Computer Generation of Process Models

Jamie Alan BARBER

A thesis submitted for the degree of
Doctor of Philosophy of the University of London
and for the
Diploma of Membership of Imperial College

Centre for Process Systems Engineering
Imperial College of Science, Technology and Medicine
London SW7 2BY
www.ps.ic.ac.uk

March 2000

BIBL. LONDN. UNIV.

If you are first you are first. If you are second, you are nothing.

Bill Shankly (1913–1981)

Abstract

This thesis is concerned with the development of a software tool for the automatic generation of combined lumped and distributed parameter process models. It presents a rigorous theoretical approach to process modelling and defines a process representation and systematic model building procedure supporting the formulation of such models from a purely physical description of a given process system.

The process representation is based on the premise that all material undergoing processing is instantaneously in a stable thermodynamic state, implying that all material is present in well-defined, continuous thermodynamic phases. Thus a process system is represented as a set of interacting phases, contained within vessels, which may be linked together via connections.

For each phase the model building procedure involves the systematic formulation of mathematical statements describing the principles of conservation of mass, energy and momentum and also the construction of constitutive and closure relationships in an appropriate form. Additional equations arise from the fact that phases are constrained within vessels and inter-vessel connection relationships are generated from a knowledge of the process topology.

The methodology has been implemented in the computer program gMaGiC (*g*eneral *M*odel *G*eneration *C*ode) using a compiler-based software architecture and the validity of the approach is demonstrated by its application to several nontrivial case studies. These are chosen to demonstrate the success of gMaGiC in modelling systems exhibiting a variety of different physico-chemical behaviour including compressible and incompressible fluids, varying thermal characteristics (isothermal, adiabatic and finite heat transfer), laminar and turbulent flow regimes, single and multicomponent mixtures (reacting and inert) and operation under steady state and dynamic conditions.

Acknowledgements

I would like to thank Professors J. D. Perkins and R. W. H. Sargent for their help and supervision during my PhD and with the preparation of this thesis.

I would also especially like to thank Professor Stephen Richardson for much helpful discussion and his patient answering of my numerous questions on fluid mechanics throughout the course of this work, and also for his encouragement and advice when I found the going tough.

The financial support of the Engineering and Physical Sciences Research Council and both the Centre for Process Systems Engineering and the Department of Chemical Engineering and Chemical Technology (through I.C.O.N.) at Imperial College is also gratefully acknowledged.

Contents

Abstract **3**

Acknowledgements **4**

1 Introduction **12**

1.1 Current Status of Computer-Aided Model Generation 14

1.2 Computational Fluid Dynamics 20

1.3 Motivation and Objectives 23

1.4 Thesis Outline 23

2 Mathematical Modelling of Single and Multiphase Systems **25**

2.1 Single Phase Systems 25

2.2 Multiphase Systems 31

2.2.1 Local Instantaneous Formulation 31

2.2.2 Averaged Models 32

2.3 Initial and Boundary Conditions 37

2.3.1 Solid Walls 38

2.3.2 Open Fluid Interfaces 40

2.3.3 Symmetry Boundaries 42

2.3.4 Cyclic Boundaries 42

2.4 Simplified Models 43

2.4.1 Models of Reduced Dimensionality 43

2.4.2 Modelling Assumptions 54

2.5 Connections Between Simplified Models 54

2.5.1 Discontinuities in Mixing Characteristics Across Open Interfaces . 55

2.5.2 Model Dimensionality 56

2.6 Mathematical Classification of Model Equations 58

2.6.1 Differential Equations 58

2.6.2 Boundary Conditions 61

2.6.3 Differential Equation and Boundary Condition Domain Validity . . 63

2.7 Scope of Work 64

2.8 Summary 66

3 Process Representation **67**
3.1 Conceptual Framework 67
3.2 Process Modelling Language 70
3.2.1 Vessel Section 70
3.2.2 Reservoir Section 75
3.2.3 Connection Section 76
3.3 Process Modelling Language Example 77
3.4 Summary 77

4 Model Building Procedure **80**
4.1 Basic Strategy 80
4.2 Model Generation Algorithms 82
4.2.1 Vessel Submodels 82
4.2.2 Reservoir Submodels 105
4.2.3 Connection Submodel 105
4.3 Summary 108

5 Implementation **111**
5.1 Software Architecture 111
5.2 The Translator 113
5.2.1 The Scanner 113
5.2.2 The Parser 114
5.2.3 Semantic Checking 114
5.3 The Model Generation Engine 115
5.3.1 The Mathematical Symbol Table 115
5.4 The Code Generator 116
5.5 Program Operation 119
5.6 Summary 119

6 Modelling Examples **122**
6.1 Choking in Compressible Flow with Friction 123
6.1.1 Process Representation 124
6.1.2 Modelling and Validation 124
6.2 Vinyl Acetate Reactor 127
6.2.1 Process Representation 128
6.2.2 Modelling and Validation 128
6.3 Double Pipe Heat Exchanger 132
6.3.1 Process Representation 133
6.3.2 Modelling and Validation 139
6.4 Laminar Pipe Flow 145

6.4.1 Process Representation 145
6.4.2 Modelling and Validation 146
6.5 Turbulent Pipe Flow 151
6.5.1 Process Representation 151
6.5.2 Modelling and Validation 152
6.6 Summary 155

7 Conclusions **156**
7.1 Thesis Summary 156
7.2 Future Work 158

References **160**

A Nomenclature **168**
A.1 Mathematical Notation 168
A.1.1 Symbols 168
A.1.2 Greek Symbols 170
A.1.3 Subscripts/Superscripts 171
A.1.4 Miscellaneous 171
A.2 Notation used in gMaGiC 172
A.2.1 Variable Labelling, Typing and Referencing 172
A.2.2 Variable Types 172

B Averaging Operators **175**
B.1 Basic Definitions 175
B.2 Special Definitions for Multiphase Modelling 175

