window-write ?*trans* "Creating problem variable instances ")
  (get-var-sql-query)
  (bind ?*no-of-var* 0)
  (loop-for-count (?cnt 1 25) do
   (if (neq (recordset-get-char-data ?*recordset* (+ ?cnt 1)) "")
   then
    (bind ?*no-of-var* (+ ?*no-of-var* 1))
    (make-instance (symbol-to-instance-name (sym-cat var ?cnt)) of
      VARIABLE
     (Identifier (recordset-get-char-data ?*recordset* (+ ?cnt 1)))
     (Unit (if (neq (recordset-get-char-data ?*recordset* (+ 25 (*
     ?cnt 8))) "") then
       (recordset-get-char-data ?*recordset* (+ 25 (* ?cnt 8)))
       else ""))
    (Description (recordset-get-char-data ?*recordset* (+ 26 (* ?cnt
     8))))
    (UpperBound  (if (neq (recordset-get-char-data ?*recordset* (+
     23 (* ?cnt 8))) "") then
       (recordset-get-char-data ?*recordset* (+ 23 (* ?cnt 8)))
       else "All"))
    (LowerBound  (if (neq (recordset-get-char-data ?*recordset* (+
     22 (* ?cnt 8))) "") then
       (recordset-get-char-data ?*recordset* (+ 22 (* ?cnt 8)))
       else "All"))
    (Set  (if (neq (recordset-get-char-data ?*recordset* (+ 20 (*
     ?cnt 8))) "") then
       (recordset-get-char-data ?*recordset* (+ 20 (* ?cnt 8)))
        else "unknown"))
    (Value  (if (neq (recordset-get-char-data ?*recordset* (+ 19 (*
     ?cnt 8))) "") then
        (recordset-get-char-data ?*recordset* (+ 19 (* ?cnt 8)))
         else "unknown"))
       (Distributed  (if (neq (recordset-get-char-data ?*recordset* (+
        21 (* ?cnt 8))) "") then
          (recordset-get-char-data ?*recordset* (+ 21 (* ?cnt 8)))
           else "N/A"))
       (MathMethod  (if (neq (recordset-get-char-data ?*recordset* (+
        24 (* ?cnt 8))) "") then
          (recordset-get-char-data ?*recordset* (+ 24 (* ?cnt 8)))
           else "N/A"))
     )
    )
   )
  (text-window-write ?*trans* "Completed ")

;Create constants
  (text-window-write ?*trans* "Creating problem constant instances ")
  (get-const-sql-query)
  (bind ?*no-of-const* 0)
  (loop-for-count (?cnt 1 35) do
   (if (neq (recordset-get-char-data ?*recordset* (+ ?cnt 1)) "")
   then
    (bind ?*no-of-const* (+ ?*no-of-const* 1))
    (make-instance (symbol-to-instance-name (sym-cat const ?cnt)) of
      CONSTANT
     (Identifier (recordset-get-char-data ?*recordset* (+ ?cnt 1)))
     (Unit (if (neq (recordset-get-char-data ?*recordset* (+ 35 (*
      ?cnt 4))) "") then
```

```
    (recordset-get-char-data ?*recordset* (+ 35 (* ?cnt 4)))
    else ""))
(Include (if (neq (recordset-get-char-data ?*recordset* (+ 34 (*
 ?cnt 4))) "") then
    (recordset-get-char-data ?*recordset* (+ 34 (* ?cnt 4)))
    else "Y"))
(Description (recordset-get-char-data ?*recordset* (+ 36 (* ?cnt
 4))))
(Value (if (neq (recordset-get-char-data ?*recordset* (+ 33 (*
 ?cnt 4))) "") then
  (if (eq (recordset-get-char-data ?*recordset* (+ 34 (* ?cnt
       4))) "Y") then
      (recordset-get-char-data ?*recordset* (+ 33 (* ?cnt 4)))
   else
        (string-to-long (recordset-get-char-data ?*recordset* (+
             33 (* ?cnt 4))))
      )
     else 0.1))
    )
   )
  )
  (text-window-write ?*trans* "Completed ")
)

;*****************************************************************
; Functions for gPROMS translation
;*****************************************************************

(deffunction translate-gPROMS ()
; Create a frame to keep track of the translation
  (bind ?*trans-frame* (frame-create 0 "Translation Status" 220 200
      420 250))
  (frame-create-status-line ?*trans-frame*)
  (bind ?*trans-panel* (panel-create ?*trans-frame* 0 0 400 400))
  (panel-set-label-position ?*trans-panel* wxVERTICAL)
  (bind ?*trans* (text-window-create ?*trans-panel*))
  (window-set-size ?*trans* 0 0 415 250)
  (text-window-set-editable ?*trans* 0)
  (window-show ?*trans-frame* 1)
  (create-problem-instances)
  (text-window-write ?*trans* "gPROMS translator invoked")
  (bind ?name (get-text-from-user "File Output Name" (sym-cat ?*prob-
     name* ".gPROMS")))
  (open ?name ?*sim-file* "w")
  (text-window-write ?*trans* (sym-cat "Writing gPROMS input file: "
     ?name " "))
  (print-gPROMS-introduction)
  (print-gPROMS-declaration)
  (print-gPROMS-model)
  (print-gPROMS-process)
  (close)
  (text-window-write ?*trans* "gPROMS translation complete")
  (if (on-close ?*trans-frame*) then (window-delete ?*trans-frame*))
)

;*****************************************************************
; Functions for SIMULINK translation
;*****************************************************************

(deffunction Scope-check-callback (?id)
  (if (eq ?*Scope* FALSE) then
```

```
        (bind ?*Scope* TRUE)
       else
        (bind ?*Scope* FALSE)
   )
   (if (eq ?*Scope* TRUE) then
     (bind ?*Display* FALSE)
     (check-box-set-value ?*Display* 0)
   )
)

(deffunction Display-check-callback (?id)
  (if (eq ?*Display* FALSE) then
      (bind ?*Display* TRUE)
     else
      (bind ?*Display* FALSE)
   )
   (if (eq ?*Display* TRUE) then
     (bind ?*Scope* FALSE)
     (check-box-set-value ?*Scope* 0)
   )
)

(deffunction Simulink-continue ()
  (bind ?name (get-text-from-user "File Output Name" (sym-cat ?*prob-
      name* ".mdl")))
  (open ?name ?*sim-file* "w")
  (text-window-write ?*trans* (sym-cat "Writing SIMULINK input file:
      " ?name " "))
  (print-simulink-introduction)
  (print-simulink-blocks)
  (close)
  (if (on-close ?*trans-frame*) then
    (window-delete ?*trans-frame*)
    (window-delete ?*vars-frame*))
   (text-window-write ?*trans* "SIMULINK translation complete")
)

(deffunction OK-button-callback (?id)
 (window-delete ?*vars-frame*)
)

(deffunction translate-SIMULINK ()
; Create a frame to keep track of the translation
  (bind ?*trans-frame* (frame-create 0 "Translation Status" 220 200
      420 250))
  (frame-create-status-line ?*trans-frame*)
  (bind ?*trans-panel* (panel-create ?*trans-frame* 0 0 400 400))
  (panel-set-label-position ?*trans-panel* wxVERTICAL)
  (bind ?*trans* (text-window-create ?*trans-panel*))
  (window-set-size ?*trans* 0 0 415 250)
  (text-window-set-editable ?*trans* 0)
  (window-show ?*trans-frame* 1)
  (create-problem-instances)
  (text-window-write ?*trans* "SIMULINK translator invoked")
 (Simulink-continue)
)

;*****************************************************************
; Function App-on-init
;*****************************************************************
```

```
(deffunction app-on-init () ""
    (unwatch all)
    (reset)
    (if (= ?*small-font* 0) then
    (bind ?*small-font* (font-create 12 wxSWISS wxNORMAL wxNORMAL 0))
    )

;;;create initial popup menu to determine frame/panel size
    (bind ?*start-frame* (frame-create 0 "Database of Simulation
      Models" 220 200 420 250))
    (frame-create-status-line ?*start-frame*)
    (bind ?*start-panel* (panel-create ?*start-frame* 0 0 400 400))
    (panel-set-label-position ?*start-panel* wxVERTICAL)
    (bind ?start-choice (list-box-create ?*start-panel* on-start-
      choice "Select Preferred Size of Display" 0 30 40 350 100 ""
      ""))
    (append-to-list ?start-choice ?*initial-choice*)
    (window-centre ?*start-frame* wxBOTH)
    (window-show ?*start-frame* 1)
    (window-show ?*start-panel* 1)
)


(deffunction popup2 (?searchbox ?var ?related-searchbox ?related-var)
      ""
    (text-set-value ?searchbox ?var)
    (if (eq ?*match-related-terms* TRUE) then
        (text-set-value ?related-searchbox ?related-var)
        (bind ?related-var ?*related-term*)
    )
    (return)
)


(deffunction popup (?choice1 ?name ?choice) ""
    (if (and (neq (length (class-subclasses (string-to-symbol
      ?choice1))) 0) (neq ?choice1 ?choice)) then
        (bind ?children (create$ (class-subclasses (string-to-symbol
            ?choice1))))
        (bind ?children (insert$ ?children 1 ?choice1))
        (bind ?choice2 (get-choice ?choice1 ?children ?*panel*))
    else
        (bind ?*related-term* "")
        (assert (find-related ?choice1))
        (run)
        (switch ?name
            (case "equipment-list" then
                (bind ?*equipment* ?choice1)
        (popup2 ?*equipment-searchbox* ?*equipment*
          ?*related-equipment-searchbox* ?*related-
          equipment*)
)
(case "operating-mode-list" then
    (bind ?*operating-mode* ?choice1)
    (popup2 ?*operating-mode-searchbox* ?*operating-mode*
      ?*related-operating-mode-searchbox* ?*related-
      operating-mode*)
)
(case "thermal-behaviour-list" then
    (bind ?*thermal-behaviour* ?choice1)
```

```
        (popup2 ?*thermal-behaviour-searchbox* ?*thermal-
          behaviour* ?*related-thermal-behaviour-searchbox*
          ?*related-thermal-behaviour*)
    )
    (case "reaction-type-list" then
        (bind ?*reaction-type* ?choice1)
        (popup2 ?*reaction-type-searchbox* ?*reaction-type*
          ?*related-reaction-type-searchbox*
          ?*related-reaction-type*)
    )
    (case "reaction-order-list" then
        (bind ?*reaction-order* ?choice1)
        (popup2 ?*reaction-order-searchbox* ?*reaction-order*
          ?*related-reaction-order-searchbox* ?*related-
          reaction-order*)
    )
    (case "reversible-list" then
        (bind ?*reversible* ?choice1)
        (popup2 ?*reversible-searchbox* ?*reversible*
          ?*related-reversible-searchbox* ?*related-
          reversible*)
    )
    (case "phase-list" then
        (bind ?*phases* ?choice1)
        (popup2 ?*phases-searchbox* ?*phases* ?*related-
          phases-searchbox* ?*related-phases*)
    )
    (case "mass-transfer-list" then
        (bind ?*mass-transfer* ?choice1)
        (popup2 ?*mass-transfer-searchbox* ?*mass-transfer*
          ?*related-mass-transfer-searchbox* ?*related-mass-
          transfer*)
    )
    (case "heat-transfer-list" then
        (bind ?*heat-transfer* ?choice1)
        (popup2 ?*heat-transfer-searchbox* ?*heat-transfer*
          ?*related-heat-transfer-searchbox* ?*related-heat-
          transfer*)
    )
    (case "pipeline-flow-list" then
        (bind ?*pipeline-flow* ?choice1)
        (popup2 ?*pipeline-flow-searchbox* ?*pipeline-flow*
          ?*related-pipeline-flow-searchbox* ?*related-
          pipeline-flow*)
    )
    (case "chemical-list" then
        (bind ?*chemical* ?choice1)
        (popup2 ?*chemical-searchbox* ?*chemical* ?*related-
          chemical-searchbox* ?*related-chemical*)
    )
        )
    )
    (popup ?choice2 ?name ?choice1)
)
```

IV.6. gPROMS translator code file: gPROMStrans.clp

```
;*****************************************************************
;  Author :          Graham Clark                                 *
;  File Name:        defgPROM.clp                                 *
;  Last Updated:     20/12/00                                     *
;  Description:      gPROMS input file creation translator version *
;*****************************************************************

;Forward declarations
(deffunction check-eqn (?eqn ?name))
(deffunction check-bdry-eqn (?eqn ?name))
(deffunction str-replace (?a ?b ?c))
(deffunction str-replace-all (?a ?b ?c))

(deffunction str-remove (?string)
 (loop-for-count (?cnt 1 ?*no-of-var*) do
  (if (<> 0 (str-compare "Y"
   (send (symbol-to-instance-name (sym-cat var ?cnt))get-Include)))
  then
   (bind ?string (str-replace-all ?string
    (send (symbol-to-instance-name(sym-cat var ?cnt))get-Identifier)
      ""))
  )
 )
 (loop-for-count (?cnt 1 ?*no-of-const*) do
  (if (<> 0 (str-compare "Y"
   (send (symbol-to-instance-name (sym-cat const ?cnt))get-Include)))
  then
   (bind ?string (str-replace-all ?string
    (send (symbol-to-instance-name(sym-cat const ?cnt))get-
        Identifier) ""))
   )
  )
  (return ?string)
)

(deffunction partial (?string)
 (bind ?parcount 0)
 (bind ?st 1)
 (bind ?end (str-length ?string))
 (while (= 0 (str-compare "TRUE" (integerp(str-index "PARTIAL"
  (sub-string ?st ?end ?string))))) do
  (bind ?parcount (+ ?parcount 1))
  (bind ?length (str-length ?string))
  (bind ?stpar (str-index ":" ?string))
  (bind ?endpar (+ ?stpar (str-index ")" (sub-string ?stpar ?length
      ?string)) -1))
  (bind ?replace (sub-string ?stpar ?endpar ?string))
  (bind ?string (str-replace ?string ?replace (sym-cat (sym-cat #
      ?parcount) #)))
  (bind ?st ?endpar)
  (make-instance (symbol-to-instance-name (sym-cat par ?parcount)) of
      PARTIALS (Value ?replace) (ReplacedWith (sym-cat # (sym-cat
      ?parcount #))))
 )
 (make-instance [par0] of PARTIALS (Number ?parcount))
 (return ?string)
)

(deffunction change-integral (?eqn ?name)
```

```
(bind ?*integral-count* (+ ?*integral-count* 1))
(bind ?start (+ (str-index "INTEGRAL(" ?eqn) 9))
(bind ?end (str-index ":" ?eqn))
(bind ?length (str-length ?eqn))
(bind ?variable (sub-string ?start (- ?end 1) ?eqn))
(bind ?finish (+ (str-index ")" (sub-string ?start ?length ?eqn))(-
     ?start 1)))
(bind ?string (sub-string ?start ?finish ?eqn))
(bind ?eqn (str-cat (sym-cat (sub-string 1 (- ?start 1) ?eqn)
 (str-replace-all ?string ?variable (sym-cat "IntNo" ?*integral-
     count*)))
 (sub-string (+ ?finish 1) ?length ?eqn)))
(send (symbol-to-instance-name ?name) put-Eqn ?eqn)
(return ?eqn)
)

(deffunction print-for-loop (?string ?name)
 (bind ?counter 1)
 (bind ?string (partial ?string))
 (loop-for-count (?cnt 1 ?*no-of-var*) do
  (if (<> 0 (str-compare "none" (nth$ 2
   (send (symbol-to-instance-name (sym-cat var ?cnt)) get-
      ArrayVar))))
  then
   (loop-for-count (?count 2 (+ (nth$ 1
    (send (symbol-to-instance-name (sym-cat var ?cnt)) get-ArrayVar))
     1)) do
    (bind ?oldstring (nth$ ?count
     (send (symbol-to-instance-name (sym-cat var ?cnt)) get-
     ArrayVar)))
    (bind ?newstring (str-replace-all ?string ?oldstring (sym-cat i
     ?counter)))
(bind ?change (sym-cat i ?counter))
(if (<> 0 (str-compare ?string ?newstring))
then
 (loop-for-count (?c 1 ?*no-of-var*) do
  (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
        name(sym-cat var ?c))get-Identifier)))
  then
   (bind ?var (sym-cat var ?c))
  )
 )
 (loop-for-count (?c 1 ?*no-of-const*) do
  (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
        name(sym-cat const ?c))get-Identifier)))
  then
   (bind ?var (sym-cat const ?c))
  )
 )
 (if (= 1 ?*gPROMS-boundary*)
then
 (if (= 0 (str-compare "Y" (send (symbol-to-instance-name
       ?var)get-Include)))
 then
  (if (= 0 (str-compare "N/A" (send(symbol-to-instance-name
       ?var)get-Distributed)))
  then
   (loop-for-count (?c 1 ?counter) do
    (printout ?*sim-file* " "))
   (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := 1
      TO " (nth$ ?count (send (symbol-to-instance-name (sym-cat
```

```
                    var ?cnt)) get-ArrayVar)) " DO" crlf)
                  (bind ?counter (+ ?counter 1))
                 else
                  (loop-for-count (?c 1 ?counter) do
                   (printout ?*sim-file* " "))
                  (bind ?split (str-index ":" (send(symbol-to-instance-name
                     ?var)get-Distributed)))
                  (bind ?len (str-length (send(symbol-to-instance-name
                     ?var)get-Distributed)))
                  (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := "
                   (sub-string 1 (- ?split 1) (send(symbol-to-instance-name
                     ?var)get-Distributed)))
;                 Lower bound
                  (if (= 0 (str-compare TRUE (integerp
                   (str-index ?oldstring (send(symbol-to-instance-name
                     ?name)get-LowerBound)))))
                   then
                    (printout ?*sim-file* "|+")
                   )
                   (printout ?*sim-file* " TO " (sub-string (+ ?split 1) ?len
                     (send(symbol-to-instance-name ?var)get-Distributed)))
;                  Upper bound
                   (if (= 0 (str-compare TRUE (integerp
                    (str-index ?oldstring (send(symbol-to-instance-name
                     ?name)get-UpperBound)))))
                    then
                     (printout ?*sim-file* "|-")
                    )
                    (printout ?*sim-file* " DO" crlf)
                    (bind ?counter (+ ?counter 1))
                  )
                 else
                  (str-replace-all ?string ?change " ")
                 )
                else
                 (if (<> 0 (str-compare "Y" (send (symbol-to-instance-name
                      ?var)get-Include)))
                  then
                   (if (= 0 (str-compare "N/A" (send(symbol-to-instance-name
                      ?var)get-Distributed)))
                    then
                     (loop-for-count (?c 1 ?counter) do
                      (printout ?*sim-file* " "))
                     (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := 1
                        TO " (nth$ ?count (send (symbol-to-instance-name (sym-cat
                        var ?cnt)) get-ArrayVar)) " DO" crlf)
                      (bind ?counter (+ ?counter 1))
                    else
                     (loop-for-count (?c 1 ?counter) do
                      (printout ?*sim-file* " "))
                     (bind ?split (str-index ":" (send(symbol-to-instance-name
                        ?var)get-Distributed)))
                     (bind ?len (str-length (send(symbol-to-instance-name
                        ?var)get-Distributed)))
                     (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := "
                      (sub-string 1 (- ?split 1) (send(symbol-to-instance-name
                        ?var)get-Distributed)))
;                    Lower bound
                     (if (= 0 (str-compare TRUE (integerp
                      (str-index ?oldstring (send(symbol-to-instance-name
                        ?name)get-LowerBound)))))
```