C Flex Input Specification **177**

D Bison Input Specification **180**

E Generated Models **186**
E.1 Choking in Compressible Flow with Friction 187
E.2 Vinyl Acetate Reactor 189
E.3 Double Pipe Heat Exchanger - Case 1 192
E.4 Double Pipe Heat Exchanger - Case 2 196
E.5 Double Pipe Heat Exchanger - Case 3 200
E.6 Double Pipe Heat Exchanger - Case 4 204
E.7 Laminar Pipe Flow 208
E.8 Turbulent Pipe Flow 211

F *gPROMS* Files For Examples of Chapter 6 **214**

List of Figures

1.1 Computer-Aided Process Modelling Tools 22

2.1 Averaged Model Formulation 34
2.2 Boundary Types in Internal Flows 39
2.3 An Open Fluid Interface between Two Flow Domains 41
2.4 Control Volume cut by two Cross-section Planes 46
2.5 Hypothetical Two Phase Vessel 51
2.6 Interface Between Domains of Different Dimensionality 57

3.1 Conceptual Representation of a Hypothetical Process System 69
3.2 Example Vessel Declarations 71
3.3 Example Geometry Attribute Declarations 72
3.4 Example Phase Attribute Declarations 73
3.5 Example Port Attribute Declarations 75
3.6 Example Assumption Declarations 75
3.7 Example Reservoir Declarations 76
3.8 Example Connection Declarations 77
3.9 Complete Language Example 78

4.1 Summary of Model Building Procedure 81

5.1 Software Architecture 112
5.2 Structure of Mathematical Symbol Table 117
5.3 Example of Equation Templates and Expression Trees 118
5.4 Example of gMaGiC Operation 120

6.1 Process Representation for Choking Compressible Flow in a Pipe 124
6.2 Language Declaration for Choking Flow 125
6.3 Pressure [Pa] versus Axial Position [m] 126
6.4 Mach Number [-] versus Axial Position [m] 127
6.5 Process Representation for Vinyl Acetate Reactor 128
6.6 Language Declaration for Vinyl Acetate Reactor 129
6.7 Temperature [K] versus Axial Position [m] 131
6.8 Pressure [Pa] versus Axial Position [m] 131
6.9 Heat Exchanger Dimensions 132

6.10 Process Representation for Double Pipe Heat Exchanger (cocurrent flow) 133
6.11 Language Declaration for Double Pipe Heat Exchanger - Case 1 135
6.12 Language Declaration for Double Pipe Heat Exchanger - Case 2 136
6.13 Language Declaration for Double Pipe Heat Exchanger - Case 3 137
6.14 Language Declaration for Double Pipe Heat Exchanger - Case 4 138
6.15 Temperature [K] versus Time [s] 141
6.16 Temperature [K] (at different times) versus Axial Position [m] 142
6.17 Temperature [K] versus Time [s] 142
6.18 Temperature [K] versus Axial Position [m] 143
6.19 Temperature [K] versus Axial Position [m] 144
6.20 Process Representation for Laminar Pipe Flow 146
6.21 Language Declaration for Laminar Pipe Flow 146
6.22 Axial Velocity [m/s] (at different axial positions) versus Radial Position [m] 148
6.23 Centreline Axial Velocity [m/s] versus Axial Position [m] 149
6.24 Radial Velocity [m/s] (at r = 0.8R) versus Axial Position [m] 150
6.25 Centreline Pressure [Pa] versus Axial Position [m] 150
6.26 Process Representation for Turbulent Pipe Flow 151
6.27 Language Declaration for Turbulent Pipe Flow 152
6.28 Fully Developed Axial Velocity [m/s] versus Radial Position [m] 155

List of Tables

2.1 Local Instantaneous Formulation 32
2.2 General Balance Equation 46
2.3 PDE Classifications 59

4.1 Lookup Table for Connections via Open Ports 109
4.2 Lookup Table for Connections via Closed Ports 110

6.1 Numerical Specifications for Choking Flow 125
6.2 Results from Choking Flow Simulation 126
6.3 Numerical Specifications for Vinyl Acetate Reactor 130
6.4 Model Alternatives for Double Pipe Heat Exchanger Example 133
6.5 Numerical Specifications for Concentric Pipe Heat Exchanger 139
6.6 Summary of Results from Four Cases of Double Pipe Heat Exchanger . . 144
6.7 Numerical Specifications for Laminar Pipe Flow 147
6.8 Numerical Specifications for Turbulent Pipe Flow 153

Chapter 1

Introduction

Mathematical modelling is an important technique in the analysis of scientific and engineering problems. It is the translation of a real world problem into an abstract mathematical form, requiring the identification of the significant features of the problem and the formulation of a mathematical structure that reproduces those features without irrelevant complexities [Sargent, 1983; Cho, 1997].

Mathematical models are widely used in nearly all chemical and process engineering activities. Examples include process design and synthesis, process control, fault diagnosis, process optimisation, operations planning and operator training [Stephanopoulos *et al.*, 1987; Pantelides and Barton, 1993; Marquardt, 1996; Perkins *et al.*, 1996]. However, it has become clear that the formulation of consistent models for these applications is a time-consuming, difficult and error-prone task [Perkins *et al.*, 1996] and often the cost of model development represents a significant part of the overall budget [Perkins and Barton, 1987]. To compound this situation, ever increasing demands on process plants to meet improved performance targets and more stringent environmental and safety legislation, together with the need for flexible and just-in-time production, necessitate the use of more sophisticated models incorporating greater detail and refinement. These facts, coupled with the widespread availability of increasingly powerful computing resources, have provided the motivation for the development of a number of software tools designed to aid process modelling.

The early flowsheeting packages allowed steady state process models to be constructed from pre-defined models of individual units in a modular fashion. This type of modelling tool is now common in industry and a large number of packages have been developed to a commercial standard (for example ASPEN PLUS [AspenTech, 1988]).

More recently there has been a trend towards developing aids for dynamic modelling to describe the transient behaviour of a given process system throughout its whole range of operation from start-up to shut-down [Basta, 1997; Fell, 1997; Sowa, 1997]. Typically an equation-oriented approach is used in which a model is essentially provided as a set of equations in a high-level declarative *modelling language* for use and solution in a process *modelling environment.* A feature of this approach is that the declarative knowledge about a process system (the model equations, assumptions *etc.*) is largely de-

coupled from the procedural knowledge (*i.e.* the algorithms for solution of the equations, optimisation routines *etc.*) and the user is saved from the burden of having to understand and code potentially complex solution algorithms.

Modelling languages are software tools designed to facilitate the structuring and documentation of process models. They are often based on object-oriented programming principles with process systems being represented as a collection of *objects*, each of which is an *instance* of a specific *class* of structured knowledge in a hierarchy of classes. This representation supports concepts such as inheritance, allowing new models to be described as combinations and refinements of existing ones and aims to promote model re-use to reduce development time, cost and effort. Representation of process systems in such a manner has been termed hierarchical submodel decomposition [Barton, 1992]. Examples of modelling languages include MODEL.LA [Stephanopoulos *et al.*, 1990], OMOLA [Mattsson and Andersson, 1992] and the language definition VEDA [Bogusch and Marquardt, 1995].

A *modelling environment* is a computer tool which supports the model-based process engineering applications mentioned above and generally includes facilities for simulation, optimisation and physical property prediction. It allows the high-level definition of a process model and may involve a specialised modelling language, for example DESIGN-KIT [Stephanopoulos *et al.*, 1987] uses the modelling language MODEL.LA, OMSIM and OMOLA are integrated in a control system design environment [Andersson, 1993] and DIVA [Kröner *et al.*, 1990] has been used both with an implementation of VEDA and in the PROMOT/DIVA [Tränkle *et al.*, 1997] environment for the modelling of distillation processes. Alternatively the language may be an integral part of the environment, as in the case in SPEEDUP [Perkins and Sargent, 1982], ASCEND [Pièla *et al.*, 1991] and *gPROMS* [Barton and Pantelides, 1994; Oh and Pantelides, 1996].

Other computer tools have been developed drawing on the same ideas as those which form the basis of modelling languages and modelling environments. For example MODKIT [Bogusch *et al.*, 1996] is described as a 'model development framework' but essentially comprises a modelling language with interfaces to the SPEEDUP and *gPROMS* modelling environments.

Most of the current state-of-the-art modelling languages and modelling environments make use of object-oriented methodology for model representation and include language structures and numerical methods for modelling combined continuous/discrete behaviour and mechanisms for the modelling of operating procedures. However most support process modelling only in terms of ordinary differential-algebraic equations and thus are limited to macroscopic (or lumped parameter) modelling. At present *gPROMS* is the only commercial process modelling environment with capabilities for distributed parameter modelling. In the current version this is limited to problems involving rectangular domains for which numerical solution is achieved using the method of lines.

The principal limitation of modelling languages and modelling environments is

that they assume the availability of a suitable mathematical model for the particular process system of interest. The formulation of appropriate and physically consistent mathematical descriptions is therefore the responsibility of the user. To date there has been little work on the development of computer aids for generating models based on the underlying physico-chemical phenomena. At present published work is limited to those few prototype tools reviewed below.

1.1 Current Status of Computer-Aided Model Generation

MODELLER [Preisig *et al.*, 1989; Preisig *et al.*, 1990; Preisig, 1995; Preisig, 1996; Westerweele *et al.*, 1999] was the first computer-aided model generation tool to be reported. It was initially developed for the automatic generation of lumped parameter dynamic models of life-support systems (systems closed in space which contain all the ingredients necessary for survival) but is equally well suited to the modelling of chemical process systems. In MODELLER a process is represented in terms of two main building blocks: *systems* and *connections*. A *system* is used to represent anything with a holdup and is defined as consisting of a single homogeneous or pseudo-homogeneous phase. A *species set* and a *reaction set* are defined within each system to deal with chemical components and reactions. The concept of a system is very close to that used in axiomatic thermodynamics and in MODELLER its mathematical description is based on macroscopic statements of the conservation principles. A *connection* is defined as an idealised topological link between two systems and contains no holdup. The transfer of extensive quantities between two systems is therefore described by pseudo-steady state *transfer laws*, usually defined in terms of the states of the two systems. Using a network of systems and connections a *physical topology*, similar to a flowsheet, of the process system can be constructed. To enable efficient definition of complex physical topology MODELLER allows the hierarchical grouping of systems and connections and supports the concept of inheritance for defining new models from existing ones. The core of the MODELLER implementation (the latest version written in Component Pascal) is the generation of an appropriate set of macroscopic balance equations in a textual output, although the authors indicate that this output can be transformed to interface with or serve as input to existing modelling languages and environments. Constitutive and closure laws for these equations (*i.e.* transfer laws, reaction stoichiometry and kinetics, physical properties, variable transformations *etc.*) are supplied by interfaces with existing knowledge bases, although it is not clear whether these interfaces have yet been implemented. Future objectives in the MODELLER project include adding facilities for modelling controllers and the generation of distributed parameter models. In addition the authors also intend interfacing MODELLER with SyMoM [Dekkers *et al.*, 1999], a computer package developed by the same group for the analysis and symbolic manipulation of dynamic models of process systems

obtained by first principles modelling.

PROFIT [Telnes, 1992] is a prototype package for the automatic generation of lumped parameter process models from first principles. In PROFIT each process unit is represented as a set of phases contained within volumes. Units are defined using a combination of textual and menu-driven inputs. For each volume the balance equations and closure laws are constructed from simple pre-defined mathematical terms, each representing a physical phenomenon, using a rule-based inference engine. Volumes are connected together via surfaces, which may be open, closed or permeable to selective phases or components within phases. One of the main features of the work is the inclusion of simple geometrical information which allows PROFIT to be used for the modelling of entire plants using simple geometrical shapes (for example representing a tank as a cylinder) and more detailed modelling of single units using combinations of shapes. This allows spatial variation of properties within a process unit to be described using a control volume or cell-type model, although the manual specification of each control volume can be a rather labour-intensive approach.