```
                then
                 (printout ?*sim-file* "|+")
                )
                (printout ?*sim-file* " TO " (sub-string (+ ?split 1) ?len
                    (send(symbol-to-instance-name ?var)get-Distributed)))
;               Upper bound
                (if (= 0 (str-compare TRUE (integerp
                 (str-index ?oldstring (send(symbol-to-instance-name
                    ?name)get-UpperBound)))))
                then
                 (printout ?*sim-file* "|-")
                )
                 (printout ?*sim-file* " DO" crlf)
                 (bind ?counter (+ ?counter 1))
                )
               else
                (str-replace-all ?string ?change " ")
               )
              )
             )
            (bind ?string ?newstring)
            )
           )
          )
         (loop-for-count (?cnt 1 ?*no-of-const*) do
          (if (<> 0 (str-compare "none" (nth$ 2
           (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
              ArrayVar))))
          then
           (loop-for-count (?count 2 (+ (nth$ 1
            (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
              ArrayVar)) 1)) do
            (bind ?oldstring (nth$ ?count
             (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
              ArrayVar)))
         (bind ?newstring (str-replace-all ?string ?oldstring (sym-cat i
           ?counter)))
         (if (<> 0 (str-compare ?string ?newstring))
         then
          (loop-for-count (?c 1 ?*no-of-var*) do
           (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
                 name(sym-cat var ?c))get-Identifier)))
           then
            (bind ?var (sym-cat var ?c))
           )
          )
          (loop-for-count (?c 1 ?*no-of-const*) do
           (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
                 name(sym-cat const ?c))get-Identifier)))
          then
           (bind ?var (sym-cat const ?c))
          )
         )
         (if (= 0 (str-compare "Y" (send (symbol-to-instance-name
                 ?var)get-Include)))
        then
         (if (= 0 (str-compare "N/A" (send(symbol-to-instance-name
                 ?var)get-Distributed)))
         then
          (loop-for-count (?c 1 ?counter) do
           (printout ?*sim-file* " "))
```

```
 (printout ?*sim-file* "    FOR " (sym-cat i ?counter) " := 1
      TO " (nth$ ?count (send (symbol-to-instance-name (sym-cat
      const ?cnt)) get-ArrayVar)) " DO" crlf)
 (bind ?counter (+ ?counter 1))
else
 (loop-for-count (?c 1 ?counter) do
  (printout ?*sim-file* " "))
 (bind ?split (str-index ":" (send(symbol-to-instance-name
      ?var)get-Distributed)))
 (bind ?len (str-length (send(symbol-to-instance-name ?var)get-
      Distributed)))
 (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := "
  (sub-string 1 (- ?split 1) (send(symbol-to-instance-name
      ?var)get-Distributed)))
;       Lower bound
 (if (= 0 (str-compare TRUE (integerp
  (str-index ?oldstring (send(symbol-to-instance-name
      ?name)get-LowerBound)))))
 then
  (printout ?*sim-file* "|+")
 )
 (printout ?*sim-file* " TO " (sub-string (+ ?split 1) ?len
      (send(symbol-to-instance-name ?var)get-Distributed)))
;       Upper bound
 (if (= 0 (str-compare TRUE (integerp
  (str-index ?oldstring (send(symbol-to-instance-name
      ?name)get-UpperBound)))))
 then
  (printout ?*sim-file* "|-")
 )
 (printout ?*sim-file* " DO" crlf)
 (bind ?counter (+ ?counter 1))
      )
     else
      (str-replace-all ?string ?change " ")
     )
    )
   (bind ?string ?newstring)
   )
  )
)
(loop-for-count (?n 1 ?counter)
 (printout ?*sim-file* " "))
(loop-for-count (?l 1 (send [par0] get-Number))
 (bind ?string (str-replace ?string
  (send (symbol-to-instance-name (sym-cat par ?l))get-ReplacedWith)
  (sym-cat "," (sub-string 2
   (str-length (send (symbol-to-instance-name (sym-cat par ?l))get-
     Value))
   (send (symbol-to-instance-name (sym-cat par ?l))get-Value)))))
 )
 (printout ?*sim-file* "    " ?string ";" crlf)
 (bind ?count ?counter)
 (loop-for-count (?cnt 1 (- ?counter 1)) do
  (bind ?count (- ?count 1))
  (loop-for-count (?c 1 ?count) do
   (printout ?*sim-file* " "))
  (printout ?*sim-file* "   END #For" crlf)
 )
 (return ?string)
)
```

```
(deffunction print-bdry-for-loop (?string ?name)
 (bind ?counter 1)
 (bind ?string (partial ?string))
 (bind ?equals (str-index "=" (send (symbol-to-instance-name ?name)
  get-FixedBound)))
 (bind ?LHS (sub-string 1 (- ?equals 1) (send (symbol-to-instance-
      name ?name)
  get-FixedBound)))
 (bind ?RHS (sub-string (+ ?equals 1) (str-length
  (send (symbol-to-instance-name ?name) get-FixedBound))
  (send (symbol-to-instance-name ?name) get-FixedBound)))
(bind ?string
 (str-replace-all ?string ?LHS ?RHS))
(loop-for-count (?cnt 1 ?*no-of-var*) do
 (if (<> 0 (str-compare "none" (nth$ 2
  (send (symbol-to-instance-name (sym-cat var ?cnt)) get-
     ArrayVar))))
 then
  (loop-for-count (?count 2 (+ (nth$ 1
   (send (symbol-to-instance-name (sym-cat var ?cnt)) get-ArrayVar))
    1)) do
   (bind ?oldstring (nth$ ?count
    (send (symbol-to-instance-name (sym-cat var ?cnt)) get-
     ArrayVar)))
   (bind ?newstring (str-replace-all ?string ?oldstring (sym-cat i
    ?counter)))
   (bind ?change (sym-cat i ?counter))
   (if (<> 0 (str-compare ?string ?newstring))
   then
    (loop-for-count (?c 1 ?*no-of-var*) do
     (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
          name(sym-cat var ?c))get-Identifier)))
     then
      (bind ?var (sym-cat var ?c))
     )
    )
    (loop-for-count (?c 1 ?*no-of-const*) do
     (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
          name(sym-cat const ?c))get-Identifier)))
     then
      (bind ?var (sym-cat const ?c))
     )
    )
    (if (= 0 (str-compare "Y" (send (symbol-to-instance-name
          ?var)get-Include)))
    then
     (if (= 0 (str-compare "N/A" (send(symbol-to-instance-name
          ?var)get-Distributed)))
     then
      (loop-for-count (?c 1 ?counter) do
       (printout ?*sim-file* " "))
      (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := 1 TO
           " (nth$ ?count (send (symbol-to-instance-name (sym-cat
          var ?cnt)) get-ArrayVar)) " DO" crlf)
      (bind ?counter (+ ?counter 1))
     else
      (loop-for-count (?c 1 ?counter) do
       (printout ?*sim-file* " "))
      (bind ?split (str-index ":" (send(symbol-to-instance-name
          ?var)get-Distributed)))
```

```
            (bind ?len (str-length (send(symbol-to-instance-name ?var)get-
                Distributed)))
            (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := "
             (sub-string 1 (- ?split 1) (send(symbol-to-instance-name
                ?var)get-Distributed)))
;           Lower bound
            (if (= 0 (str-compare TRUE (integerp
             (str-index ?oldstring (send(symbol-to-instance-name
                ?name)get-LowerBound)))))
            then
             (printout ?*sim-file* "|+")
            )
            (printout ?*sim-file* " TO " (sub-string (+ ?split 1) ?len
                (send(symbol-to-instance-name ?var)get-Distributed)))
;           Upper bound
            (if (= 0 (str-compare TRUE (integerp
             (str-index ?oldstring (send(symbol-to-instance-name
                ?name)get-UpperBound)))))
            then
             (printout ?*sim-file* "|-")
            )
            (printout ?*sim-file* " DO" crlf)
            (bind ?counter (+ ?counter 1))
           )
          else
           (str-replace-all ?string ?change " ")
          )
         )
        (bind ?string ?newstring)
       )
      )
     )
    (loop-for-count (?cnt 1 ?*no-of-const*) do
     (if (<> 0 (str-compare "none" (nth$ 2
      (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
         ArrayVar))))
     then
      (loop-for-count (?count 2 (+ (nth$ 1
       (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
         ArrayVar)) 1)) do
       (bind ?oldstring (nth$ ?count
        (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
         ArrayVar)))
       (bind ?newstring (str-replace-all ?string ?oldstring (sym-cat i
         ?counter)))
       (if (<> 0 (str-compare ?string ?newstring))
       then
        (loop-for-count (?c 1 ?*no-of-var*) do
         (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
               name(sym-cat var ?c))get-Identifier)))
         then
          (bind ?var (sym-cat var ?c))
         )
        )
        (loop-for-count (?c 1 ?*no-of-const*) do
         (if (= 0 (str-compare ?oldstring (send(symbol-to-instance-
               name(sym-cat const ?c))get-Identifier)))
         then
          (bind ?var (sym-cat const ?c))
         )
        )
```

```
(if (= 0 (str-compare "Y" (send (symbol-to-instance-name
      ?var)get-Include)))
then
(if (= 0 (str-compare "N/A" (send(symbol-to-instance-name
      ?var)get-Distributed)))
 then
  (loop-for-count (?c 1 ?counter) do
   (printout ?*sim-file* " "))
  (printout ?*sim-file* "    FOR " (sym-cat i ?counter) " := 1
      TO " (nth$ ?count (send (symbol-to-instance-name (sym-cat
      const ?cnt)) get-ArrayVar)) " DO" crlf)
  (bind ?counter (+ ?counter 1))
 else
  (loop-for-count (?c 1 ?counter) do
   (printout ?*sim-file* " "))
      (bind ?split (str-index ":" (send(symbol-to-instance-name
          ?var)get-Distributed)))
      (bind ?len (str-length (send(symbol-to-instance-name ?var)get-
          Distributed)))
      (printout ?*sim-file* "   FOR " (sym-cat i ?counter) " := "
       (sub-string 1 (- ?split 1) (send(symbol-to-instance-name
          ?var)get-Distributed)))
;     Lower bound
      (if (= 0 (str-compare TRUE (integerp
       (str-index ?oldstring (send(symbol-to-instance-name
          ?name)get-LowerBound)))))
      then
       (printout ?*sim-file* "|+")
      )
      (printout ?*sim-file* " TO " (sub-string (+ ?split 1) ?len
          (send(symbol-to-instance-name ?var)get-Distributed)))
;     Upper bound
      (if (= 0 (str-compare TRUE (integerp
       (str-index ?oldstring (send(symbol-to-instance-name
            ?name)get-UpperBound)))))
        then
         (printout ?*sim-file* "|-")
        )
        (printout ?*sim-file* " DO" crlf)
        (bind ?counter (+ ?counter 1))
       )
      else
       (str-replace-all ?string ?change " ")
      )
     )
    (bind ?string ?newstring)
   )
  )
)
(loop-for-count (?n 1 ?counter)
 (printout ?*sim-file* " "))
(loop-for-count (?l 1 (send [par0] get-Number))
 (bind ?string (str-replace ?string
  (send (symbol-to-instance-name (sym-cat par ?l))get-ReplacedWith)
  (sym-cat "," (sub-string 2
   (str-length (send (symbol-to-instance-name (sym-cat par ?l))get-
     Value))
   (send (symbol-to-instance-name (sym-cat par ?l))get-Value)))))
)
(printout ?*sim-file* "    " ?string ";" crlf)
(bind ?count ?counter)
```

```
  (loop-for-count (?cnt 1 (- ?counter 1)) do
   (bind ?count (- ?count 1))
   (loop-for-count (?c 1 ?count) do
    (printout ?*sim-file* " "))
   (printout ?*sim-file* "   END #For" crlf)
  )
  (return ?string)
)

(deffunction write-array-variables (?name ?stbracket)
 (bind ?count 0)
 (bind ?arrayvar
  (sub-string (+ ?stbracket 1) (- (str-length
  (send(symbol-to-instance-name ?name) get-Identifier)) 1)
   (send(symbol-to-instance-name ?name) get-Identifier)))
 (while (= 0 (str-compare TRUE (integerp (str-index "," ?arrayvar))))
      do
  (bind ?arrayvar (str-replace ?arrayvar "," " "))
  (bind ?count (+ ?count 1))
 )
 (send (symbol-to-instance-name ?name) put-ArrayVar
  (insert$ (explode$ ?arrayvar) 1 (+ ?count 1)))
 (return ?arrayvar)
)

(deffunction print-array-declare (?name)
 (bind ?stbracket (str-index "("
  (send(symbol-to-instance-name ?name)get-Identifier)))
 (printout ?*sim-file* "    " (sub-string 1 (- ?stbracket 1)
  (send(symbol-to-instance-name ?name)get-Identifier))
  "        AS ARRAY")
 (printout ?*sim-file*
  (sub-string ?stbracket(str-length
  (send(symbol-to-instance-name ?name)get-Identifier))
  (send(symbol-to-instance-name ?name)get-Identifier)))
 (write-array-variables ?name ?stbracket)
 (return ?name)
)

(deffunction print-dist-declare (?name)
 (bind ?stbracket (str-index "("
  (send(symbol-to-instance-name ?name)get-Identifier)))
 (printout ?*sim-file* "    " (sub-string 1 (- ?stbracket 1)
  (send(symbol-to-instance-name ?name)get-Identifier))
  "        AS DISTRIBUTION")
 (printout ?*sim-file*
  (sub-string ?stbracket(str-length
  (send(symbol-to-instance-name ?name)get-Identifier))
  (send(symbol-to-instance-name ?name)get-Identifier)))
 (write-array-variables ?name ?stbracket)
 (return ?name)
)

(deffunction print-gPROMS-const-array (?name)
 (bind ?stbracket (str-index "("
  (send(symbol-to-instance-name ?name)get-Identifier)))
 (print-array-declare ?name)
 (printout ?*sim-file* " OF REAL # "
  (send (symbol-to-instance-name ?name) get-Description)crlf)
 (return ?name)
)
```

```
(deffunction print-var-array-dist (?name ?cnt)
 (bind ?stbracket (str-index "("
  (send(symbol-to-instance-name ?name)get-Identifier)))
 (print-array-declare ?name)
 (printout ?*sim-file* " OF Type" ?cnt  " # "
  (send (symbol-to-instance-name ?name) get-Description) crlf)
  (send (symbol-to-instance-name ?name) put-Type
   (sym-cat Type ?cnt))
 (return ?name)
)

(deffunction print-var-dist-dist (?name ?cnt)
 (bind ?stbracket (str-index "("
  (send(symbol-to-instance-name ?name)get-Identifier)))
 (print-dist-declare ?name)
 (printout ?*sim-file* " OF Type" ?cnt  " # "
  (send (symbol-to-instance-name ?name) get-Description) crlf)
  (send (symbol-to-instance-name ?name) put-Type
   (sym-cat Type ?cnt))
 (return ?name)
)

(deffunction print-gPROMS-const (?cnt)
 (if (= 0 (str-compare ")" (sub-string (str-length (send
  (symbol-to-instance-name(sym-cat const ?cnt)) get-Identifier))
  (str-length (send(symbol-to-instance-name(sym-cat const ?cnt))get-
      Identifier))
  (send(symbol-to-instance-name(sym-cat const ?cnt))get-
      Identifier))))
 then
  (bind ?name (sym-cat const ?cnt))
  (print-gPROMS-const-array ?name)
 else
  (printout ?*sim-file* "    " (send (symbol-to-instance-name
   (sym-cat const ?cnt)) get-Identifier) "        AS ")
  (if (<> 0 (str-compare TRUE (integerp (send
   (symbol-to-instance-name(sym-cat const ?cnt)) get-Value))))
  then
   (printout ?*sim-file* "REAL # "
    (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
      Description) crlf)
  else
   (printout ?*sim-file* "INTEGER # "
    (send (symbol-to-instance-name (sym-cat const ?cnt)) get-
      Description) crlf))
 )
 (return ?cnt)
)

(deffunction print-dist ()
 (bind ?count 1)
 (loop-for-count (?cnt 1 ?*no-of-var*) do
  (if (<> 0 (str-compare "N/A"
   (send (symbol-to-instance-name(sym-cat var ?cnt)) get-
      Distributed)))
  then
   (if (= 1 ?count)
   then
    (printout ?*sim-file* "  DISTRIBUTION_DOMAIN" crlf)
   )
```

```
    (bind ?count (+ 1 ?count))
    (printout ?*sim-file* "     " (send (symbol-to-instance-name
     (sym-cat var ?cnt)) get-Identifier) "  AS  ("
     (send (symbol-to-instance-name(sym-cat var ?cnt)) get-
       Distributed) ")" crlf)
   )
  )
  (return ?count)
)

(deffunction array-dist-choice (?name ?cnt)
 (bind ?printcnt 0)
 (loop-for-count (?count 1 ?*no-of-var*) do
  (if (<> 0 (str-compare "N/A"
   (send (symbol-to-instance-name (sym-cat var ?count))get-
     Distributed)))
 then
  (if (= 0 (str-compare "TRUE" (integerp (str-index
   (send (symbol-to-instance-name (sym-cat var ?count)) get-
     Identifier)
   (send (symbol-to-instance-name ?name) get-Identifier)))))
  then
   (bind ?printcnt (+ ?printcnt 1))
   (if (= 1 ?printcnt)
   then
    (print-var-dist-dist ?name ?cnt)
   )
  else
   (bind ?printcnt (+ ?printcnt 1))
   (if (= 1 ?printcnt)
   then
    (print-var-array-dist ?name ?cnt)
    )
   )
  )
    else
     (bind ?printcnt (+ ?printcnt 1))
     (if (= 1 ?printcnt)
     then
      (print-var-array-dist ?name ?cnt)
     )
  )
  (return ?cnt)
)