An unnamed prototype package for the automatic generation of lumped parameter process models has been developed at Imperial College by Vázquez-Román [1992], and later extended by Perkins *et al.* [1996] and Cho [1997], using a process representation close to that used in MODELLER. The conceptual approach is based on the premise that all material undergoing processing is instantaneously in a stable thermodynamic state, implying that all material is present in well defined thermodynamic phases. In this way a process system is described as a set of interacting *phases* contained within *vessels*, which may be linked together via *connections*. Inter-phase and inter-vessel mass and energy interactions are described by physico-chemical *transfer laws*. The concept of a *reservoir* is used to define entities whose states are unchanged by transfers (for example the atmosphere). In the latest version of the prototype the additional concepts of *aggregations* and *ports* are included to describe the phase distribution within a vessel and discontinuous flow phenomena arising from the position of a connection to a vessel. The latest version of the prototype (implemented in C) uses a textual specification of the process representation and automatically generates macroscopic mass and energy balance equations and produces transfer laws (taken from a pre-defined library) and closure relations in an appropriate form. A key feature of this work is the generation of models exhibiting combined continuous/discrete behaviour and the inclusion of facilities for hierarchical submodel decomposition aimed at promoting model reuse and reducing development time, cost and effort. The generated models are written in the format of the *gPROMS* modelling language for direct use in the *gPROMS* modelling environment. This previous work at Imperial College provided, in part, some of the motivation and fundamental ideas upon which that presented in this thesis is based (see section 1.3). A modelling support system, KBMoSS, has also been developed by Vázquez-Román *et al.* [1996] at the University of

Edinburgh incorporating the first version of the prototype package.

A formal process modelling methodology has been developed by Drengstig *et al.* [1997] based on a topological and a phenomenological process abstraction using a graphical representation scheme. The topological abstraction is used to decompose a process system into a network of *control volumes* and connecting *boundaries*, which are termed *elementary* if they correspond to a single phase and its boundaries or *composite* if composed of a number of elementary entities (for example a process unit). This concept is close to that used in MODELLER and by Vázquez-Román [1992], Perkins *et al.* [1996] and Cho [1997] (see above). The phenomenological abstraction is used to represent the physico-chemical behaviour of the process system in terms of three general process characteristics; transport, reaction/generation and accumulation of mass and energy. These characteristics are represented by graphical symbols which are connected together based on the topological abstraction and the user's understanding of the process. The authors intend to employ this methodology in the development of a computer program for the automatic generation of lumped parameter process models.

Lakner *et al.* [1999] have developed an assumption-driven case-specific model editor for the automatic generation and transformation of lumped parameter process models. The model editor, implemented in PROLOG, is based on a systematic modelling procedure comprising two main steps: specification of an initial detailed model followed by subsequent refinement of this model using simplifying assumptions. The first step is carried out by the *model builder*, which is implemented in the form of an interactive interface asking the user for specification of the balance (control) volumes, conserved extensive quantities, basic rate mechanisms (convection, transfer, reaction) and constitutive equations. For each conserved quantity differential balance equations are constructed within each balance volume using the specified transport terms. The transport mechanisms relating to each of these terms are supplied in the form of individual equations whilst thermodynamic and physical property relationships, balance volume constraints and control and equipment relations serve to complete the model. Consistency of the resulting model equations is ensured by construction, achieved by allowing selection only from predefined model elements. The second step in the modelling procedure is performed by the *model modifier*, which uses a set of assumption classes, ordered by their influence on the model equations, to reduce the model to a simplified form. The simplification procedure is based on a formal representation of modelling assumptions and their implications proposed by Hangos and Cameron [1997], which defines a syntax of modelling assumptions and uses the notions of syntactical correctness and semantical consistency both to check their validity and to transform the initial model generated in the first step into a simplified form. An attractive feature of this approach is that it allows the user to investigate the effect of modelling assumptions on process models and generate alternative, consistent models which are clearly documented.

BIMAP [Dieterich and Eigenberger, 1995; Dieterich *et al.*, 1997] was the first implemented tool to be reported for the automatic generation of distributed parameter process models and is designed for use in chemical reaction engineering applications. It is based on a general form of the equations of change in one dimension which are then modified to a specific form using assumptions and simplifications specified by the user via a menu-driven graphical interface. The mathematical derivation of the specific form of the balance equations from the general form is performed by MAPLE, a program for symbolic computation. This is essentially the same approach as presented elsewhere [Froment and Bischoff, 1990; Gerstlauer *et al.*, 1993] and the fixed number of modelling assumptions and simplifications result in the program having a rigid structure. In addition to the balance equations BIMAP also generates some of the model closure relations and, in the current implementation, the user may choose from a number of equations of state and diffusive mass transfer models and also has the option of using one of several empirical pressure drop relations rather than a momentum balance. The output consists of the generated model in LaTeX form together with a list of the assumptions and simplifications specified by the user. The principal limitation of BIMAP is that the generated models are restricted to spatial variations in one dimension and there is no facility for the automatic generation of an appropriate set of boundary conditions (although the authors have developed a separate program, MODUSIM [Dieterich and Eigenberger, 1997], for this purpose - see below). Furthermore, it is intended for single process units (primarily reactors) and is not suited to modelling of entire plants. BIMAP does include facilities for the modelling of simple multiphase systems using single phase conservation equations coupled through a (constant) local void fraction and interfacial area together with interphase exchange terms. However, the heterogeneous reactor example given in the published work, modelled using conservation equations in spherical coordinates for the catalyst and cylindrical coordinates for the fluid phase, does not represent a rigorous formulation and indeed may give incorrect results upon solution in many instances [Froment and Bischoff, 1990].