(deffunction print-gPROMS-var (?cnt)
 (if (= 0 (str-compare "N/A"
  (send (symbol-to-instance-name(sym-cat var ?cnt)) get-
      Distributed)))
 then
(if (= 0 (str-compare ")" (sub-string (str-length (send
 (symbol-to-instance-name(sym-cat var ?cnt)) get-Identifier))
 (str-length (send(symbol-to-instance-name(sym-cat var ?cnt))get-
    Identifier))
 (send(symbol-to-instance-name(sym-cat var ?cnt))get-Identifier))))
then
 (bind ?name (sym-cat var ?cnt))
 (array-dist-choice ?name ?cnt)
else
 (printout ?*sim-file* "     " (send (symbol-to-instance-name
  (sym-cat var ?cnt)) get-Identifier) "        AS Type" ?cnt " # "
```

```
        (send (symbol-to-instance-name (sym-cat var ?cnt)) get-
          Description) crlf)
       (send (symbol-to-instance-name (sym-cat var ?cnt)) put-Type
        (sym-cat Type ?cnt))
     )
    )
    (return ?cnt)
   )

   (deffunction print-eqn-if (?name)
    (bind ?stelse (str-index "ELSE" ?name))
    (bind ?stthen (str-index "THEN" ?name))
    (bind ?ifthen (sub-string 1 (- ?stthen 1) ?name))
    (bind ?thenelse (sub-string (+ ?stthen 5) (- ?stelse 1) ?name))
    (bind ?elseend (sub-string (+ ?stelse 5) (str-length ?name) ?name))
    (printout ?*sim-file* "     " ?ifthen " THEN" crlf)
    (check-eqn ?thenelse ?name)
    (printout ?*sim-file* "     ELSE" crlf)
    (check-eqn ?elseend ?name)
    (printout ?*sim-file* "     END # If" crlf)
    (printout ?*sim-file* crlf)
    (return ?name)
   )

   (deffunction print-eqn-while (?name)
    (bind ?stdo (str-index "DO" ?name))
    (bind ?whiledo (sub-string 1 (- ?stdo 1) ?name))
    (bind ?doend (sub-string (+ ?stdo 3) (str-length ?name) ?name))
    (printout ?*sim-file* "     " ?whiledo " DO" crlf)
    (check-eqn ?doend ?name)
    (printout ?*sim-file* "     END # While" crlf)
    (return ?name)
   )

   (deffunction print-bdry-loop ()
    (bind ?count 1)
    (loop-for-count (?cnt 1 ?*no-of-eqn*) do
     (bind ?name (sym-cat eqn ?cnt))
     (bind ?eqn (send (symbol-to-instance-name ?name) get-Eqn))
     (bind ?eqn (str-remove ?eqn))
     (if (<> 0 (str-compare "All" (send (symbol-to-instance-name ?name)
      get-FixedBound)))
     then
      (bind ?*gPROMS-boundary* 1)
      (if (= 1 ?count)
      then
       (printout ?*sim-file* "  BOUNDARY")
      )
      (bind ?count (+ 1 ?count))
      (printout ?*sim-file* crlf "     # "
       (send (symbol-to-instance-name ?name) get-Description) crlf)
      (if (= 0 (str-compare "TRUE" (integerp (str-index "INTEGRAL"
         ?eqn))))
      then
       (change-integral ?eqn ?name)
      )
      (check-bdry-eqn ?eqn ?name)
     )
    )
    (return ?count)
   )
```

```
(deffunction print-gPROMS-eqn-loop (?cnt)
 (bind ?name (sym-cat eqn ?cnt))
 (if (= 1 ?*gPROMS-boundary*)
 then
  (if (= 0 (str-compare "All" (send (symbol-to-instance-name (sym-cat
      eqn ?cnt))
   get-FixedBound)))
  then
   (printout ?*sim-file* crlf "    # "
    (send (symbol-to-instance-name ?name) get-Description) crlf)
   (bind ?eqn (send (symbol-to-instance-name ?name) get-Eqn))
   (bind ?eqn (str-remove ?eqn))
   (if (= 0 (str-compare "TRUE" (integerp (str-index "INTEGRAL"
      ?eqn))))
   then
    (bind ?eqn (change-integral ?eqn ?name))
   )
   (check-eqn ?eqn ?name)
 )
else
 (if (= 0 (str-compare "All" (send (symbol-to-instance-name (sym-cat
     eqn ?cnt))
  get-FixedBound)))
 then
  (printout ?*sim-file* crlf "    # "
   (send (symbol-to-instance-name ?name) get-Description) crlf)
  (bind ?eqn (send (symbol-to-instance-name ?name) get-Eqn))
  (if (= 0 (str-compare "TRUE" (integerp (str-index "INTEGRAL"
     ?eqn))))
  then
   (bind ?eqn (change-integral ?eqn ?name))
   )
   (check-eqn ?eqn ?name)
  )
 )
 (return ?cnt)
)

(deffunction set-const (?cnt)
 (if (<> 0 (str-compare ")" (sub-string (str-length (send
  (symbol-to-instance-name(sym-cat const ?cnt)) get-Identifier))
  (str-length (send(symbol-to-instance-name(sym-cat const ?cnt))get-
  Identifier)) (send(symbol-to-instance-name(sym-cat const ?cnt))get-
  Identifier))))
 then
  (printout ?*sim-file* "      " (send (symbol-to-instance-name
   (sym-cat const ?cnt)) get-Identifier) "    := " (send
   (symbol-to-instance-name (sym-cat const ?cnt)) get-Value) "; # "
   (send (symbol-to-instance-name (sym-cat const ?cnt)) get-Unit)
   crlf)
else
 (bind ?stbracket (str-index "("
  (send(symbol-to-instance-name(sym-cat const ?cnt))get-
     Identifier)))
 (printout ?*sim-file* "      " (sub-string 1 (- ?stbracket 1)
  (send(symbol-to-instance-name(sym-cat const ?cnt))get-Identifier))
  "    := ")
 (if (= 0 (str-compare TRUE (integerp (str-index ","
  (send(symbol-to-instance-name(sym-cat const ?cnt))get-Value)))))
 then
```

```
    (printout ?*sim-file* "["
     (send(symbol-to-instance-name (sym-cat const ?cnt)) get-Value)
     "]; # "
    (send (symbol-to-instance-name (sym-cat const ?cnt)) get-Unit)
      crlf)
  else
   (printout ?*sim-file* (send(symbol-to-instance-name (sym-cat const
     ?cnt)) get-Value)
    "; # "
    (send (symbol-to-instance-name (sym-cat const ?cnt)) get-Unit)
      crlf)
  )
)
(return ?cnt)
)

(deffunction init-var (?cnt)
 (if (<> 0 (str-compare "unknown" (lowcase (send
  (symbol-to-instance-name(sym-cat var ?cnt)) get-Value))))
 then
  (if (<> 0 (str-compare ")" (sub-string (str-length (send
   (symbol-to-instance-name(sym-cat var ?cnt)) get-Identifier))
   (str-length (send(symbol-to-instance-name(sym-cat var ?cnt))get-
      Identifier))
   (send(symbol-to-instance-name(sym-cat var ?cnt))get-Identifier))))
  then
   (printout ?*sim-file* "       " (send (symbol-to-instance-name
    (sym-cat var ?cnt)) get-Identifier) "     = " (send
    (symbol-to-instance-name (sym-cat var ?cnt)) get-Value) "; # "
    (send (symbol-to-instance-name (sym-cat var ?cnt)) get-Unit)
      crlf)
  else
(if (= 0 (str-compare TRUE (integerp (str-index ","
 (send(symbol-to-instance-name(sym-cat var ?cnt))get-Value)))))
then
 (bind ?stbracket (str-index "("
  (send(symbol-to-instance-name(sym-cat var ?cnt))get-
   Identifier)))
 (bind ?stcomma (str-index ","
  (send(symbol-to-instance-name(sym-cat var ?cnt))get-Value)))
 (bind ?length (str-length
  (send(symbol-to-instance-name(sym-cat var ?cnt))get-Value)))
 (bind ?stcommaold 0)
 (bind ?count 1)
 (bind ?start 1)
 (while  (= 0 (str-compare TRUE (integerp ?stcomma))) do
  (bind ?stcomma (+ ?stcomma ?stcommaold))
  (bind ?value (sub-string ?start (- ?stcomma 1)
   (send (symbol-to-instance-name (sym-cat var ?cnt))get-Value)))
  (bind ?end (- ?stcomma 1))
(if (>= 0 (- ?start ?end))
then
 (printout ?*sim-file* "       " (sub-string 1 ?stbracket
  (send(symbol-to-instance-name(sym-cat var ?cnt))get-
 Identifier))
  ?count ")     = " ?value "; # "
  (send(symbol-to-instance-name (sym-cat var ?cnt))get-Unit)
 crlf)
)
(bind ?start (+ ?stcomma 1))
(bind ?count (+ ?count 1))
```

```
   (bind ?stcommaold ?stcomma)
   (bind ?stcomma (str-index "," (sub-string ?start ?length
    (send(symbol-to-instance-name(sym-cat var ?cnt))get-Value))))
  )
  (bind ?value (sub-string ?start ?length
   (send (symbol-to-instance-name (sym-cat var ?cnt))get-Value)))
  (printout ?*sim-file* "      " (sub-string 1 ?stbracket
   (send(symbol-to-instance-name(sym-cat var ?cnt))get-Identifier))
   ?count ")     = " ?value "; # "
   (send(symbol-to-instance-name (sym-cat var ?cnt))get-Unit) crlf)
 else
  (bind ?stbracket (str-index "("
   (send(symbol-to-instance-name(sym-cat var ?cnt))get-
    Identifier)))
 ; FOR loops.....
     (bind ?name (sym-cat (send(symbol-to-instance-name(sym-cat var
       ?cnt))get-Identifier)
      "     = "))
     (bind ?name (sym-cat ?name
      (send(symbol-to-instance-name (sym-cat var ?cnt)) get-Value)))
     (bind ?name (sym-cat ?name "; # "))
     (bind ?name (sym-cat ?name
      (send (symbol-to-instance-name (sym-cat var ?cnt)) get-Unit)))
     (bind ?name (str-remove ?name))
     (print-for-loop ?name (sym-cat var ?cnt))
    )
   )
  )
  (return ?cnt)
 )
 (deffunction set-var (?cnt)
  (if (<> 0 (str-compare ")" (sub-string (str-length (send
   (symbol-to-instance-name(sym-cat var ?cnt)) get-Identifier))
   (str-length (send(symbol-to-instance-name(sym-cat var ?cnt))get-
       Identifier))
   (send(symbol-to-instance-name(sym-cat var ?cnt))get-Identifier))))
  then
   (printout ?*sim-file* "      " (send (symbol-to-instance-name
    (sym-cat var ?cnt)) get-Identifier) "    := " (send
    (symbol-to-instance-name (sym-cat var ?cnt)) get-Set) "; # "
    (send (symbol-to-instance-name (sym-cat var ?cnt)) get-Unit) crlf)
  else
   (bind ?stbracket (str-index "("
    (send(symbol-to-instance-name(sym-cat var ?cnt))get-Identifier)))
   (printout ?*sim-file* "      " (sub-string 1 (- ?stbracket 1)
    (send(symbol-to-instance-name(sym-cat var ?cnt))get-Identifier))
    "    := ")
   (if (= 0 (str-compare TRUE (integerp (str-index ","
    (send(symbol-to-instance-name(sym-cat var ?cnt))get-Set)))))
   then
    (printout ?*sim-file* "["
     (send(symbol-to-instance-name (sym-cat var ?cnt)) get-Set)
     "]; # "
    (send (symbol-to-instance-name (sym-cat var ?cnt)) get-Unit) crlf)
   else
    (printout ?*sim-file* (send(symbol-to-instance-name (sym-cat var
       ?cnt)) get-Set)
     "; # "
     (send (symbol-to-instance-name (sym-cat var ?cnt)) get-Unit)
       crlf)
```

```
   )
  )
  (return ?cnt)
 )

 (deffunction assign-variables ()
  (bind ?count 0)
  (loop-for-count (?cnt 1 ?*no-of-var*) do
   (if (<> 0 (str-compare "unknown"
    (send (symbol-to-instance-name (sym-cat var ?cnt))get-Set)))
   then
    (bind ?count (+ ?count 1))
    (if (= 1 ?count)
    then
     (printout ?*sim-file* "  ASSIGN" crlf)
     (printout ?*sim-file* "    WITHIN " ?*unit-name* " DO" crlf)
    )
    (set-var ?cnt)
   )
  )
  (if (> ?count 0)
  then
   (printout ?*sim-file* "    END # Within" crlf crlf)
  )
  (return ?count)
 )

 (deffunction initialise-variables ()
  (bind ?count 0)
  (loop-for-count (?cnt 1 ?*no-of-var*) do
   (if (<> 0 (str-compare "unknown"
    (send (symbol-to-instance-name (sym-cat var ?cnt))get-Value)))
   then
    (bind ?count (+ ?count 1))
    (if (= 1 ?count)
    then
     (printout ?*sim-file* "  INITIAL" crlf)
     (printout ?*sim-file* "    WITHIN " ?*unit-name* " DO" crlf)
    )
    (init-var ?cnt)
   )
  )
  (if (> ?count 0)
  then
   (printout ?*sim-file* "    END # Within" crlf crlf)
  )
  (return ?count)
 )

 (deffunction print-math-method ()
 (bind ?count 0)
 (loop-for-count (?cnt 1 ?*no-of-var*) do
  (if (<> 0 (str-compare "N/A"
   (send (symbol-to-instance-name (sym-cat var ?cnt))get-
     MathMethod)))
  then
   (bind ?count (+ ?count 1))
   (if (= 1 ?count)
   then
    (printout ?*sim-file* crlf "    # Mathematical Solution Methods"
     crlf)
```

```
    )
    (printout ?*sim-file* "      "
     (send (symbol-to-instance-name (sym-cat var ?cnt))get-
       Identifier))
    (printout ?*sim-file* " := ["
     (send  (symbol-to-instance-name (sym-cat var ?cnt))get-
       MathMethod)
     "];" crlf)
   )
  )
  (return ?count)
)

(deffunction print-gPROMS-introduction ()
 (printout ?*sim-file*
"#*************************************************" crlf)
 (printout ?*sim-file* (str-cat "# Problem Title : " ?*problem-
   title*) crlf)
 (printout ?*sim-file* (str-cat "# Created : " (now)) crlf)
 (printout ?*sim-file* "# By IMIPS v0.03 gPROMS Translator" crlf)
 (printout ?*sim-file*
"#*************************************************" crlf)
 (printout ?*sim-file* "" crlf)
 (return ?*sim-file*)
)

(deffunction print-gPROMS-declaration ()
 (printout ?*sim-file* "DECLARE" crlf)
 (printout ?*sim-file* "  TYPE" crlf)
 (loop-for-count (?cnt 1 ?*no-of-var*) do
  (printout ?*sim-file* "    Type" ?cnt  "  =  1  : -1E6 : 1E6  UNIT
    = \"" (send (symbol-to-instance-name (sym-cat var ?cnt)) get-
     Unit) "\" # " (send (symbol-to-instance-name (sym-cat var ?cnt))
     get-Description) crlf)
 )
 (printout ?*sim-file* "END # Declare" crlf)
 (printout ?*sim-file* "" crlf)
 (return ?*sim-file*)
)

(deffunction print-gPROMS-model ()
 (printout ?*sim-file* "MODEL "?*prob-name* crlf)
 (printout ?*sim-file* "  PARAMETER" crlf)
 (loop-for-count (?cnt 1 ?*no-of-const*) do
  (print-gPROMS-const ?cnt))
 (printout ?*sim-file* crlf)
; DISTRIBUTION_DOMAIN
 (print-dist)
 (printout ?*sim-file* crlf)
 (printout ?*sim-file* "  VARIABLE" crlf)
 (loop-for-count (?cnt 1 ?*no-of-var*) do
  (print-gPROMS-var ?cnt))
 (printout ?*sim-file* crlf)
; BOUNDARY
 (print-bdry-loop)
 (printout ?*sim-file* crlf)
 (printout ?*sim-file* "  EQUATION")
 (loop-for-count (?cnt 1 ?*no-of-eqn*) do
  (print-gPROMS-eqn-loop ?cnt))
 (printout ?*sim-file* "END # Model "?*prob-name* crlf)
 (printout ?*sim-file* crlf)
```

```
 (return ?*sim-file*)
)

(deffunction print-gPROMS-process ()
 (printout ?*sim-file* "PROCESS Sim" crlf)
 (printout ?*sim-file* "  UNIT" crlf)
 (printout ?*sim-file* "    " ?*unit-name* " AS " ?*prob-name* crlf)
 (printout ?*sim-file* crlf)
 (printout ?*sim-file* "  SET" crlf)
 (printout ?*sim-file* "    WITHIN " ?*unit-name* " DO" crlf)
 (loop-for-count (?cnt 1 ?*no-of-const*) do
  (set-const ?cnt))
 (print-math-method)
 (printout ?*sim-file* "    END # Within" crlf)
 (printout ?*sim-file* crlf)
 (assign-variables)
 (initialise-variables)
 (printout ?*sim-file* crlf)
 (printout ?*sim-file* "  SOLUTIONPARAMETERS" crlf)
 (printout ?*sim-file* "    ReportingInterval := " ?*rep-int* ";"
      crlf)
 (printout ?*sim-file* "    BlockDecomposition := ON;" crlf)
 (printout ?*sim-file* crlf)
 (printout ?*sim-file* "  SCHEDULE" crlf)
 (printout ?*sim-file* "    CONTINUE FOR " ?*model-time* crlf)
 (printout ?*sim-file* crlf)
 (printout ?*sim-file* "END")
 (return ?*sim-file*)
)
```

IV.7. Simulink translator code file: Simulinktrans.clp

```
;*****************************************************************
;  Author :          Graham Clark                                 *
;  File Name:        simulinktrans.clp                            *
;  Last Updated:     20/12/00                                     *
;  Description:      SIMULINK input file creation translator version *
;*****************************************************************

;Forward declarations
(deffunction check-eqn (?eqn ?name))
(deffunction check-bdry-eqn (?eqn ?name))
(deffunction str-replace (?a ?b ?c))
(deffunction str-replace-all (?a ?b ?c))
(deffunction str-count-all (?a ?b))
(deffunction symbol-link (?eqn ?cnt))
(deffunction initial-identifier (?cnt))
(deffunction print-simulink-blocks ())
;*****************************************************************
;                              BLOCKS
;*****************************************************************

;Constant
(deffunction constant (?name ?desc ?value ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                Constant" crlf)
 (printout ?*sim-file* "      Name                     \"" ?name "\"" crlf)
 (printout ?*sim-file* "      Description              \"" ?desc "\"" crlf)
 (bind ?l (* 50 ?cnt))
 (bind ?t 20)
 (bind ?r (+ 30 (* 50 ?cnt)))
 (bind ?b 50)
 (printout ?*sim-file* "      Position                 [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation              \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "      Value                    \"" ?value "\"" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Integrator
(deffunction integrator (?l ?t ?r ?b ?initial ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                Integrator"
      crlf)
 (printout ?*sim-file* "      Name                     \"Integrator" ?cnt
      "\"" crlf)
 (printout ?*sim-file* "      Description              \"Description\""
      crlf)
(printout ?*sim-file* "      Ports                    [1, 1, 0, 0, 0]"
      crlf)
(printout ?*sim-file* "      Position                 [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
(printout ?*sim-file* "      Orientation              \"right\"" crlf)
      ;{right}, left, up, down
(printout ?*sim-file* "      ExternalReset            \"none\"" crlf)
(printout ?*sim-file* "      InitialConditionSource   \"internal\""
      crlf)
(printout ?*sim-file* "      InitialCondition         \"" ?initial
      "\"" crlf)
```

```
 (printout ?*sim-file* "      LimitOutput       off" crlf)
 (printout ?*sim-file* "      UpperSaturationLimit    \"inf\"" crlf)
 (printout ?*sim-file* "      LowerSaturationLimit    \"-inf\"" crlf)
 (printout ?*sim-file* "      ShowSaturationPort      off" crlf)
 (printout ?*sim-file* "      ShowStatePort           off" crlf)
 (printout ?*sim-file* "      AbsoluteTolerance       \"auto\"" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Derivative
(deffunction derivative (?l ?t ?r ?b ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                    Derivative"
      crlf)
 (printout ?*sim-file* "      Name                    \"Derivative" ?cnt
      "\"" crlf)
 (printout ?*sim-file* "      Description             \"Description\""
      crlf)
 (printout ?*sim-file* "      Position                     [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation             \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Logic
(deffunction logic (?l ?t ?r ?b ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                        Logic" crlf)
 (printout ?*sim-file* "      Name                    \"Logical Operator"
      ?cnt "\"" crlf)
 (printout ?*sim-file* "      Description             \"Description\""
      crlf)
 (printout ?*sim-file* "      Ports                   [2, 1, 0, 0, 0]"
      crlf)
 (printout ?*sim-file* "      Position                     [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation             \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "      Operator                         \"AND\"" crlf)
           ;{AND}, OR, NAND, NOR, XOR, NOT
 (printout ?*sim-file* "      Inputs                           \"2\"" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Math
(deffunction math (?l ?t ?r ?b ?cnt ?operator)
 (printout ?*sim-file* "    Block {" crlf)
(printout ?*sim-file* "      BlockType                        Math" crlf)
(printout ?*sim-file* "      Name                    \"Math Function" ?cnt
     "\"" crlf)
(printout ?*sim-file* "      Description             \"Description\""
     crlf)
(printout ?*sim-file* "      Ports                   [1, 1, 0, 0, 0]"
     crlf)
(printout ?*sim-file* "      Position                     [" ?l ", " ?t
     ", " ?r ", " ?b "]" crlf)
```

```
 (printout ?*sim-file* "      Orientation       \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "      Operator                  \"" ?operator
      "\"" crlf) ;{exp}, log, log10, square, sqrt, pow, reciprocal,
      hypot, rem, mod
 (printout ?*sim-file* "      OutputSignalType        \"auto\"" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Product
(deffunction product (?l ?t ?r ?b ?cnt ?type)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                 Product" crlf)
 (printout ?*sim-file* "      Name                  \"Product" ?cnt "\""
      crlf)
 (printout ?*sim-file* "      Description       \"Description\""
      crlf)
 (printout ?*sim-file* "      Ports             [2, 1, 0, 0, 0]"
      crlf)
 (printout ?*sim-file* "      Position                  [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation       \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "      Inputs                    \"" ?type "\""
      crlf)
 (printout ?*sim-file* "      SaturateOnIntegerOverflow     on" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Sum
(deffunction sum (?l ?t ?r ?b ?cnt ?type)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                 Sum" crlf)
 (printout ?*sim-file* "      Name                  \"Sum" ?cnt "\""
      crlf)
 (printout ?*sim-file* "      Description       \"Description\""
      crlf)
 (printout ?*sim-file* "      Ports             [2, 1, 0, 0, 0]"
      crlf)
 (printout ?*sim-file* "      Position                  [" ?l ", " ?t
", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation       \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "      ShowName                  off" crlf)
 (printout ?*sim-file* "      IconShape                 \"round\""
      crlf)
 (printout ?*sim-file* "      Inputs                    \"" ?type "\""
      crlf)
 (printout ?*sim-file* "      SaturateOnIntegerOverflow     on" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Mux
(deffunction mux (?l ?t ?r ?b ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                     Mux" crlf)
 (printout ?*sim-file* "      Name                      \"Mux" ?cnt "\""
      crlf)
```

```
  (printout ?*sim-file* "      Description      \"Description\""
       crlf)
  (printout ?*sim-file* "      Ports            [2, 1, 0, 0, 0]"
       crlf)
  (printout ?*sim-file* "      Position               [" ?l ", " ?t
       ", " ?r ", " ?b "]" crlf)
  (printout ?*sim-file* "      Orientation      \"right\"" crlf)
       ;{right}, left, up, down
  (printout ?*sim-file* "      ShowName               off" crlf)
  (printout ?*sim-file* "      Inputs                 \"2\"" crlf)
  (printout ?*sim-file* "      DisplayOption          \"bar\"" crlf)
  (printout ?*sim-file* "    }" crlf)
  (return ?*sim-file*)
)

;Scope
(deffunction scope (?l ?t ?r ?b ?cnt)
  (printout ?*sim-file* "    Block {" crlf)
  (printout ?*sim-file* "      BlockType                Scope" crlf)
  (printout ?*sim-file* "      Name             \"Scope" ?cnt "\""
       crlf)
  (printout ?*sim-file* "      Description      \"Description\""
       crlf)
  (printout ?*sim-file* "      Ports            [1, 0, 0, 0, 0]"
       crlf)
  (printout ?*sim-file* "      Position               [" ?l ", " ?t
       ", " ?r ", " ?b "]" crlf)
  (printout ?*sim-file* "      Orientation      \"right\"" crlf)
       ;{right}, left, up, down
  (printout ?*sim-file* "      Floating               off" crlf)
  (printout ?*sim-file* "      Location               [188, 365, 512,
       604]" crlf)
  (printout ?*sim-file* "      Open             off" crlf)
  (printout ?*sim-file* "      NumInputPorts          \"1\"" crlf)
  (printout ?*sim-file* "      TickLabels       \"OneTimeTick\""
       crlf)
  (printout ?*sim-file* "      ZoomMode               \"on\"" crlf)
  (printout ?*sim-file* "      List {" crlf)
  (printout ?*sim-file* "      ListType         AxesTitles" crlf)
  (printout ?*sim-file* "      axes1            "%<SignalLabel>""
       crlf)
  (printout ?*sim-file* "      }" crlf)
  (printout ?*sim-file* "      Grid             \"on\"" crlf)
  (printout ?*sim-file* "      TimeRange              \"auto\"" crlf)
  (printout ?*sim-file* "      YMin             \"-5\"" crlf)
  (printout ?*sim-file* "      YMax             \"5\"" crlf)
  (printout ?*sim-file* "      SaveToWorkspace        off" crlf)
  (printout ?*sim-file* "      SaveName               \"ScopeData\""
       crlf)
  (printout ?*sim-file* "      DataFormat
       \"StructureWithTime\"" crlf)
  (printout ?*sim-file* "      LimitMaxRows                on" crlf)
  (printout ?*sim-file* "      MaxRows                \"5000\"" crlf)
  (printout ?*sim-file* "      Decimation       \"1\"" crlf)
  (printout ?*sim-file* "      SampleInput      off" crlf)
  (printout ?*sim-file* "      SampleTime       \"0\"" crlf)
  (printout ?*sim-file* "    }" crlf)
  (return ?*sim-file*)
)

;Sign
```

```
(deffunction sign (?l ?t ?r ?b ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                  Signum" crlf)
 (printout ?*sim-file* "      Name                   \"Sign" ?cnt "\""
      crlf)
 (printout ?*sim-file* "      Description            \"Description\""
      crlf)
 (printout ?*sim-file* "      Position                   [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation            \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Trigonometry
(deffunction trig (?l ?t ?r ?b ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                      Trigonometry"
      crlf)
 (printout ?*sim-file* "      Name                   \"Trigonometric
      Function" ?cnt "\"" crlf)
 (printout ?*sim-file* "      Description            \"Description\""
      crlf)
 (printout ?*sim-file* "      Ports                  [1, 1, 0, 0, 0]"
      crlf)
 (printout ?*sim-file* "      Position                   [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation            \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "      Operator                       \"sin\"" crlf)
      ;{sin}, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh
 (printout ?*sim-file* "      OutputSignalType       \"auto\"" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;In
(deffunction in (?l ?t ?r ?b ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
 (printout ?*sim-file* "      BlockType                  Inport" crlf)
 (printout ?*sim-file* "      Name                   \"In" ?cnt "\"" crlf)
 (printout ?*sim-file* "      Description            \"Description\""
      crlf)
 (printout ?*sim-file* "      Position                   [" ?l ", " ?t
      ", " ?r ", " ?b "]" crlf)
 (printout ?*sim-file* "      Orientation            \"right\"" crlf)
      ;{right}, left, up, down
 (printout ?*sim-file* "      Port                   \"1\"" crlf)
 (printout ?*sim-file* "      PortWidth                  \"-1\"" crlf)
 (printout ?*sim-file* "      SampleTime             \"-1\"" crlf)
 (printout ?*sim-file* "      DataType                   \"auto\"" crlf)
 (printout ?*sim-file* "      SignalType             \"auto\"" crlf)
 (printout ?*sim-file* "      Interpolate            on" crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;Out
(deffunction out (?l ?t ?r ?b ?cnt)
 (printout ?*sim-file* "    Block {" crlf)
```

```
(printout ?*sim-file* "      BlockType                Outport" crlf)
(printout ?*sim-file* "      Name                  \"Out" ?cnt "\""
     crlf)
(printout ?*sim-file* "      Description           \"Description\""
     crlf)
(printout ?*sim-file* "      Position                 [" ?l ", " ?t
     ", " ?r ", " ?b "]" crlf)
(printout ?*sim-file* "      Orientation           \"right\"" crlf)
     ;{right}, left, up, down
(printout ?*sim-file* "      Port                  \"1\"" crlf)
(printout ?*sim-file* "      OutputWhenDisabled       \"held\"" crlf)
(printout ?*sim-file* "      InitialOutput            \"[]\"" crlf)
(printout ?*sim-file* "    }" crlf)
(return ?*sim-file*)
)

;*****************************************************************
; Block linking
;*****************************************************************

(deffunction print-link (?from ?fromport ?to ?toport)
(printout t "print-link" crlf)
 (printout ?*sim-file* "    Line {" crlf)
 (printout ?*sim-file* "      SrcBlock                \"" ?from "\""
     crlf)
 (printout ?*sim-file* "      SrcPort                  " ?fromport
     crlf)
 (printout ?*sim-file* "      Points                   [0, 0]" crlf)
 (printout ?*sim-file* "      DstBlock              \"" ?to "\"" crlf)
 (printout ?*sim-file* "      DstPort                  " ?toport crlf)
 (printout ?*sim-file* "    }" crlf)
 (return ?*sim-file*)
)

;*****************************************************************
; Constant checking and printing
;*****************************************************************
(deffunction print-simulink-const-array (?cnt)
(printout t "print-simulink-const-array" crlf)
  (bind ?name (send (symbol-to-instance-name (sym-cat const ?cnt))
      get-Identifier))
  (bind ?desc (send (symbol-to-instance-name (sym-cat const ?cnt))
      get-Description))
  (bind ?value (send (symbol-to-instance-name (sym-cat const ?cnt))
      get-Value))
  (bind ?value (str-cat "[" (str-replace-all ?value "," " ") "]"))
  (constant ?name ?desc ?value ?cnt)
 (return ?cnt)
)

(deffunction print-simulink-const (?cnt)
(printout t "print-simulink-const" crlf)
 (if (= 0 (str-compare ")" (sub-string (str-length (send
  (symbol-to-instance-name(sym-cat const ?cnt)) get-Identifier))
  (str-length (send(symbol-to-instance-name(sym-cat const ?cnt))get-
      Identifier))
  (send(symbol-to-instance-name(sym-cat const ?cnt))get-
      Identifier))))
 then
  (bind ?name (send (symbol-to-instance-name (sym-cat const ?cnt))
      get-Identifier))
```

```
  (if (= 0 (str-compare "TRUE" (integerp (str-index "," ?name))))
  then
   (bind ?*simulink-array* 1)
  else
   (print-simulink-const-array ?cnt)
  )
 else
  (bind ?name (send (symbol-to-instance-name (sym-cat const ?cnt))
      get-Identifier))
  (bind ?desc (send (symbol-to-instance-name (sym-cat const ?cnt))
      get-Description))
  (bind ?value (send (symbol-to-instance-name (sym-cat const ?cnt))
      get-Value))
  (constant ?name ?desc ?value ?cnt)
  )
  (return ?cnt)
 )

 (deffunction print-integral (?cnt)
  (bind ?*t* (+ 20 (* 50 ?*eqn-no*)))
  (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
  (bind ?*l* (+ ?*l* 50))
  (bind ?*r* (+ ?*r* 50))
  (derivative ?*l* ?*t* ?*r* ?*b* ?cnt)
  (bind ?*symbol-name* (sym-cat "Integrator" ?cnt))
  (return ?*symbol-name*)
 )

 (deffunction print-differential (?cnt)
  (loop-for-count (?var-count 1 ?*no-of-var*) do
   (bind ?name (string-to-symbol (str-cat "var" ?var-count)))
   (bind ?diff (sym-cat "$" (send (symbol-to-instance-name ?name) get-
       Identifier)))
   (bind ?*t* (+ 20 (* 50 ?*eqn-no*)))
   (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
   (bind ?*l* (+ ?*l* 50))
   (bind ?*r* (+ ?*r* 50))
  ; Set initial value
   (if (= 0 (str-compare "unknown" (send (symbol-to-instance-name
       ?name) get-Value)))
   then
    (bind ?initial "0")
   else
    (bind ?initial (send (symbol-to-instance-name ?name) get-Value))
   )
   (integrator ?*l* ?*t* ?*r* ?*b* ?initial ?cnt)
  )
  (bind ?*symbol-name* (sym-cat "Derivative" ?cnt))
  (return ?*symbol-name*)
 )

 (deffunction print-logarithm (?cnt)
  (bind ?*t* (+ 20 (* 50 ?*eqn-nq*)))
  (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
  (bind ?*l* (+ ?*l* 50))
  (bind ?*r* (+ ?*r* 50))
  (math ?*l* ?*t* ?*r* ?*b* ?cnt "log")
  (bind ?*symbol-name* (sym-cat "Math Function" ?cnt))
  (return ?*symbol-name*)
 )
```

```
(deffunction print-exponential (?cnt)
 (bind ?*t* (+ 20 (* 50 ?*eqn-no*)))
 (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
 (bind ?*l* (+ ?*l* 50))
 (bind ?*r* (+ ?*r* 50))
 (math ?*l* ?*t* ?*r* ?*b* ?cnt "exp")
 (bind ?*symbol-name* (sym-cat "Math Function" ?cnt))
 (return ?*symbol-name*)
)

(deffunction print-multiply (?cnt)
 (bind ?*t* (+ 20 (* 50 ?*eqn-no*)))
 (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
 (bind ?*l* (+ ?*l* 50))
 (bind ?*r* (+ ?*r* 50))
 (product ?*l* ?*t* ?*r* ?*b* ?cnt "***")
 (bind ?*symbol-name* (sym-cat "Product" ?cnt))
 (return ?*symbol-name*)
)

(deffunction print-divide (?cnt)
 (bind ?*t* (+ 20 (* 50 ?*eqn-no*)))
 (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
 (bind ?*l* (+ ?*l* 50))
 (bind ?*r* (+ ?*r* 50))
 (product ?*l* ?*t* ?*r* ?*b* ?cnt "*/")
 (bind ?*symbol-name* (sym-cat "Product" ?cnt))
 (return ?*symbol-name*)
)

(deffunction print-add (?cnt)
 (bind ?*t* (+ 20 (* 50 ?*eqn-no*)))
 (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
 (bind ?*l* (+ ?*l* 50))
 (bind ?*r* (+ ?*r* 50))
 (sum ?*l* ?*t* ?*r* ?*b* ?cnt "|++")
 (bind ?*symbol-name* (sym-cat "Sum" ?cnt))
 (return ?*symbol-name*)
)

(deffunction print-subtract (?cnt)
 (bind ?*t* (+ 20 (* 50 ?*eqn-no*)))
 (bind ?*b* (+ 50 (* 50 ?*eqn-no*)))
 (bind ?*l* (+ ?*l* 50))
 (bind ?*r* (+ ?*r* 50))
 (sum ?*l* ?*t* ?*r* ?*b* ?cnt "|+-")
 (bind ?*symbol-name* (sym-cat "Sum" ?cnt))
 (return ?*symbol-name*)
)

;*****************************************************************
; Equation Checking and block printing
;*****************************************************************

(deffunction print-simulink-eqn-loop (?cnt)
(printout t "print-simulink-eqn-loop" crlf)
 (bind ?name (sym-cat eqn ?cnt))
 (if (= 0 (str-compare "All" (send (symbol-to-instance-name (sym-cat
      eqn ?cnt))
  get-FixedBound)))
 then
```