A prototype implementation of an unnamed computer program for the automatic generation of combined lumped and distributed parameter process models has been developed at the Technical University of Denmark by Jensen and Gani [1996]. The methodology is based upon the application of the conservation principles over *control shells*. These must be defined such that they contain either a single phase (or pseudo-homogeneous mixture) whose properties are continuous across the entire control shell or a perfectly mixed multiphase system whose properties exhibit no spatial variation. A single set of microscopic balance equations or a set of macroscopic balances for each phase is then generated for each control shell from a general *reference model* according to assumptions and simplifications specified (from a fixed set) by the user. The issue of turbulence modelling is not discussed in the published work and, in fact, the general reference model given contains microscopic equations based on the local instantaneous formulation and is

therefore limited to the description of laminar flows for all practical purposes (see chapter 2). The automatic generation of boundary conditions is not considered in the prototype version and these must be user defined. The implementation does include facilities for the generation of some of the closure relations and again allows empirical pressure correlations to be used instead of a momentum balance. This methodology is likely to be adequate for lumped parameter and (laminar) single phase distributed parameter modelling but the practicality of a control shell approach for multiphase distributed parameter models is open to question. The example given for the modelling of a distillation tray using microscopic balances for the liquid and vapour regions and a macroscopic balance around the interface seems to be rather cumbersome and the question of appropriate boundary conditions for this case is not considered. It is also unclear from the published work whether this implementation allows dimensional reduction of the microscopic balance equation by integral averaging (see chapter 2) and hence whether it is possible, for example, to include an average term for radial heat transfer in a pipe modelled using spatial variation in the axial direction only. More recently the same authors have developed ModDev [Jensen and Gani, 1999], a computer aided modelling system that assists in the derivation, analysis and manipulation of model equations. ModDev consists of a generic modelling language, ModDef, which allows the representation of a process model in terms of equation embedded building blocks and a model building system, ModGen, built around the prototype program described above but limited to the generation of lumped parameter process models.

Model.la [Bieszczad *et al.*, 1999] is a modelling environment developed from the modelling language of the same name [Stephanopoulos *et al.*, 1990] supporting the representation, automatic generation and numerical solution of lumped and distributed parameter process models. In Model.la process systems are represented based on their topological structure, material characterisation and the mechanistic characterisation of the physico-chemical phenomena assumed to occur within them. The topological structure is described by two fundamental modelling elements: modeled-units and fluxes. Modeled-units represent control volumes. Fluxes represent the transport of material or energy between modeled-units. Model.la also supports the hierarchical grouping of basic elements into composite modeled-units to ease topological representation. Modeled-units are characterised by a material-content, representing the phases (single phase or multiple phases in equilibrium), chemical species and reactions assumed to occur within them. Vessel geometry may also be declared to capture interactions between the volume and height of a material-content and to allow the port position of fluxes to be specified. Additional modelling elements arise from the phenomena-based mechanistic characterisation of the topological and material modelling elements. These include transport-mechanisms (for fluxes), rate-laws (for reactions) and (thermodynamic) property-models (for phases). Model.la uses a modelling logic to generate models from the process rep-

resentation both in the form of a language-based description (using the format of the original Model.la language) and an equation-based representation. The latter consists of lumped or distributed parameter mass and energy (but not momentum) conservation equations for each modeled-unit, equilibrium relationships for each material-content and thermodynamic and physical property relations for each phase and convective flux. In addition, transport-mechanism, rate-laws and property-models give rise to mechanistic and empirical relationships whilst geometrical assumptions introduce conditional equations describing discontinuous physical behaviour arising from port/phase position within a modeled-unit. The Model.la implementation (written in Borland C++) consists of three main parts: the model generator, the properties manager and the numerical engine. The model generator provides an interface for declaration of the process representation and incorporates the model generation logic outlined above. The properties manager is responsible for the physical and thermodynamic property relationships, which are invoked by accessing a commercial database. Finally, the numerical engine includes facilities for structural analysis of the generated models (including selection of design variables and differential index checking) and interfaces with the *gPROMS* modelling environment [Barton and Pantelides, 1994; Oh and Pantelides, 1996] for numerical solution. No examples of generated models are given in the published work. It would appear, however, that Model.la is not suitable for the modelling of systems with detailed fluid mechanics since its model generation methodology is based around the conservation of mass and energy only.

In all of the prototype tools described above connection equations (for lumped parameter models) and boundary conditions (for distributed parameter models) must be either supplied by the user or invoked from a pre-defined library. MODUSIM [Dieterich and Eigenberger, 1997], a simulation program written in Fortran77, was the first computer aided modelling tool to be reported with capabilities for the automatic generation of boundary conditions and connection equations (although it has no facilities for the automatic generation of the interior conservation equations). In MODUSIM connection equations and boundary conditions are generated by equating fluxes and intensive variables (for example concentration and temperature) between two connected submodels until a sufficient number of equations have been generated for the set to be determinate. MODUSIM appears to have been developed based on chemical reaction engineering applications and only the generation of Danckwerts-type boundary conditions [Danckwerts, 1953] is discussed in the published work. It is not clear whether this methodology will always generate the correct boundary conditions between systems with different mixing characteristics [Barber *et al.*, 1998] and whether alternative boundary conditions, such as specifying an empirical flux relation or no-slip at a solid boundary or using symmetry conditions, can be generated automatically.

To summarise the above discussion, figure 1.1 (below) attempts to show the

relationship between the various modelling tools by considering their scope on a process modelling 'coordinate' which represents the process modelling activity from process representation through to model solution.

For further reading recent reviews of computer-aided process modelling tools and methodology have been given in papers by Marquardt [1991; 1994; 1996], Pantelides and Britt [1995] and Stephanopoulos and Han [1996].

1.2 Computational Fluid Dynamics

An area of computer-aided process modelling which has become increasingly important over the last twenty years is that of computational fluid dynamics (CFD). This may be described as the activity of determining information of practical interest (for example velocity and pressure distributions) about fluid flow problems by computational methods (see, for example, [Anderson *et al.*, 1984; Muralidhar and Sundararajan, 1995; Versteeg and Malalasekera, 1995]). It involves the numerical solution of the governing differential equations and closure laws (see chapter 2) for the different categories of fluid type and dynamic behaviour. Whilst it is clear that mathematical descriptions used in CFD and computer-aided process modelling (which is defined here to mean those computer packages and tools described in sections 1 and 1.1 above) share a common theoretical basis in continuum mechanics, the development of computer-based CFD and process modelling tools has proceeded largely independently of one another, although some comparison between the two has been made [Sargent, 1983; Oh, 1995; Oh and Pantelides, 1996]. It is likely that this is due to their differing applications and is reflected in the scope and methodology of the two approaches.

The main objective in CFD is the quantitative prediction of fluid flow phenomena in a given system, with a particular focus on the effect of complex geometries and geometrical feature lines on the flow field. The emphasis is not on modelling of entire process units or plants as in 'conventional' process modelling but rather on the microscopic details of (usually) a single piece of a process unit. For example, distillation columns or even entire distillation trains comprising several columns are often analysed using traditional (commonly macroscopic) process models whilst CFD, on the other hand, would be used perhaps for the design of a single tray, valve or inlet duct. A typical CFD package comprises three main parts: pre-processing, numerical solution and post-processing.

Pre-processing is the process of problem specification and generation of some of the information required for numerical solution. It usually involves the following steps:

- surface modelling (definition of the system geometry)
- mesh generation
- model specification

- boundary and initial condition specification
- input of material properties

Surface modelling is the process of generating a mathematical description of the surface topography of the flow domain. In the current state-of-the-art CFD software surface modelling is performed using interfaces to computer-based graphics or computer-aided design packages, allowing the mathematical description to be generated automatically from a graphical representation. Mesh generation is the procedure of spatial subdivision of the flow domain contained within the surfaces defined in surface modelling. The mesh is used by the numerical solution procedures for the discretisation of the governing differential equations. The development of automatic mesh generation procedures is currently an area of active research. It is worth noting that surface modelling and mesh generation are often difficult and lengthy tasks, estimated to require over 50% of the time spent in industry on CFD projects [Versteeg and Malalasekera, 1995]. The choice of model variables, governing differential equations and closure laws together with appropriate boundary and initial conditions is then made from a fixed set of pre-defined models and finally the material properties for the selected model are specified. Although not listed above pre-processing may also include specification of some of the numerical solution parameters such as convergence tolerances.

Discussion of the numerical solution of the model equations is outside the scope of this thesis and instead the reader is referred to (for example) Anderson *et al.* [1984], Wulff [1990], Fletcher [1991], Muralidhar and Sundararajan [1995] or Versteeg and Malalasekera [1995].

Post-processing is used to manage and display the results of the analysis and involves the following:

- solution monitoring
- solution field output and display
- derived quantity calculation
- data export

Until recently the complexity of CFD packages meant that they were only used by expert users and post-processing was considered unimportant. However, with CFD becoming more widely used in industry, post-processing has received considerable attention in recent years and the current state-of-the-art CFD packages use sophisticated software tools for management and visualisation of the potentially vast amount of data generated in a CFD analysis.

A description of some commercial CFD packages is given by [Spalding and Hewitt, 1992].