```
(if (= 0 ?*simulink-partial*)
then
 (bind ?*eqn-no* ?cnt)
 (bind ?*eqn* (send (symbol-to-instance-name ?name) get-Eqn))
 ;check for partial derivatives
 (if (neq 0 (str-compare "TRUE" (integerp (str-index "PARTIAL"
    ?*eqn*))))
 then
  (if (eq 0 (str-compare "TRUE" (integerp (str-index "=" ?*eqn*))))
  then
   (bind ?LHS (sub-string 1 (- (str-index "=" ?*eqn*) 1) ?*eqn*))
   (bind ?marker 0)
   (loop-for-count (?count 1 ?*no-of-var*) do
     (bind ?varname (send (symbol-to-instance-name(sym-cat var
          ?count))get-Identifier))
      (if (<> 1 ?marker) then
       (if (eq 0 (str-compare "TRUE" (integerp (str-index ?varname
          ?LHS))))
       then
;Variable to solve for here...
        (bind ?*eqn* (sub-string (+ 1 (str-index "=" ?*eqn*)) (str-
           length ?*eqn*) ?*eqn*))
        (bind ?marker 1)
       )
      )
     )
    )
    (bind ?marker 0)
    (loop-for-count (?varno 1 ?*no-of-var*) do
     (bind ?varname (send (symbol-to-instance-name(sym-cat var
      ?varno))get-Identifier))
     (if (eq 0 (str-compare "TRUE" (integerp (str-index "("
      ?varname))))
     then
      (if (eq 0 (str-compare "TRUE" (integerp (str-index ?varname
           ?*eqn*))))
      then
       (bind ?newname (sub-string 1 (- (str-index "(" ?varname) 1)
           ?varname))
       (bind ?*eqn* (str-replace ?*eqn* ?varname ?newname))
      )
     )
    )
    (loop-for-count (?constno 1 ?*no-of-const*) do
     (bind ?constname (send (symbol-to-instance-name(sym-cat const
           ?constno))get-Identifier))
     (if (eq 0 (str-compare "TRUE" (integerp (str-index "("
           ?constname))))
     then
      (if (eq 0 (str-compare "TRUE" (integerp (str-index ?constname
           ?*eqn*))))
      then
       (bind ?newname (sub-string 1 (- (str-index "(" ?constname) 1)
           ?constname))
       (bind ?*eqn* (str-replace ?*eqn* ?constname ?newname))
      )
     )
    )
    (if (= 0 (str-compare "TRUE" (integerp (str-index " " ?*eqn*))))
    then (bind ?*eqn* (str-replace ?*eqn* " " "")))
    ;bracket pairing checking
```

```
(if (= 0 (str-compare "TRUE" (integerp (str-index "(" ?*eqn*))))
then (bind ?no-of-bracket (str-count-all ?*eqn* "("))
(if (<> 0 ?no-of-bracket)
then
 (bind ?bracket-number 1)
 (bind ?new-eqn ?*eqn*)
 (while (= 0 (str-compare "TRUE" (integerp (str-index "("
        ?*eqn*)))) do
  (bind ?end-bracket (str-index ")" ?new-eqn))
  (bind ?new-eqn (sub-string 1 ?end-bracket ?new-eqn))
  (bind ?count (- (str-length ?new-eqn) 1))
  (bind ?partial-eqn (sub-string ?count (str-length ?new-eqn)
        ?new-eqn))
  (while (<> 0 (str-compare "TRUE" (integerp (str-index "("
        ?partial-eqn)))) do
   (bind ?count (- ?count 1))
   (bind ?partial-eqn (sub-string ?count (str-length ?new-eqn)
        ?new-eqn))
  )
  (bind ?new-eqn (str-replace ?*eqn* ?partial-eqn (sym-cat "Fake"
        ?cnt ?bracket-number)))

(bind ?eqn ?partial-eqn)
(if (= 0 (str-compare "TRUE" (integerp (str-index "=" ?eqn)))) then
 (bind ?pos (str-index "=" ?*eqn*))
 (bind ?eqn (sub-string (+ 1 ?pos) (str-length ?*eqn*) ?*eqn*))
)
(if (= 0 (str-compare "TRUE" (integerp (str-index " " ?eqn)))) then
 (if (= 1 (str-index " " ?eqn)) then (bind ?eqn (sub-string 2 (str-
     length ?eqn) ?eqn)))
)
(loop-for-count (?count 1 ?*no-of-const*) do
 (if (= 0 (str-compare "TRUE" (integerp (str-index
           (send (symbol-to-instance-name (sym-cat "const"
            ?count))get-Identifier) ?eqn)))) then
   (if (= 1 (str-index (send (symbol-to-instance-name (sym-cat
      "const" ?count))get-Identifier) ?eqn))
   then (bind ?eqn (sub-string (+ 1 (str-length (send (symbol-to-
      instance-name (sym-cat "const" ?count)) get-Identifier)))
      (str-length ?eqn) ?eqn)))
 )
)
(if (= 0 (str-compare "TRUE" (integerp (str-index "(" ?eqn)))) then
 (bind ?eqn (sub-string 2 (str-length ?*eqn*) ?*eqn*))
)
(if (= 0 (str-compare "TRUE" (integerp (str-index ")" ?eqn)))) then
 (bind ?eqn (sub-string 2 (str-length ?*eqn*) ?*eqn*))
)
(symbol-link ?eqn ?cnt)

    (bind ?*eqn* ?new-eqn)
    (bind ?bracket-number (+ ?bracket-number 1))
   )
  )
  else
  (initial-identifier ?cnt)
  )
else
(message-box "Simulink cannot, at present, deal with partial
   derivatives.
```

```
  As your problem includes these please try another simulator."
   wxOK 1 0 "Simulink Error")
(bind ?*simulink-partial* 1)
(window-delete ?*trans-frame*)
(printout ?*sim-file* "    Annotation {" crlf)
(printout ?*sim-file* "      Position                    [200, 100]"
   crlf)
(printout ?*sim-file* "      Text                        \"Unable to
   translate to Simulink File.\"")
(printout ?*sim-file* "                              \" Partial
   Derivatives used in problem.\"" crlf)
(printout ?*sim-file* "    }" crlf)
(printout ?*sim-file* "  }" crlf)
(printout ?*sim-file* "}" crlf)
(close)
   )
  )
 )
 (return ?cnt)
)

(deffunction print-const-links ()
(printout t "print-const-links" crlf)
 (return ?*sim-file*)
)

(deffunction initial-identifier (?cnt)
(printout t "initial-identifier" crlf)
 (bind ?eqn ?*eqn*)
 (if (= 0 (str-compare "TRUE" (integerp (str-index "=" ?eqn)))) then
  (bind ?pos (str-index "=" ?*eqn*))
  (bind ?eqn (sub-string (+ 1 ?pos) (str-length ?*eqn*) ?*eqn*))
 )
(if (= 0 (str-compare "TRUE" (integerp (str-index " " ?eqn)))) then
 (if (= 1 (str-index " " ?eqn)) then (bind ?eqn (sub-string 2 (str-
     length ?eqn) ?eqn)))
)
(loop-for-count (?count 1 ?*no-of-const*) do
 (if (= 0 (str-compare "TRUE" (integerp (str-index
     (send (symbol-to-instance-name (sym-cat "const" ?count))get-
     Identifier) ?eqn)))) then
  (if (= 1 (str-index (send (symbol-to-instance-name (sym-cat
     "const" ?count))get-Identifier) ?eqn))
  then (bind ?eqn (sub-string (+ 1 (str-length (send (symbol-to-
          instance-name (sym-cat "const" ?count)) get-Identifier)))
          (str-length ?eqn) ?eqn)))
 )
)
(if (= 0 (str-compare "TRUE" (integerp (str-index "(" ?eqn)))) then
 (bind ?eqn (sub-string 2 (str-length ?*eqn*) ?*eqn*))
 )
 (if (= 0 (str-compare "TRUE" (integerp (str-index ")" ?eqn)))) then
  (bind ?eqn (sub-string 2 (str-length ?*eqn*) ?*eqn*))
 )
 (symbol-link ?eqn ?cnt)
 (return ?cnt)
)

(deffunction symbol-link (?eqn ?cnt)
(printout t "symbol-link" crlf)
 (bind ?from ?*symbol-name*)
```

```
(if (= 0 (str-compare "TRUE" (integerp (str-index "+" ?eqn))))
then
 (if (= 1 (str-index "+" ?eqn)) then (print-add ?*sum-cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 2 ?length ?eqn))
 (bind ?to (str-cat "Sum" ?*sum-cnt*))
 (bind ?*sum-cnt* (+ 1 ?*sum-cnt*))
 (bind ?print-link 0)
))
(if (= 0 (str-compare "TRUE" (integerp (str-index "*" ?eqn))))
then
 (if (= 1 (str-index "*" ?eqn)) then (print-multiply ?*mult-cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 2 ?length ?eqn))
 (bind ?to (str-cat "Product" ?*mult-cnt*))
 (bind ?*mult-cnt* (+ 1 ?*mult-cnt*))
 (bind ?print-link 0)
))
(if (= 0 (str-compare "TRUE" (integerp (str-index "/" ?eqn))))
then
 (if (= 1 (str-index "/" ?eqn)) then (print-divide ?*mult-cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 2 ?length ?eqn))
 (bind ?to (str-cat "Product" ?*mult-cnt*))
 (bind ?*mult-cnt* (+ 1 ?*mult-cnt*))
 (bind ?print-link 0)
))
(if (= 0 (str-compare "TRUE" (integerp (str-index "-" ?eqn))))
then
 (if (= 1 (str-index "-" ?eqn)) then (print-subtract ?*sum-cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 2 ?length ?eqn))
 (bind ?to (str-cat "Sum" ?*sum-cnt*))
 (bind ?*sum-cnt* (+ 1 ?*sum-cnt*))
 (bind ?print-link 0)
))
(if (= 0 (str-compare "TRUE" (integerp (str-index "$" ?eqn))))
then
 (if (= 1 (str-index "$" ?eqn)) then (print-differential ?*diff-
     cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 2 ?length ?eqn))
 (bind ?to (str-cat "Derivative" ?*diff-cnt*))
 (bind ?*diff-cnt* (+ 1 ?*diff-cnt*))
 (bind ?print-link 0)
))
(if (= 0 (str-compare "TRUE" (integerp (str-index "LOG" ?eqn))))
then
 (if (= 1 (str-index "LOG" ?eqn)) then (print-logarithm ?*log-cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 4 ?length ?eqn))
 (bind ?to (str-cat "Math Function" ?*log-cnt*))
 (bind ?*log-cnt* (+ 1 ?*log-cnt*))
 (bind ?print-link 0)
))
(if (= 0 (str-compare "TRUE" (integerp (str-index "EXP" ?eqn))))
then
 (if (= 1 (str-index "EXP" ?eqn)) then (print-exponential ?*log-
     cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 4 ?length ?eqn))
```

```
 (bind ?to (str-cat "Math Function" ?*log-cnt*))
 (bind ?*log-cnt* (+ 1 ?*log-cnt*))
 (bind ?print-link 0)
))
(if (= 0 (str-compare "TRUE" (integerp (str-index "INTEGRAL"
     ?eqn))))
then
 (if (= 1 (str-index "INTEGRAL" ?eqn)) then (print-integral ?*int-
     cnt*)
 (bind ?length (str-length ?eqn))
 (bind ?eqn (sub-string 9 ?length ?eqn))
 (bind ?to (str-cat "Integrator" ?*int-cnt*))
 (bind ?*int-cnt* (+ 1 ?*int-cnt*))
 (bind ?print-link 0)
 ))
 (bind ?*eqn* ?eqn)
 (return ?eqn)
)


;*****************************************************************
;*                Print out SIMULINK input file                  *
;*****************************************************************

;Printout general information in file
(deffunction print-simulink-introduction ()
(printout t "print-simulink-introduciton" crlf)
 (printout ?*sim-file* "Model {" crlf)
 (printout ?*sim-file* (str-cat "  Name                    \"" ?*prob-
     name* "\"") crlf)
 (printout ?*sim-file* "  Version           3.00" crlf)
 (printout ?*sim-file* "  SimParamPage            \"Solver\"" crlf)
 (printout ?*sim-file* "  SampleTimeColors        off" crlf)
 (printout ?*sim-file* "  InvariantConstants      off" crlf)
 (printout ?*sim-file* "  WideVectorLines   off" crlf)
 (printout ?*sim-file* "  ShowLineWidths    off" crlf)
 (printout ?*sim-file* "  ShowPortDataTypes       off" crlf)
 (printout ?*sim-file* "  StartTime         \"0.0\"" crlf)
 (printout ?*sim-file* (str-cat "  StopTime              \"" ?*model-
     time* "\"") crlf)
 (printout ?*sim-file* "  SolverMode              \"Auto\"" crlf)
 (printout ?*sim-file* "  Solver            \"ode45\"" crlf)
 (printout ?*sim-file* "  RelTol            \"1e-3\"" crlf)
 (printout ?*sim-file* "  AbsTol            \"auto\"" crlf)
 (printout ?*sim-file* "  Refine            \"1\"" crlf)
 (printout ?*sim-file* "  MaxStep           \"auto\"" crlf)
 (printout ?*sim-file* "  InitialStep             \"auto\"" crlf)
 (printout ?*sim-file* "  FixedStep         \"auto\"" crlf)
 (printout ?*sim-file* "  MaxOrder          5" crlf)
 (printout ?*sim-file* "  OutputOption
     \"RefineOutputTimes\"" crlf)
 (printout ?*sim-file* "  OutputTimes             \"[]\"" crlf)
 (printout ?*sim-file* "  LoadExternalInput       off" crlf)
 (printout ?*sim-file* "  ExternalInput           \"[t, u]\"" crlf)
 (printout ?*sim-file* "  SaveTime          on" crlf)
 (printout ?*sim-file* "  TimeSaveName            \"tout\"" crlf)
 (printout ?*sim-file* "  SaveState         off" crlf)
 (printout ?*sim-file* "  StateSaveName           \"xout\"" crlf)
 (printout ?*sim-file* "  SaveOutput              on" crlf)
 (printout ?*sim-file* "  OutputSaveName    \"yout\"" crlf)
 (printout ?*sim-file* "  LoadInitialState        off" crlf)
```

```
(printout ?*sim-file* "  InitialState                \"xInitial\"" crlf)
(printout ?*sim-file* "  SaveFinalState     off" crlf)
(printout ?*sim-file* "  FinalStateName     \"xFinal\"" crlf)
(printout ?*sim-file* "  SaveFormat                 \"Matrix\"" crlf)
(printout ?*sim-file* "  LimitMaxRows               off" crlf)
(printout ?*sim-file* "  MaxRows            \"1000\"" crlf)
(printout ?*sim-file* "  Decimation                 \"1\"" crlf)
(printout ?*sim-file* "  AlgebraicLoopMsg           \"warning\"" crlf)
(printout ?*sim-file* "  MinStepSizeMsg     \"warning\"" crlf)
(printout ?*sim-file* "  UnconnectedInputMsg        \"warning\"" crlf)
(printout ?*sim-file* "  UnconnectedOutputMsg       \"warning\"" crlf)
(printout ?*sim-file* "  UnconnectedLineMsg         \"warning\"" crlf)
(printout ?*sim-file* "  InheritedTsInSrcMsg        \"warning\"" crlf)
(printout ?*sim-file* "  IntegerOverflowMsg         \"warning\"" crlf)
(printout ?*sim-file* "  UnnecessaryDatatypeConvMsg \"none\"" crlf)
(printout ?*sim-file* "  Int32ToFloatConvMsg        \"warning\"" crlf)
(printout ?*sim-file* "  SignalLabelMismatchMsg     \"none\"" crlf)
(printout ?*sim-file* "  ConsistencyChecking        \"off\"" crlf)
(printout ?*sim-file* "  ZeroCross                  on" crlf)
(printout ?*sim-file* "  SimulationMode     \"normal\"" crlf)
(printout ?*sim-file* "  BlockDataTips                 on" crlf)
(printout ?*sim-file* "  BlockParametersDataTip        on" crlf)
(printout ?*sim-file* "  BlockAttributesDataTip        off" crlf)
(printout ?*sim-file* "  BlockPortWidthsDataTip        off" crlf)
(printout ?*sim-file* "  BlockDescriptionStringDataTip     off"
    crlf)
(printout ?*sim-file* "  BlockMaskParametersDataTip off" crlf)
(printout ?*sim-file* "  ToolBar                    on" crlf)
(printout ?*sim-file* "  StatusBar                  on" crlf)
(printout ?*sim-file* "  BrowserShowLibraryLinks off" crlf)
(printout ?*sim-file* "  BrowserLookUnderMasks   off" crlf)
(printout ?*sim-file* "  OptimizeBlockIOStorage  on" crlf)
(printout ?*sim-file* "  BufferReuse                on" crlf)
(printout ?*sim-file* "  BooleanDataType    off" crlf)
(printout ?*sim-file* "  RTWSystemTargetFile        \"grt.tlc\"" crlf)
(printout ?*sim-file* "  RTWInlineParameters        off" crlf)
(printout ?*sim-file* "  RTWRetainRTWFile           off" crlf)
(printout ?*sim-file* "  RTWTemplateMakefile
    \"grt_default_tmf\"" crlf)
(printout ?*sim-file* "  RTWMakeCommand     \"make_rtw\"" crlf)
(printout ?*sim-file* "  RTWGenerateCodeOnly        off" crlf)
(printout ?*sim-file* "  ExtModeMexFile     \"ext_comm\"" crlf)
(printout ?*sim-file* "  ExtModeBatchMode           off" crlf)
(printout ?*sim-file* "  ExtModeTrigType    \"manual\"" crlf)
(printout ?*sim-file* "  ExtModeTrigMode    \"oneshot\"" crlf)
(printout ?*sim-file* "  ExtModeTrigPort    \"1\"" crlf)
(printout ?*sim-file* "  ExtModeTrigElement         \"any\"" crlf)
(printout ?*sim-file* "  ExtModeTrigDuration        1000" crlf)
(printout ?*sim-file* "  ExtModeTrigHoldOff         0" crlf)
(printout ?*sim-file* "  ExtModeTrigDelay           0" crlf)
(printout ?*sim-file* "  ExtModeTrigDirection       \"rising\"" crlf)
(printout ?*sim-file* "  ExtModeTrigLevel           0" crlf)
(printout ?*sim-file* "  ExtModeArchiveMode         \"off\"" crlf)
(printout ?*sim-file* "  ExtModeAutoIncOneShot      off" crlf)
(printout ?*sim-file* "  ExtModeIncDirWhenArm       off" crlf)
(printout ?*sim-file* "  ExtModeAddSuffixToVar      off" crlf)
(printout ?*sim-file* "  ExtModeWriteAllDataToWs off" crlf)
(printout ?*sim-file* "  ExtModeArmWhenConnect      off" crlf)
(printout ?*sim-file* (str-cat "  Created                    \"" (now)
    "\"") crlf)
(printout ?*sim-file* "  UpdateHistory
```