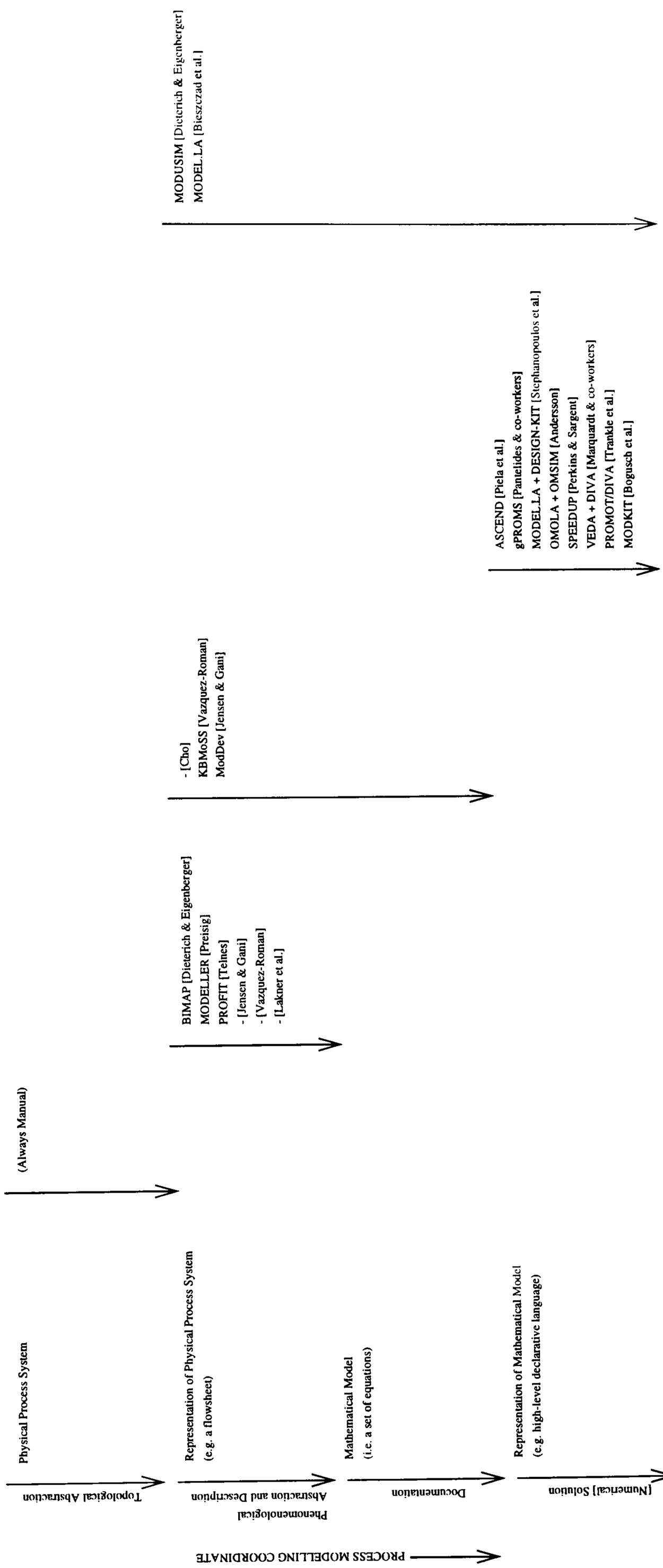


Figure 1.1: Computer-Aided Process Modelling Tools

1.3 Motivation and Objectives

In the opening section of this chapter an introduction to computer-aided software tools for process modelling was given. It was noted that the principal limitation of these tools is that they require the user to provide a suitable mathematical model for the particular process system of interest. It was also noted that there has been little development of computer aids for actually generating such models based on the underlying physical characteristics and behaviour. Those few prototype tools that have been reported in the literature which try to address this issue were then reviewed critically.

In section 1.2 an introduction to the field of computational fluid dynamics (CFD) was given. It was noted that the development of computer-based CFD packages and traditional process modelling tools has proceeded largely independently of one another, despite the fact that, fundamentally at least, both are dealing with systems based on the same physico-chemical phenomena.

Motivated by the limitations of modelling languages and modelling environments, this thesis is concerned with the development of a software tool for the automatic generation of mathematical models of process systems. Despite the progress that has been made, this area is still in its infancy. To date no truly general purpose model-generation approach is available and existing software tools can only be regarded as prototypes. Moreover, the research that has been carried out has focused largely on the generation of more traditional process models (see section 1.1). There is no published work which makes any real attempt to reconcile the approaches of the traditional process modelling and CFD communities and to develop a methodology which is truly consistent with both. Together these facts provide the motivation for the specific focus of the work presented in this thesis.

The major objective of this research then, which is a continuation of earlier work at Imperial College by Vázquez-Román [1992] and Cho [1997], is the development of a software tool for the automatic generation of combined lumped and distributed parameter process models from purely physical descriptions. The particular focus of the work is on the generation of models describing both spatial and temporal variation of the system properties which are consistent with the underlying fluid mechanics. To date, no work of comparable scope and generality has been reported.

1.4 Thesis Outline

This thesis is composed of 7 chapters. Chapter 1 discusses the background to the work and gives a critical review of the current status of computer-aided process model generation. It identifies the motivation for the project and sets out its main objectives. In chapter 2 the theory of mathematical modelling of single and multiphase systems is considered.

This material forms the basis of the model generation approach developed in this thesis. A process representation and systematic model building procedure are defined in chapters 3 and 4 respectively. Together these permit the generation of mathematical models of process systems from purely physical descriptions. Details of their implementation in the computer program gMaGiC are given in chapter 5, and in chapter 6 the validity of both the methodology and software are demonstrated by their application to several nontrivial cases studies. These are chosen to illustrate the success of the approach in modelling systems exhibiting a variety of physico-chemical behaviour. Finally in chapter 7 conclusions are drawn and some suggestions for future work are given.

Chapter 2

Mathematical Modelling of Single and Multiphase Systems

In this chapter the formulation of the governing three dimensional partial differential equations for the mathematical modelling of single and multiphase systems is considered together with a discussion of appropriate initial and boundary conditions. Consideration is also given to the derivation of simplified models. In the context considered here a mathematical model refers to the mathematical description of a process system based on physico-chemical transport phenomena. Other model bases such as population balances and empirical models are not discussed. This approach relies on continuum mechanics and provides a sound theoretical basis upon which to build a computer aided process modelling tool, as well as forming the core of both traditional and computational fluid mechanics.

2.1 Single Phase Systems

The mathematical modelling of a system comprising a single phase confined within a defined boundary is based upon mathematical statements describing the principles of conservation of mass, energy and momentum.

The most fundamental description would be based on molecular considerations but for engineering purposes this approach involves such intractable mathematics as to render it unfeasible in all but a few special cases. A more practical description is that based on continuum mechanics where, by making use of the continuum hypothesis (see, for example, Richardson [1989]), the material properties are regarded as continuous functions of time and position. The governing field equations (2.1 - 2.3[1]), the so-called local instantaneous equations, equations of change or microscopic equations, are then derived by applying the conservation principles over a stationary volume element or by an integral balance over an arbitrary control volume (see, for example, Bird *et al.* [1960] or Richardson [1989]). All nomenclature is as defined in appendix A.

[1]Several alternative but completely equivalent formulations of (2.3) are available.

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho \mathbf{v}) = 0 \tag{2.1}$$

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla.(\rho \mathbf{v}\mathbf{v}) - \nabla.\tau + \nabla p - \rho \mathbf{g} = 0 \tag{2.2}$$

$$\frac{\partial}{\partial t}(\rho \hat{E}) + \nabla.(\rho \mathbf{v} \hat{H}) + \nabla.\mathbf{q} - \nabla.(\tau.\mathbf{v}) - \rho(\mathbf{v}.\mathbf{g}) = 0 \tag{2.3}$$

where $\hat{E}$ and $\hat{H}$ are the total energy and total enthalpy respectively and include the kinetic energy (2.4).

$$\hat{E} = \hat{U} + \frac{1}{2}\mathbf{v}.\mathbf{v} \qquad\qquad \hat{H} = \hat{h} + \frac{1}{2}\mathbf{v}.\mathbf{v} \tag{2.4}$$

For incompressible fluids, in which there are no density variations, there is no linkage between the energy equation (2.3) and the mass and momentum conservation equations (2.1 and 2.2). Consequently the flow field can be solved by consideration of mass and momentum conservation only. The energy equation only needs to be solved alongside the others if the system involves heat transfer.