```
      \"UpdateHistoryNever\"" crlf)
(printout ?*sim-file* "  ModifiedByFormat          \"%<Auto>\"" crlf)
(printout ?*sim-file* "  LastModifiedBy    \"IMIPS v0.03 SIMULINK
     Translator\"" crlf)
(printout ?*sim-file* "  ModifiedDateFormat        \"%<Auto>\"" crlf)
(printout ?*sim-file* (str-cat "  LastModifiedDate     \"" (now)
     "\"") crlf)
(printout ?*sim-file* "  ModelVersionFormat
     \"1.%<AutoIncrement:1>\"" crlf)
(printout ?*sim-file* "  ConfigurationManager    \"none\"" crlf)
(printout ?*sim-file* "  BlockDefaults {" crlf)
(printout ?*sim-file* "    Orientation               \"right\"" crlf)
(printout ?*sim-file* "    ForegroundColor           \"black\"" crlf)
(printout ?*sim-file* "    BackgroundColor           \"white\"" crlf)
(printout ?*sim-file* "    DropShadow                off" crlf)
(printout ?*sim-file* "    NamePlacement      \"normal\"" crlf)
(printout ?*sim-file* "    FontName                  \"Helvetica\""
     crlf)
(printout ?*sim-file* "    FontSize                  10" crlf)
(printout ?*sim-file* "    FontWeight                \"normal\"" crlf)
(printout ?*sim-file* "    FontAngle                 \"normal\"" crlf)
(printout ?*sim-file* "    ShowName                  on" crlf)
(printout ?*sim-file* "  }" crlf)
(printout ?*sim-file* "  AnnotationDefaults {" crlf)
(printout ?*sim-file* "    HorizontalAlignment       \"center\"" crlf)
(printout ?*sim-file* "    VerticalAlignment         \"middle\"" crlf)
(printout ?*sim-file* "    ForegroundColor           \"black\"" crlf)
(printout ?*sim-file* "    BackgroundColor           \"white\"" crlf)
(printout ?*sim-file* "    DropShadow                off" crlf)
(printout ?*sim-file* "    FontName                  \"Helvetica\""
     crlf)
(printout ?*sim-file* "    FontSize                  10" crlf)
(printout ?*sim-file* "    FontWeight                \"normal\"" crlf)
(printout ?*sim-file* "    FontAngle                 \"normal\"" crlf)
(printout ?*sim-file* "  }" crlf)
(printout ?*sim-file* "  LineDefaults {" crlf)
(printout ?*sim-file* "    FontName                  \"Helvetica\""
     crlf)
(printout ?*sim-file* "    FontSize                  9" crlf)
(printout ?*sim-file* "    FontWeight                \"normal\"" crlf)
(printout ?*sim-file* "    FontAngle                 \"normal\"" crlf)
(printout ?*sim-file* "  }" crlf)
(printout ?*sim-file* "  System {" crlf)
(printout ?*sim-file* (str-cat "    Name              \"" ?*prob-name*
     "\"") crlf)
(printout ?*sim-file* "    Location                  [2, 70, 1276,
     974]" crlf)
(printout ?*sim-file* "    Open                   on" crlf)
(printout ?*sim-file* "    ModelBrowserVisibility  off" crlf)
(printout ?*sim-file* "    ModelBrowserWidth       200" crlf)
(printout ?*sim-file* "    ScreenColor               \"automatic\""
     crlf)
(printout ?*sim-file* "    PaperOrientation          \"landscape\""
     crlf)
(printout ?*sim-file* "    PaperPositionMode         \"auto\"" crlf)
(printout ?*sim-file* "    PaperType                 \"a4letter\""
     crlf)
(printout ?*sim-file* "    PaperUnits                \"inches\"" crlf)
(printout ?*sim-file* "    ZoomFactor                \"100\"" crlf)
(printout ?*sim-file* "    AutoZoom                  on" crlf)
(printout ?*sim-file* "    ReportName                \"simulink-
```

```
        default.rpt\"" crlf)
  (return ?*sim-file*)
 )

 (deffunction print-simulink-blocks ()
 (printout t "print-simulink-blocks" crlf)
 ;Constants
  (loop-for-count (?cnt 1 ?*no-of-const*) do
   (print-simulink-const ?cnt))
    (if (= 0 ?*simulink-array*)
    then
 ;Check equations and print out relevant blocks and links
     (loop-for-count (?cnt 1 ?*no-of-eqn*) do
      (print-simulink-eqn-loop ?cnt))
     (if (neq 1 ?*simulink-partial*) then
      (print-const-links))
    else
     (message-box "The translator cannot, at present, deal with multi
       value constant arrays (matrices).
       As your problem includes these please try another simulator."
       wxOK 1 0 "Translation Error")
     (window-delete ?*trans-frame*)
 ;    (window-delete ?*vars-frame*)
     (printout ?*sim-file* "    Annotation {" crlf)
     (printout ?*sim-file* "      Position                [200, 100]"
       crlf)
     (printout ?*sim-file* "      Text                    \"Unable to
       translate to Simulink File.\"")
     (printout ?*sim-file* "                            \" Multi value
       constant arrays (matrices) used in problem.\"" crlf)
     (printout ?*sim-file* "  }" crlf)
     (printout ?*sim-file* "}" crlf)
     (close)
      )
    ;Close file
     (printout ?*sim-file* "  }" crlf)
     (printout ?*sim-file* "}" crlf)
     (return ?*sim-file*)
     (bind ?*simulink-partial 0)
    )
```

V. Examples of Simulator Code

V.1. Translated Simple Batch Extraction Simulink Input File (Case Study 1)

```
Model {
  Name                    "cstudy1"
  Version            3.00
  SimParamPage            "Solver"
  SampleTimeColors        off
  InvariantConstants      off
  WideVectorLines    off
  ShowLineWidths     off
  ShowPortDataTypes       off
  StartTime          "0.0"
  StopTime           "20"
  SolverMode              "Auto"
  Solver             "ode45"
  RelTol             "1e-3"
  AbsTol             "auto"
  Refine             "1"
  MaxStep            "auto"
  InitialStep             "auto"
  FixedStep          "auto"
  MaxOrder           5
  OutputOption            "RefineOutputTimes"
  OutputTimes             "[]"
  LoadExternalInput       off
  ExternalInput           "[t, u]"
  SaveTime           on
  TimeSaveName            "tout"
  SaveState          off
  StateSaveName           "xout"
  SaveOutput              on
  OutputSaveName     "yout"
  LoadInitialState        off
  InitialState            "xInitial"
  SaveFinalState     off
  FinalStateName     "xFinal"
  SaveFormat              "Matrix"
  LimitMaxRows            off
  MaxRows            "1000"
  Decimation              "1"
  AlgebraicLoopMsg        "warning"
  MinStepSizeMsg     "warning"
  UnconnectedInputMsg     "warning"
  UnconnectedOutputMsg    "warning"
  UnconnectedLineMsg      "warning"
  InheritedTsInSrcMsg     "warning"
  IntegerOverflowMsg      "warning"
  UnnecessaryDatatypeConvMsg "none"
  Int32ToFloatConvMsg     "warning"
  SignalLabelMismatchMsg  "none"
  ConsistencyChecking     "off"
  ZeroCross          on
  SimulationMode     "normal"
  BlockDataTips           on
  BlockParametersDataTip  on
  BlockAttributesDataTip  off
  BlockPortWidthsDataTip  off
  BlockDescriptionStringDataTip     off
```

```
  BlockMaskParametersDataTip off
  ToolBar              on
  StatusBar            on
  BrowserShowLibraryLinks off
  BrowserLookUnderMasks   off
  OptimizeBlockIOStorage  on
  BufferReuse             on
  BooleanDataType   off
  RTWSystemTargetFile     "grt.tlc"
  RTWInlineParameters     off
  RTWRetainRTWFile        off
  RTWTemplateMakefile     "grt_default_tmf"
  RTWMakeCommand    "make_rtw"
  RTWGenerateCodeOnly     off
  ExtModeMexFile    "ext_comm"
  ExtModeBatchMode        off
  ExtModeTrigType   "manual"
  ExtModeTrigMode   "oneshot"
  ExtModeTrigPort   "1"
  ExtModeTrigElement      "any"
  ExtModeTrigDuration     1000
  ExtModeTrigHoldOff      0
  ExtModeTrigDelay        0
  ExtModeTrigDirection    "rising"
  ExtModeTrigLevel        0
  ExtModeArchiveMode      "off"
  ExtModeAutoIncOneShot   off
  ExtModeIncDirWhenArm    off
  ExtModeAddSuffixToVar   off
  ExtModeWriteAllDataToWs off
  ExtModeArmWhenConnect   off
  Created           "Thu Feb 03 09:45:58 2000"
  UpdateHistory           "UpdateHistoryNever"
  ModifiedByFormat        "%<Auto>"
  LastModifiedBy    "planteng"
  ModifiedDateFormat      "%<Auto>"
  LastModifiedDate        "Wed Nov 15 12:58:39 2000"
  ModelVersionFormat      "1.%<AutoIncrement:27>"
  ConfigurationManager    "none"
  BlockDefaults {
    Orientation             "right"
    ForegroundColor         "black"
    BackgroundColor         "white"
    DropShadow              off
    NamePlacement     "normal"
    FontName                "Helvetica"
    FontSize                10
    FontWeight              "normal"
    FontAngle               "normal"
    ShowName                on
  }
  AnnotationDefaults {
    HorizontalAlignment     "center"
    VerticalAlignment       "middle"
    ForegroundColor         "black"
    BackgroundColor         "white"
    DropShadow              off
    FontName                "Helvetica"
    FontSize                10
    FontWeight              "normal"
    FontAngle               "normal"
```

```
}
LineDefaults {
  FontName                 "Helvetica"
  FontSize                 9
  FontWeight               "normal"
  FontAngle                "normal"
}
System {
  Name                 "cstudy1"
  Location                 [460, 172, 922, 432]
  Open                 on
  ModelBrowserVisibility  off
  ModelBrowserWidth        200
  ScreenColor              "automatic"
  PaperOrientation         "landscape"
  PaperPositionMode        "auto"
  PaperType                "a4letter"
  PaperUnits               "inches"
  ZoomFactor               "100"
  AutoZoom                 on
  ReportName               "simulink-default.rpt"
  Block {
    BlockType                Integrator
    Name                 "Integrator1"
    Description          "Description"
    Ports                [1, 1, 0, 0, 0]
    Position                 [100, 70, 130, 100]
    ExternalReset            "none"
    InitialConditionSource   "internal"
    InitialCondition         "1"
    LimitOutput          off
    UpperSaturationLimit     "inf"
    LowerSaturationLimit     "-inf"
    ShowSaturationPort       off
    ShowStatePort            off
    AbsoluteTolerance        "auto"
  }
  Block {
    BlockType                Integrator
    Name                 "Integrator7"
    Description          "Description"
    Ports                [1, 1, 0, 0, 0]
    Position                 [100, 120, 130, 150]
    ExternalReset            "none"
    InitialConditionSource   "internal"
    InitialCondition         "0"
    LimitOutput          off
    UpperSaturationLimit     "inf"
    LowerSaturationLimit     "-inf"
    ShowSaturationPort       off
    ShowStatePort            off
    AbsoluteTolerance        "auto"
  }
  Block {
    BlockType                Constant
    Name                 "K"
    Description          "Mass Transfer Coefficient"
    Position                 [50, 20, 80, 50]
    Value                "2.5"
  }
  Block {
```

```
  BlockType                 Constant
  Name                      "M"
  Description               "Equilibrium Constant"
  Position                  [100, 20, 130, 50]
  Value                     "0.8"
}
Block {
  BlockType                 Product
  Name                      "Product10"
  Description               "Description"
  Ports                     [2, 1, 0, 0, 0]
  Position                  [250, 122, 280, 153]
  Inputs                    "*/"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                 Product
  Name                      "Product12"
  Description               "Description"
  Ports                     [2, 1, 0, 0, 0]
  Position                  [100, 172, 130, 203]
  Inputs                    "/*"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                 Product
  Name                      "Product2"
  Description               "Description"
  Ports                     [2, 1, 0, 0, 0]
  Position                  [150, 72, 180, 103]
  Inputs                    "2"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                 Product
  Name                      "Product3"
  Ports                     [2, 1, 0, 0, 0]
  Position                  [200, 72, 230, 103]
  Inputs                    "*/"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                 Product
  Name                      "Product4"
  Description               "Description"
  Ports                     [2, 1, 0, 0, 0]
  Position                  [250, 72, 280, 103]
  Inputs                    "/*"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                 Product
  Name                      "Product8"
  Description               "Description"
  Ports                     [2, 1, 0, 0, 0]
  Position                  [150, 122, 180, 153]
  Inputs                    "2"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                 Product
```

```
    Name                    "Product9"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [200, 122, 230, 153]
    Inputs                          "*/"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Sum
    Name                    "Sum11"
    Description             "Description"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [300, 130, 320, 150]
    NamePlacement                   "alternate"
    ShowName                        off
    IconShape                       "round"
    Inputs                          "|-+"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Sum
    Name                    "Sum13"
    Description             "Description"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [155, 175, 175, 195]
    ShowName                        off
    IconShape                       "round"
    Inputs                          "|+-"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Sum
    Name                    "Sum5"
    Description             "Description"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [300, 75, 320, 95]
    ShowName                        off
    IconShape                       "round"
    Inputs                          "|+-"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Sum
    Name                    "Sum6"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [340, 75, 360, 95]
    ShowName                        off
    IconShape                       "round"
    Inputs                          "|-+"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Scope
    Name                    "X"
    Ports                   [1, 0, 0, 0, 0]
    Position                        [15, 64, 45, 96]
    Orientation             "left"
    Floating                        off
    Location                        [46, 383, 370, 622]
    Open                    on
    NumInputPorts                   "1"
    TickLabels              "OneTimeTick"
```

```
    ZoomMode                "on"
    List {
    ListType            AxesTitles
    axes1               "%<SignalLabel>"
    }
    Grid                "on"
    TimeRange               "auto"
    YMin                "0"
    YMax                "1"
    SaveToWorkspace         off
    SaveName                "ScopeData"
    DataFormat          "StructureWithTime"
    LimitMaxRows            on
    MaxRows                 "5000"
    Decimation          "1"
    SampleInput         off
    SampleTime          "0"
  }
  Block {
    BlockType               Constant
    Name                "X0"
    Description         "Initial Concentration"
    Position                [250, 20, 280, 50]
    Value               "1"
  }
  Block {
    BlockType               Scope
    Name                "Y"
    Ports               [1, 0, 0, 0, 0]
    Position                [15, 114, 45, 146]
    Orientation         "left"
    Floating                off
    Location                [48, 107, 372, 346]
    Open                on
    NumInputPorts           "1"
    TickLabels          "OneTimeTick"
    ZoomMode                "on"
    List {
    ListType            AxesTitles
    axes1               "%<SignalLabel>"
    }
    Grid                "on"
    TimeRange               "auto"
    YMin                "0"
    YMax                "1"
    SaveToWorkspace         off
    SaveName                "ScopeData"
    DataFormat          "StructureWithTime"
    LimitMaxRows            on
    MaxRows                 "5000"
    Decimation          "1"
    SampleInput         off
    SampleTime          "0"
  }
  Block {
    BlockType               Scope
    Name                "Z"
    Description         "Description"
    Ports               [1, 0, 0, 0, 0]
    Position                [15, 164, 45, 196]
    Orientation         "left"
```

```
    Floating                off
    Location                [43, 663, 367, 902]
    Open                on
    NumInputPorts           "1"
    TickLabels          "OneTimeTick"
    ZoomMode                "on"
    List {
    ListType            AxesTitles
    axes1               "%<SignalLabel>"
    }
    Grid                "on"
    TimeRange               "auto"
    YMin                "0"
    YMax                "1"
    SaveToWorkspace         off
    SaveName                "ScopeData"
    DataFormat          "StructureWithTime"
    LimitMaxRows            on
    MaxRows                 "5000"
    Decimation          "1"
    SampleInput         off
    SampleTime          "0"
  }
  Block {
    BlockType               Constant
    Name                "vol1"
    Description         "Batch Volume"
    Position                [150, 20, 180, 50]
    Value               "10"
  }
  Block {
    BlockType               Constant
    Name                "vol2"
    Description         "Batch Volume"
    Position                [200, 20, 230, 50]
    Value               "40"
  }
  Line {
    SrcBlock                "Product12"
    SrcPort                 1
    Points                  [-75, 0]
    DstBlock                "Z"
    DstPort                 1
  }
  Line {
    SrcBlock                "Product10"
    SrcPort                 1
    Points                  [-150, 0]
    DstBlock                "Product8"
    DstPort                 1
  }
  Line {
    SrcBlock                "Product4"
    SrcPort                 1
    Points                  [0, 5]
    DstBlock                "Product2"
    DstPort                 2
  }
  Line {
    SrcBlock                "Sum5"
    SrcPort                 1
```

```
    Points                  [-90, 0]
    DstBlock                "Product10"
    DstPort                 1
  }
  Line {
    SrcBlock                "Product8"
    SrcPort                 1
    Points                  [0, -5]
    DstBlock                "Integrator7"
    DstPort                 1
  }
  Line {
    SrcBlock                "Sum11"
    SrcPort                 1
    Points                  [-90, 0]
    DstBlock                "Product4"
    DstPort                 2
  }
  Line {
    SrcBlock                "Product2"
    SrcPort                 1
    Points                  [-100, 0]
    DstBlock                "Integrator1"
    DstPort                 1
  }
  Line {
    SrcBlock                "Integrator1"
    SrcPort                 1
    Points                  [0, -20; 30, 0]
    Branch {
    Points                  [0, 70; 145, 0]
    DstBlock          "Sum11"
    DstPort                 2
  }
  Branch {
  Points                      [-10, 0; 0, 70]
  Branch {
    DstBlock          "X"
    DstPort           1
  }
  Branch {
    Points            [30, 0]
    Branch {
      DstBlock                  "Sum5"
      DstPort           1
    }
    Branch {
      Points            [0, -20]
      DstBlock                  "Sum13"
      DstPort           2
      }
    }
    }
  }
  Line {
    SrcBlock                "K"
    SrcPort                 1
    Points                  [0, 0]
    Branch {
    DstBlock          "Product8"
    DstPort                 2
```

```
    }
    Branch {
    DstBlock            "Sum6"
    DstPort                  1
    }
  }
  Line {
    SrcBlock                 "Sum6"
    SrcPort                  1
    Points                   [0, -5]
    DstBlock                 "Product2"
    DstPort                  1
  }
  Line {
    SrcBlock                 "vol2"
    SrcPort                  1
    DstBlock                 "Product10"
    DstPort                  2
  }
  Line {
    SrcBlock                 "vol1"
    SrcPort                  1
    Points                   [50, 0]
    DstBlock                 "Product4"
    DstPort                  1
  }
  Line {
    SrcBlock                 "X0"
    SrcPort                  1
    Points                   [0, 0]
    Branch {
    Points                   [0, 100; -145, 0]
    DstBlock            "Sum13"
    DstPort                  1
    }
    Branch {
    DstBlock            "Product12"
    DstPort                  1
    }
  }
  Line {
    SrcBlock                 "Sum13"
    SrcPort                  1
    Points                   [-95, 0]
    DstBlock                 "Product12"
    DstPort                  2
  }
  Line {
    SrcBlock                 "M"
    SrcPort                  1
    Points                   [0, 0]
    Branch {
    Points                   [0, 30; 175, 0; 0, 30]
    DstBlock            "Product3"
    DstPort                  2
    }
    Branch {
    Points                   [215, 0; 0, 110]
    DstBlock            "Product9"
    DstPort                  2
    }
```

```
        }
        Line {
          SrcBlock                "Integrator7"
          SrcPort                 1
          Points                  [0, -25; 145, 0; 0, -5]
          Branch {
          Points                  [-100, 5]
          Branch {
            Points                [5, 0]
            DstBlock              "Product3"
            DstPort               1
          }
          Branch {
            Points                [-120, 0]
            DstBlock              "Y"
            DstPort               1
          }
          }
          Branch {
          Points                  [0, -5; -100, 0; 0, 30]
          DstBlock              "Product9"
          DstPort                 1
          }
        }
        Line {
          SrcBlock                "Product3"
          SrcPort                 1
          Points                  [0, 20]
          DstBlock                "Sum5"
          DstPort                 2
        }
        Line {
          SrcBlock                "Product9"
          SrcPort                 1
          DstBlock                "Sum11"
          DstPort                 1
        }
      }
    }
```

V.2. Translated Simple Batch Extraction gPROMS Input File (Case Study 1)

```
#*************************************************************
# Problem Title : Simple Batch Extraction
# Created : Thu Nov 02 13:23:15 2000
# By IMIPS v0.03 gPROMS Translator
#*************************************************************

DECLARE
  TYPE
    Type1  =  1  : -1E6 : 1E6  UNIT = "" # Concentration in liquid
    Type2  =  1  : -1E6 : 1E6  UNIT = "" # Concentration in Vapor
    Type3  =  1  : -1E6 : 1E6  UNIT = "" # Extent
END # Declare

MODEL CStudy1
  PARAMETER
    K          AS REAL # Mass Transfer Coefficient
    M          AS REAL # Equilibrium Constant
    Vol1          AS REAL # Batch Volume
    Vol2          AS REAL # Batch Volume
    X0          AS REAL # Initial Concentration


  VARIABLE
    X          AS Type1 # Concentration in liquid
    Y          AS Type2 # Concentration in Vapor
    Z          AS Type3 # Extent


  EQUATION
    # Change in concentration in Vol1
    $X = -K*((X-Y)/(M*Vol1));

    # Change in concentration in Vol2
    $Y = K*((X-Y)/(M*Vol2));

    # Conversion
    Z = (X0-X)/X;
END # Model CStudy1

PROCESS Sim
  UNIT
    CS1 AS CStudy1

  SET
    WITHIN CS1 DO
      K     := 2.5; #
      M     := 0.8; #
      Vol1     := 10; #
      Vol2     := 40; #
      X0     := 1; #
    END # Within

INITIAL
  WITHIN CS1 DO
    X      = 1; #
    Y      = 0; #
  END # Within
```

```
  SOLUTIONPARAMETERS
    ReportingInterval := 0.5;
    BlockDecomposition := ON;

  SCHEDULE
    CONTINUE FOR 20

END
```

V.3. Translated Catalytic Tube Reactor gPROMS Input File (Case Study 2)

```
#*************************************************
# Problem Title : Catalytic Tube Reactor
# Created : Wed Nov 01 12:05:07 2000
# By IMIPS v0.03 gPROMS Translator
#*************************************************

DECLARE
  TYPE
    Type1   =  1   : -1E6 : 1E6  UNIT = "m" # Axial position
    Type2   =  1   : -1E6 : 1E6  UNIT = "m" # Radial position
    Type3   =  1   : -1E6 : 1E6  UNIT = "kg/mol" # Concentration
    Type4   =  1   : -1E6 : 1E6  UNIT = "Pa" # Partial pressures
    Type5   =  1   : -1E6 : 1E6  UNIT = "K" # Temperature
    Type6   =  1   : -1E6 : 1E6  UNIT = "mol/kg.s.Pa2" # Reaction
constant
    Type7   =  1   : -1E6 : 1E6  UNIT = "mol/kg.s" # Reaction rate
    Type8   =  1   : -1E6 : 1E6  UNIT = "mol/m3" # Feed concentration
    Type9   =  1   : -1E6 : 1E6  UNIT = "Pa" # Feed partial pressures
    Type10   =  1   : -1E6 : 1E6  UNIT = "K" # Feed temperature
    Type11   =  1   : -1E6 : 1E6  UNIT = "m/s" # Superficial gas
velocity
    Type12   =  1   : -1E6 : 1E6  UNIT = "kg/s" # Coolant flowrate
    Type13   =  1   : -1E6 : 1E6  UNIT = "K" # Coolant temperature
    Type14   =  1   : -1E6 : 1E6  UNIT = "K" # Coolant temperature in
    Type15   =  1   : -1E6 : 1E6  UNIT = "W" # Total heat load absorbed
by coolant
END # Declare

MODEL CStudy1
  PARAMETER
  L          AS REAL # Reactor Length
  R          AS REAL # Reactor Radius
  rhob          AS REAL # Bed Density
  rhof          AS REAL # Fluid Density
  Cpf         AS REAL # Fluid Specific Heat Capacity
  Dz          AS REAL # Axial Diffusivity
  Dr          AS REAL # Radial Diffusivity
  kz          AS REAL # Axial Conductivity
  kr          AS REAL # Radial Conductivity
  BedVoid          AS REAL # Bed Voidage Fraction
  deltaH         AS REAL # Reaction Enthalpy
  A          AS REAL # Pre-Exponential Arrhenius Constant
  E          AS REAL # Activation Energy
  hw          AS REAL # Wall Heat Transfer Coefficient
  Rg          AS REAL # Ideal Gas Coefficient
  rhoc          AS REAL # Coolant Density
  Cpc         AS REAL # Coolant Specific Heat Capacity
  Vc          AS REAL # Cooling Jacket Volume
  Area          AS REAL # Overall Heat Transfer Area
  OverallU          AS REAL # Overall Heat Transfer Coefficient
  NoComp         AS INTEGER # Number of Components Present
  Nu          AS ARRAY(NoComp) OF REAL #

DISTRIBUTION_DOMAIN
  ax  AS  (0:L)
  rad  AS  (0:R)

VARIABLE
  C          AS DISTRIBUTION(NoComp,ax,rad) OF Type3 # Concentration
```

```
    P          AS DISTRIBUTION(NoComp,ax,rad) OF Type4 # Partial
pressures
    T          AS DISTRIBUTION(ax,rad) OF Type5 # Temperature
    k          AS DISTRIBUTION(ax,rad) OF Type6 # Reaction constant
    Rate          AS DISTRIBUTION(ax,rad) OF Type7 # Reaction rate
    Cfeed          AS ARRAY(NoComp) OF Type8 # Feed concentration
    Pfeed          AS ARRAY(NoComp) OF Type9 # Feed partial pressures
    Tfeed          AS Type10 # Feed temperature
    u          AS Type11 # Superficial gas velocity
    Fc          AS Type12 # Coolant flowrate
    Tc          AS Type13 # Coolant temperature
    Tcin          AS Type14 # Coolant temperature in
    Q          AS Type15 # Total heat load absorbed by coolant

  BOUNDARY
# Boundary condition at reactor entrance (z=0)
FOR i1 := 0 TO R DO
 -BedVoid*Dz*PARTIAL(C(,0,i1),ax) = u*(Cfeed()-C(,0,i1));
END #For

# Boundary condition at reactor entrance (z=0)
FOR i1 := 0 TO R DO
 -kz*PARTIAL(T(0,i1),ax) = rhof*Cpf*u*(Tfeed-T(0,i1));
END #For

# Boundary condition at reactor exit (z=L)
FOR i1 := 0 TO R DO
 PARTIAL(C(,L,i1),ax) = 0;
END #For

# Boundary condition at reactor exit (z=L)
FOR i1 := 0 TO R DO
 PARTIAL(T(L,i1),ax) = 0;
END #For

# Boundary condition at reactor centre (r=0)
FOR i1 := 0|+ TO L|- DO
 PARTIAL(C(,i1,0),rad) = 0;
END #For

# Boundary condition at reactor centre (r=0)
FOR i1 := 0|+ TO L|- DO
 PARTIAL(T(i1,0),rad) = 0;
END #For

# Boundary condition at reactor perimeter (r=R)
FOR i1 := 0|+ TO L|- DO
 PARTIAL(C(,i1,R),rad) = 0;
END #For

    # Boundary condition at reactor perimeter (r=R)
    FOR i1 := 0|+ TO L|- DO
     -kr*PARTIAL(T(i1,R),rad) = hw*(T(i1,R)-Tc);
    END #For

  EQUATION
    # Component mass balance
    FOR i1 := 0|+ TO L|- DO
     FOR i2 := 0|+ TO R|- DO
      $C(,i1,i2) = -u*PARTIAL(C(,i1,i2),ax)
+BedVoid*Dz*PARTIAL(C(,i1,i2),ax,ax)
```

```
+Bedvoid*Dr*PARTIAL(C(,i1,i2),rad,rad)
+BedVoid*Dr*(1/i2)*PARTIAL(C(,i1,i2),rad) +rhob*Nu()*Rate(i1,i2);
      END #For
     END #For

     # Energy balance
     FOR i1 := 0|+ TO L|- DO
      FOR i2 := 0|+ TO R|- DO
       rhof*Cpf*$T(i1,i2) = -rhof*Cpf*u*PARTIAL(T(i1,i2),ax)
+kz*PARTIAL(T(i1,i2),ax,ax) +kr*PARTIAL(T(i1,i2),rad,rad)
+kr*(1/i2)*PARTIAL(T(i1,i2),rad) +rhob*Rate(i1,i2)*(deltaH);
      END #For
     END #For

# Ideal gas law
FOR i1 := 0 TO L DO
 FOR i2 := 0 TO R DO
  P(,i1,i2) = C(,i1,i2)*Rg*T(i1,i2);
 END #For
END #For

# Reaction constant
FOR i1 := 0 TO L DO
 FOR i2 := 0 TO R DO
  k(i1,i2) = A*EXP(-E/Rg/T(i1,i2));
 END #For
END #For

# Reaction rate
FOR i1 := 0 TO L DO
 FOR i2 := 0 TO R DO
  Rate(i1,i2) = k(i1,i2)*P(1,i1,i2)*P(2,i1,i2);
      END #For
     END #For

     # Coolant energy balance
     rhoc*Vc*Cpc*$Tc = Fc*Cpc*(Tcin-Tc) + Q;

     # Heat transfer relationship
     Q = OverallU*Area*INTEGRAL(IntNo1:=0:L;T(IntNo1,R)-Tc);

     # Feed conditions
     Pfeed() = Cfeed()*Rg*Tfeed;
END # Model CStudy1

PROCESS Sim
  UNIT
    CS101 AS CStudy1

SET
  WITHIN CS101 DO
    L     := 4; # m
    R     := 0.025; # m
    rhob     := 1300; # kg/m3
    rhof     := 1.293; # kg/m3
    Cpf     := 992; # J/kg.K
    Dz     := 0.01; # m2/s
    Dr     := 0.0001; # m2/s
    kz     := 0.5; # W/m.K
    kr     := 0.05; # W/m.K
    BedVoid     := 0.25; #
```

```
  deltaH    := -1.20E+06; # J/mol
  A     := 1.15E-02; # mol/kg.s.Pa2
  E     := 113370; # J/mol
  hw     := 96; # W/m2.K
  Rg     := 8.314; # J/mol.K
  rhoc     := 2000; # kg/m3
  Cpc     := 123900; # J/kg.K
  Vc     := 0.0100; # m3
  Area     := 0.628; # m2
  OverallU     := 9.6; # W/m2.K
  NoComp     := 2; #
  Nu     := [-1,-1]; #

# Mathematical Solution Methods
  ax := [BFDM, 2, 40];
    rad := [OCFEM, 3, 5];
  END # Within

ASSIGN
  WITHIN CS101 DO
    Pfeed     := [1100,21100]; # Pa
    Tfeed     := 550; # K
    u     := 1; # m/s
    Fc     := 0.05; # kg/s
    Tcin     := 550; # K
  END # Within

INITIAL
  WITHIN CS101 DO
  FOR i1 := 0|+ TO L|- DO
   FOR i2 := 0|+ TO R|- DO
    C(,i1,i2)      = 0; # kg/mol;
   END #For
  END #For
  FOR i1 := 0|+ TO L|- DO
   FOR i2 := 0|+ TO R|- DO
    T(i1,i2)      = 550; # K;
   END #For
  END #For
    Tc      = 550; # K
  END # Within


SOLUTIONPARAMETERS
  ReportingInterval := 0.1;
  BlockDecomposition := ON;

SCHEDULE
  CONTINUE FOR 10

END
```

V.4. Translated Cooling Reactor Simulink Input File (Case Study 3)

```
Model {
  Name                  "cstudy3"
  Version               3.00
  SimParamPage          "Solver"
  SampleTimeColors      off
  InvariantConstants    off
  WideVectorLines   off
  ShowLineWidths    off
  ShowPortDataTypes     off
  StartTime             "0.0"
  StopTime              "7200"
  SolverMode            "Auto"
  Solver                "ode45"
  RelTol                "1e-3"
  AbsTol                "auto"
  Refine                "1"
  MaxStep               "auto"
  InitialStep           "auto"
  FixedStep             "auto"
  MaxOrder              5
  OutputOption          "RefineOutputTimes"
  OutputTimes           "[]"
  LoadExternalInput     off
  ExternalInput         "[t, u]"
  SaveTime              on
  TimeSaveName          "tout"
  SaveState             off
  StateSaveName         "xout"
  SaveOutput            on
  OutputSaveName    "yout"
  LoadInitialState      off
  InitialState          "xInitial"
  SaveFinalState    off
  FinalStateName    "xFinal"
  SaveFormat            "Matrix"
  LimitMaxRows          off
  MaxRows               "1000"
  Decimation            "1"
  AlgebraicLoopMsg      "warning"
  MinStepSizeMsg    "warning"
  UnconnectedInputMsg       "warning"
  UnconnectedOutputMsg      "warning"
  UnconnectedLineMsg        "warning"
  InheritedTsInSrcMsg       "warning"
  IntegerOverflowMsg        "warning"
  UnnecessaryDatatypeConvMsg "none"
  Int32ToFloatConvMsg       "warning"
  SignalLabelMismatchMsg    "none"
  ConsistencyChecking       "off"
  ZeroCross             on
  SimulationMode    "normal"
  BlockDataTips             on
  BlockParametersDataTip    on
  BlockAttributesDataTip    off
  BlockPortWidthsDataTip    off
  BlockDescriptionStringDataTip     off
  BlockMaskParametersDataTip off
  ToolBar               on
  StatusBar             on
```

```
  BrowserShowLibraryLinks off
  BrowserLookUnderMasks   off
  OptimizeBlockIOStorage  on
  BufferReuse             on
  BooleanDataType   off
  RTWSystemTargetFile     "grt.tlc"
  RTWInlineParameters     off
  RTWRetainRTWFile        off
  RTWTemplateMakefile     "grt_default_tmf"
  RTWMakeCommand    "make_rtw"
  RTWGenerateCodeOnly     off
  ExtModeMexFile    "ext_comm"
  ExtModeBatchMode        off
  ExtModeTrigType   "manual"
  ExtModeTrigMode   "oneshot"
  ExtModeTrigPort   "1"
  ExtModeTrigElement      "any"
  ExtModeTrigDuration     1000
  ExtModeTrigHoldOff      0
  ExtModeTrigDelay        0
  ExtModeTrigDirection    "rising"
  ExtModeTrigLevel        0
  ExtModeArchiveMode      "off"
  ExtModeAutoIncOneShot   off
  ExtModeIncDirWhenArm    off
  ExtModeAddSuffixToVar   off
  ExtModeWriteAllDataToWs off
  ExtModeArmWhenConnect   off
  Created           "Thu Jul 20 16:35:30 2000"
  UpdateHistory           "UpdateHistoryNever"
  ModifiedByFormat        "%<Auto>"
  LastModifiedBy    "planteng"
  ModifiedDateFormat      "%<Auto>"
  LastModifiedDate        "Thu Nov 02 13:33:15 2000"
  ModelVersionFormat      "1.%<AutoIncrement:13>"
  ConfigurationManager    "none"
  BlockDefaults {
    Orientation             "right"
    ForegroundColor         "black"
    BackgroundColor         "white"
    DropShadow              off
    NamePlacement     "normal"
    FontName                "Helvetica"
    FontSize                10
    FontWeight              "normal"
    FontAngle               "normal"
    ShowName                on
  }
  AnnotationDefaults {
    HorizontalAlignment     "center"
    VerticalAlignment       "middle"
    ForegroundColor         "black"
    BackgroundColor         "white"
    DropShadow              off
    FontName                "Helvetica"
    FontSize                10
    FontWeight              "normal"
    FontAngle               "normal"
  }
  LineDefaults {
    FontName                "Helvetica"
```

```
  FontSize                9
  FontWeight              "normal"
  FontAngle               "normal"
}
System {
  Name                  "cstudy3"
  Location              [450, 302, 1191, 626]
  Open                  on
  ModelBrowserVisibility  off
  ModelBrowserWidth       200
  ScreenColor             "automatic"
  PaperOrientation        "landscape"
  PaperPositionMode       "auto"
  PaperType               "a4letter"
  PaperUnits              "inches"
  ZoomFactor              "100"
  AutoZoom                on
  ReportName              "simulink-default.rpt"
  Block {
    BlockType               Constant
    Name                  "Area"
    Description           "Heat Transfer Area"
    Position                [250, 20, 280, 50]
    Value                 "1.5"
  }
  Block {
    BlockType               Constant
    Name                  "Cvb"
    Description           "Bulk Thermal Heat Capacity"
    Position                [100, 20, 130, 50]
    Value                 "4000"
  }
  Block {
    BlockType               Constant
    Name                  "Cvc"
    Description           "Coolant Thermal Heat Capacity"
    Position                [150, 20, 180, 50]
    Value                 "4000"
  }
  Block {
    BlockType               Integrator
    Name                  "Integrator1"
    Ports                 [1, 1, 0, 0, 0]
    Position                [100, 70, 130, 100]
    ExternalReset           "none"
    InitialConditionSource  "internal"
    InitialCondition        "80"
    LimitOutput           off
    UpperSaturationLimit    "inf"
    LowerSaturationLimit    "-inf"
    ShowSaturationPort      off
    ShowStatePort           off
    AbsoluteTolerance       "auto"
  }
  Block {
    BlockType               Integrator
    Name                  "Integrator2"
    Ports                 [1, 1, 0, 0, 0]
    Position                [100, 120, 130, 150]
    ExternalReset           "none"
    InitialConditionSource  "internal"
```

```
  InitialCondition        "20"
  LimitOutput       off
  UpperSaturationLimit    "inf"
  LowerSaturationLimit    "-inf"
  ShowSaturationPort      off
  ShowStatePort           off
  AbsoluteTolerance       "auto"
}
Block {
  BlockType               Product
  Name                "Product1"
  Ports               [2, 1, 0, 0, 0]
  Position                [150, 72, 180, 103]
  Inputs                  "2"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType               Product
  Name                "Product10"
  Ports               [2, 1, 0, 0, 0]
  Position                [350, 122, 380, 153]
  Inputs                  "2"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType               Product
  Name                "Product11"
  Ports               [2, 1, 0, 0, 0]
  Position                [400, 122, 430, 153]
  Inputs                  "2"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType               Product
  Name                "Product12"
  Ports               [2, 1, 0, 0, 0]
  Position                [450, 122, 480, 153]
  Inputs                  "2"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType               Product
  Name                "Product13"
  Ports               [2, 1, 0, 0, 0]
  Position                [500, 122, 530, 153]
  Inputs                  "*/"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType               Product
  Name                "Product2"
  Ports               [2, 1, 0, 0, 0]
  Position                [200, 72, 230, 103]
  Inputs                  "2"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType               Product
  Name                "Product3"
  Ports               [2, 1, 0, 0, 0]
  Position                [250, 72, 280, 103]
```

```
    Inputs                          "2"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Product
    Name                    "Product4"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [300, 72, 330, 103]
    Inputs                          "2"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Product
    Name                    "Product5"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [350, 72, 380, 103]
    Inputs                          "*/"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Product
    Name                    "Product6"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [150, 122, 180, 153]
    Inputs                          "2"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Product
    Name                    "Product7"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [200, 122, 230, 153]
    Inputs                          "2"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Product
    Name                    "Product8"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [250, 122, 280, 153]
    Inputs                          "2"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Product
    Name                    "Product9"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [300, 122, 330, 153]
    Inputs                          "2"
    SaturateOnIntegerOverflow       on
  }
  Block {
    BlockType                       Sum
    Name                    "Sum1"
    Ports                   [2, 1, 0, 0, 0]
    Position                        [400, 70, 430, 100]
    ShowName                        off
    IconShape                       "round"
    Inputs                          "|+-"
    SaturateOnIntegerOverflow       on
  }
```

```
Block {
  BlockType                  Sum
  Name                "Sum2"
  Ports               [2, 1, 0, 0, 0]
  Position                   [550, 120, 580, 150]
  ShowName                   off
  IconShape                  "round"
  Inputs                     "|+-"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                  Sum
  Name                "Sum3"
  Ports               [2, 1, 0, 0, 0]
  Position                   [600, 120, 630, 150]
  ShowName                   off
  IconShape                  "round"
  Inputs                     "|++"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                  Sum
  Name                "Sum4"
  Ports               [2, 1, 0, 0, 0]
  Position                   [650, 120, 680, 150]
  ShowName                   off
  IconShape                  "round"
  Inputs                     "|+-"
  SaturateOnIntegerOverflow     on
}
Block {
  BlockType                  Scope
  Name                "T"
  Ports               [1, 0, 0, 0, 0]
  Position                   [30, 104, 60, 136]
  Orientation         "left"
  Floating                   off
  Location                   [89, 239, 413, 478]
  Open                on
  NumInputPorts              "1"
  TickLabels          "OneTimeTick"
  ZoomMode                   "on"
  List {
  ListType            AxesTitles
  axes1               "%<SignalLabel>"
  }
  Grid                "on"
  TimeRange                  "auto"
  YMin                "-5"
  YMax                "5"
  SaveToWorkspace            off
  SaveName                   "ScopeData"
  DataFormat          "StructureWithTime"
  LimitMaxRows               on
  MaxRows                    "5000"
  Decimation          "1"
  SampleInput         off
  SampleTime          "0"
}
Block {
  BlockType                  Scope
```

```
    Name                "Tc"
    Ports               [1, 0, 0, 0, 0]
    Position            [30, 164, 60, 196]
    Orientation         "left"
    Floating            off
    Location            [89, 516, 413, 755]
    Open                on
    NumInputPorts       "1"
    TickLabels          "OneTimeTick"
    ZoomMode            "on"
    List {
    ListType            AxesTitles
    axes1               "%<SignalLabel>"
    }
    Grid                "on"
    TimeRange           "auto"
    YMin                "-5"
    YMax                "5"
    SaveToWorkspace     off
    SaveName            "ScopeData"
    DataFormat          "StructureWithTime"
    LimitMaxRows        on
    MaxRows             "5000"
    Decimation          "1"
    SampleInput         off
    SampleTime          "0"
  }
  Block {
    BlockType           Constant
    Name                "Tcin"
    Description         "Coolant Inlet Temperature"
    Position            [350, 20, 380, 50]
    Value               "20"
  }
  Block {
    BlockType           Constant
    Name                "U"
    Description         "Overall Heat Transfer Coefficient"
    Position            [300, 20, 330, 50]
    Value               "1"
  }
  Block {
    BlockType           Constant
    Name                "Vol"
    Description         "Bulk Volume"
    Position            [50, 20, 80, 50]
    Value               "1"
  }
  Block {
    BlockType           Constant
    Name                "Volc"
    Description         "Coolant Volume"
    Position            [200, 20, 230, 50]
    Value               "0.025"
  }
  Block {
    BlockType           Constant
    Name                "q"
    Description         "Coolant Flowrate"
    Position            [400, 20, 430, 50]
    Value               "0.1"
```

```
}
Block {
  BlockType              Constant
  Name                   "rhob"
  Description            "Bulk Density"
  Position               [500, 20, 530, 50]
  Value                  "1000"
}
Block {
  BlockType              Constant
  Name                   "rhoc"
  Description            "Coolant Density"
  Position               [450, 20, 480, 50]
  Value                  "1000"
}
Line {
  SrcBlock               "Cvb"
  SrcPort                1
  DstBlock               "Product1"
  DstPort                1
}
Line {
  SrcBlock               "rhob"
  SrcPort                1
  Points                 [0, 75; -400, 0]
  DstBlock               "Product1"
  DstPort                2
}
Line {
  SrcBlock               "Vol"
  SrcPort                1
  Points                 [0, 60; 85, 0; 0, -15]
  DstBlock               "Product2"
  DstPort                1
}
Line {
  SrcBlock               "Product1"
  SrcPort                1
  DstBlock               "Product2"
  DstPort                2
}
Line {
  SrcBlock               "Product2"
  SrcPort                1
  Points                 [0, 15; 100, 0]
  DstBlock               "Product5"
  DstPort                2
}
Line {
  SrcBlock               "Product3"
  SrcPort                1
  DstBlock               "Product4"
  DstPort                1
}
Line {
  SrcBlock               "Sum1"
  SrcPort                1
  Points                 [0, 10]
  DstBlock               "Product4"
  DstPort                2
}
```

```
Line {
  SrcBlock                "Product4"
  SrcPort                 1
  DstBlock                "Product5"
  DstPort                 1
}
Line {
  SrcBlock                "Product5"
  SrcPort                 1
  Points                  [0, 25; -300, 0]
  DstBlock                "Integrator1"
  DstPort                 1
}
Line {
  SrcBlock                "Product13"
  SrcPort                 1
  Points                  [0, 35; -450, 0]
  DstBlock                "Integrator2"
  DstPort                 1
}
Line {
  SrcBlock                "U"
  SrcPort                 1
  Points                  [0, 0]
  Branch {
  Points                  [0, 45]
  DstBlock                "Product3"
  DstPort                 1
  }
  Branch {
  Points                  [0, 20; -175, 0; 0, 75]
  DstBlock                "Product6"
  DstPort                 1
  }
}
Line {
  SrcBlock                "Area"
  SrcPort                 1
  Points                  [0, 0]
  Branch {
  Points                  [5, 0; 0, 60]
  DstBlock                "Product3"
  DstPort                 2
  }
  Branch {
  Points                  [0, 15; -160, 0; 0, 95]
  DstBlock                "Product6"
  DstPort                 2
  }
}
Line {
  SrcBlock                "Product6"
  SrcPort                 1
  DstBlock                "Product7"
  DstPort                 1
}
Line {
  SrcBlock                "Sum2"
  SrcPort                 1
  Points                  [0, 30; -400, 0]
  DstBlock                "Product7"
```

```
    DstPort                 2
  }
  Line {
    SrcBlock                "Integrator1"
    SrcPort                 1
    Points                  [5, 0]
    Branch {
    Points                  [395, 0]
    DstBlock            "Sum2"
    DstPort                 1
    }
    Branch {
    Points                  [0, 55; 50, 0; 0, 15]
    Branch {
      Points              [225, 0]
      DstBlock            "Sum1"
      DstPort             2
    }
    Branch {
      Points              [0, -35]
      DstBlock            "T"
      DstPort             1
    }
    }
  }
  Line {
    SrcBlock                "Integrator2"
    SrcPort                 1
    Points                  [20, 0]
    Branch {
    Points                  [0, 20; 410, 0]
    DstBlock            "Sum2"
    DstPort                 2
    }
    Branch {
    Points                  [0, -15; 510, 0]
    DstBlock            "Sum4"
    DstPort                 2
    }
    Branch {
    Points                  [0, 25; 180, 0]
    Branch {
      Points              [0, -5; 120, 0; 0, -70]
      DstBlock            "Sum1"
      DstPort             1
    }
    Branch {
      Points              [0, 20]
      DstBlock            "Tc"
      DstPort             1
    }
    }
  }
  Line {
    SrcBlock                "Product7"
    SrcPort                 1
    Points                  [0, 50; 350, 0]
    DstBlock                "Sum3"
    DstPort                 1
  }
  Line {
```

```
    SrcBlock                "q"
    SrcPort                 1
    Points                  [0, 85; -200, 0]
    DstBlock                "Product8"
    DstPort                 1
  }
  Line {
    SrcBlock                "Product8"
    SrcPort                 1
    DstBlock                "Product9"
    DstPort                 1
  }
  Line {
    SrcBlock                "Product9"
    SrcPort                 1
    DstBlock                "Product10"
    DstPort                 1
  }
  Line {
    SrcBlock                "Tcin"
    SrcPort                 1
    Points                  [0, -10; 250, 0]
    DstBlock                "Sum4"
    DstPort                 1
  }
  Line {
    SrcBlock                "Sum4"
    SrcPort                 1
    Points                  [0, 45; -350, 0]
    DstBlock                "Product10"
    DstPort                 2
  }
  Line {
    SrcBlock                "Product10"
    SrcPort                 1
    Points                  [0, 60; 230, 0]
    DstBlock                "Sum3"
    DstPort                 2
  }
  Line {
    SrcBlock                "Sum3"
    SrcPort                 1
    Points                  [0, -5]
    DstBlock                "Product13"
    DstPort                 1
  }
  Line {
    SrcBlock                "rhoc"
    SrcPort                 1
    Points                  [0, 0]
    Branch {
    Points                      [10, 0; 0, 140; -260, 0]
    DstBlock                "Product8"
    DstPort                     2
    }
    Branch {
    Points                      [0, 95]
    DstBlock                "Product11"
    DstPort                     1
    }
  }
```

```
Line {
  SrcBlock                "Cvc"
  SrcPort                 1
  Points                  [0, 0]
  Branch {
  Points                  [0, 15; 100, 0]
  DstBlock              "Product9"
  DstPort                 2
  }
  Branch {
  Points                  [0, 110]
  DstBlock              "Product11"
  DstPort                 2
  }
}
Line {
  SrcBlock                "Product11"
  SrcPort                 1
  DstBlock                "Product12"
  DstPort                 1
}
Line {
  SrcBlock                "Volc"
  SrcPort                 1
  Points                  [200, 0]
  DstBlock                "Product12"
  DstPort                 2
}
Line {
  SrcBlock                "Product12"
  SrcPort                 1
  DstBlock                "Product13"
  DstPort                 2
```

```
  }
  }
}
```

V.5. Translated Cooling Reactor gPROMS Input File (Case Study 3)

```
#*****************************************************
# Problem Title : Cooling Reactor
# Created : Thu Nov 23 15:23:09 2000
# By IMIPS v0.03 gPROMS Translator
#*****************************************************

DECLARE
  TYPE
    Type1   =   1   : -1E6 : 1E6   UNIT = "" #
    Type2   =   1   : -1E6 : 1E6   UNIT = "" #
END # Declare

MODEL CStudy3
  PARAMETER
    Vol          AS REAL # Bulk Volume
    Cvb          AS REAL # Bulk Thermal Heat Capacity
    Cvc          AS REAL # Coolant Thermal Heat Capacity
    Volc          AS REAL # Coolant Volume
    Area          AS REAL # Heat Transfer Area
    U          AS REAL # Overall Heat Transfer Coefficient
    Tcin          AS REAL # Coolant Inlet Temperature
    q          AS REAL # Coolant Flowrate
    rhoc          AS REAL # Coolant Density
    rhob          AS REAL # Bulk Density


  VARIABLE
    Tc          AS Type1 #
    T          AS Type2 #

  EQUATION
    #
    $T*rhob*Cvb*Vol = U*Area*(Tc-T);

    #
    $Tc*rhoc*Cvc*Volc = U*Area*(T-Tc)+q*rhoc*Cvc*(Tcin-Tc);
END # Model CStudy3

PROCESS Sim
  UNIT
    CS3 AS CStudy3


  SET
    WITHIN CS3 DO
      Vol      := 0.28; # m3
      Cvb      := 4000; # kJ/m3.K
      Cvc      := 6000; # kJ/m3.K
      Volc     := 0.049; # m3
      Area     := 1.88; # m2
      U     := 1200; # kW/m2.K
      Tcin     := 5; # C
      q     := 8.333e-3; # m3/s
      rhoc     := 800; # kg/m3
      rhob     := 1000; # kg/m3
    END # Within

INITIAL
  WITHIN CS3 DO
```

```
      Tc      = 5; #
      T       = 120; #
    END # Within


  SOLUTIONPARAMETERS
    ReportingInterval := 200;
    BlockDecomposition := ON;

  SCHEDULE
    CONTINUE FOR 7200

END
```

V.6. Translated CSTR, Van de Vusse reaction gPROMS Input File (Case Study 4)

```
#*************************************************************
# Problem Title : Liquid Phase CSTR
# Created : Fri Nov 03 09:44:10 2000
# By IMIPS v0.03 gPROMS Translator
#*************************************************************

DECLARE
  TYPE
    Type1   =  1   : -1E6 : 1E6   UNIT = "" #
    Type2   =  1   : -1E6 : 1E6   UNIT = "" #
    Type3   =  1   : -1E6 : 1E6   UNIT = "" #
    Type4   =  1   : -1E6 : 1E6   UNIT = "" #
    Type5   =  1   : -1E6 : 1E6   UNIT = "" #
    Type6   =  1   : -1E6 : 1E6   UNIT = "" #
    Type7   =  1   : -1E6 : 1E6   UNIT = "" #
    Type8   =  1   : -1E6 : 1E6   UNIT = "" #
    Type9   =  1   : -1E6 : 1E6   UNIT = "" #
    Type10   =  1   : -1E6 : 1E6   UNIT = "" #
END # Declare

MODEL CStudy4
  PARAMETER
    NoComp          AS INTEGER # Number of Components
    NoReac          AS INTEGER # Number of Reactions
    ValveConstant          AS REAL # Valve Constant
    Diameter          AS REAL # Tank Diameter
    Hp          AS REAL # Outlet Pipe Height From Base
    Density          AS ARRAY(Nocomp) OF REAL # Component Densities
    ReactionConstant          AS ARRAY(NoReac) OF REAL # Reaction Rates
    Order          AS ARRAY(NoComp,NoReac) OF REAL # Order
    NU          AS ARRAY(NoComp,NoReac) OF REAL # Stoichiometric
coefficients
    Pi          AS REAL # Pi


  VARIABLE
    Fin          AS Type1 #
    Xin          AS ARRAY(NoComp) OF Type2 #
    Fout          AS Type3 #
    Xout          AS ARRAY(NoComp) OF Type4 #
    Holdup          AS ARRAY(NoComp) OF Type5 #
    Cout          AS ARRAY(NoComp) OF Type6 #
    TotalHoldup          AS Type7 #
    TotalVolume          AS Type8 #
    Height          AS Type9 #
    Rate          AS ARRAY(NoReac) OF Type10 #


  EQUATION
    # Mass Balance
    FOR i1 := 1 TO NoComp DO
     $Holdup(i1)=Fin*Xin(i1)-
Fout*Xout(i1)+TotalVolume*SIGMA(NU(i1,)*Rate);
    END #For

    # Reaction Rates
    FOR i1 := 1 TO NoReac DO
```

```
 Rate(i1)=ReactionConstant(i1)*PRODUCT(Cout^Order(,i1));
END

# Total Volume
TotalVolume=SIGMA(Holdup()/Density());

# Total Holdup
TotalHoldup=SIGMA(Holdup());

# Molar Fractions
Holdup()=Xout()*TotalHoldup;

# Molar Concentrations
Holdup()=Cout()*TotalVolume;

    # Calculation of liquid level
    TotalVolume=Pi*Diameter^2*Height/4;

    # Calculation of flowrate out
    IF Height > Hp  THEN
    Fout = ValveConstant*(Height-Hp) ;
    ELSE
    Fout = 0;
    END # If

END # Model CStudy4

PROCESS Sim
  UNIT
    CS4 AS CStudy4

  SET
  WITHIN CS4 DO
    NoComp     := 4; #
    NoReac     := 3; #
    ValveConstant     := 0.3; #
    Diameter     := 3; # m
    Hp     := 2; # m
    Density     := [17.48,17.15,10.24,55.56]; # kg/m3
    ReactionConstant     := [8E-3,1.3E-2,1E-2]; # m3/kmol.s
    Order     := [1,0,1,0,1,0,0,0,0,0,0,0]; #
    NU     := [-1,0,-2,1,-1,0,0,1,0,0,0,1]; #
    Pi     := 3.1415926; #
  END # Within

ASSIGN
  WITHIN CS4 DO
    Fin     := 1; #
    Xin     := [0.5,0.1,0.3,0.1]; #
  END # Within

INITIAL
  WITHIN CS4 DO
    Xout(1)      = 0.5; #
    Xout(2)      = 0.1; #
    Xout(3)      = 0.3; #
    TotalHoldup      = 70.69; #
  END # Within


SOLUTIONPARAMETERS
```

```
    ReportingInterval := 200;
    BlockDecomposition := ON;

  SCHEDULE
    CONTINUE FOR 7200

END
```