The local instantaneous equations together with boundary and initial conditions and the thermal and calorific equations of state (2.5 and 2.6 respectively) give the complete description of the flow system.

$$\rho = \rho(p, T, \mathbf{x}) \tag{2.5}$$

$$\hat{h} = \hat{h}(p, T, \mathbf{x}) \tag{2.6}$$

In theory this approach is valid for both laminar and turbulent flows and applies to both Newtonian and non-Newtonian fluids [Bird, 1957].

However the apparent randomness of turbulent motion means that direct (numerical) solution of the local instantaneous equations is not practical for turbulent flows. Extremely fine temporal and spatial discretisation of the flow domain would be required to capture processes at the wide range of frequencies and length scales encountered in a turbulent system (Speziale [1991] estimated that simulation of a turbulent pipe flow at a Reynolds number of 500000 would require computing resources 10^6 times faster than the state-of-the-art supercomputers available at that time). Instead a statistical approach is adopted using an averaging procedure to smooth out the turbulent fluctuations which are then accounted for in a semi-empirical manner.

For incompressible fluids the most common approach is to use conventional or Reynolds averaging, whereby the average value of each dependent variable $\overline{x}$ is defined, according to (2.7), in terms of an instantaneous component x and a fluctuating component (due to the turbulence) x'.

$$x = \overline{x} + x' \tag{2.7}$$

Clearly, by the foregoing definitions the double-average of each variable must be equal to the single average (2.8) and, from (2.7), the average of the fluctuating component must therefore be zero (2.9). Note however that in general the average of the product of the fluctuating components is not equal to zero (2.10).

$$\overline{\overline{x}} = \overline{x} \tag{2.8}$$

$$\overline{x'} = 0 \tag{2.9}$$

$$\overline{x'x'} \neq 0 \tag{2.10}$$

From a theoretical point of view, for the general case of three dimensional unsteady turbulent flow, the average of each variable $\overline{x}$ must be an ensemble average. However, in practice it is common to replace ensemble averages by time averages in unsteady flows by assuming that the turbulent fluctuations are much more rapid than the time variation of the mean flow (see also section 2.2.2). For stationary flows the ergodic hypothesis is that the ensemble and time averages are identical and in this case ensemble or time averaged variables may be used [Hinze, 1975]. Definitions of both a time and ensemble average are given in appendix B.

Using (2.7) the dependent variables in the local instantaneous (constant density) mass and momentum conservation equations may be expressed in terms of their average and fluctuating components and (time or ensemble) averaged using the relations given above.

$$\nabla.\overline{\mathbf{v}} = 0 \tag{2.11}$$

$$\frac{\partial}{\partial t}(\rho\overline{\mathbf{v}}) + \nabla.(\rho\overline{\mathbf{v}}\overline{\mathbf{v}}) + \nabla\overline{p} - \rho\mathbf{g} = \nabla.(\overline{\tau} - \rho\overline{\mathbf{v}'\mathbf{v}'}) \tag{2.12}$$

Note that irrespective of the type of averaging used (ensemble or time), the terms in equations (2.11 and 2.12) correspond exactly to those in the local instantaneous equations (for incompressible flows) but refer to averaged variables. In addition extra terms arise representing the average of the cross products of the fluctuating components. These new terms, known as the Reynolds stresses, must be modelled as functions of the mean flow quantities in order to close the system of governing equations. This is known as turbulence modelling. The most common approach is based on the generalised Boussinesq approximation where, because of their tensorial nature, the Reynolds stresses are defined in analogy to their molecular counterparts in a semi-empirical manner (see, for example,

Anderson *et al.* [1984], Fletcher [1988] or Muralidhar and Sundararajan [1995]). For example, by analogy with Newton's Law of viscosity for the molecular stress tensor, the Reynolds stress term in the momentum equation can be defined as (using index notation)

$$-\rho\overline{v_i'v_j'} = (\overline{\tau}_{ij})_t = -\frac{2}{3}\rho k\delta_{ij} + \mu_t\left(\frac{\partial\overline{v}_i}{\partial z_j} + \frac{\partial\overline{v}_j}{\partial z_i}\right) \tag{2.13}$$

where μ_t is the eddy or turbulent viscosity and k the kinetic energy of turbulence. This is particularly convenient since it allows the averaged momentum equation (2.12) to be written as

$$\frac{\partial}{\partial t}(\rho\overline{\mathbf{v}}) + \nabla.(\rho\overline{\mathbf{v}\mathbf{v}}) + \nabla\overline{p} - \rho\mathbf{g} = \nabla.(\overline{\tau}_e) \tag{2.14}$$

where

$$\overline{\tau}_e = \overline{\tau} + \overline{\tau}_t \tag{2.15}$$

A number of empirical or semi-empirical (closure) relationships are available for the determination of μ_t and k, ranging in refinement from simple algebraic or zero- (partial differential) equation models (for μ_t only), such as Prandtl's mixing length, through to two-equation models involving both algebraic and partial differential equations, such as the well-known k-ϵ viscosity model (see, for example, Launder and Spalding [1972] or Wilcox [1993]).

A similar treatment of the energy conservation equation (2.3) encounters difficulties since numerous additional terms arise from the cross products of the fluctuating velocities with the other variables in the equation. The normal approach in CFD codes is to define a scalar transport equation, *viz.*

$$\frac{\partial}{\partial t}(\rho\phi) + \nabla.(\rho\phi\mathbf{v}) = \nabla.(\Gamma\nabla\phi) + S_\phi \tag{2.16}$$

in which the scalar quantity ϕ is taken to be either enthalpy or temperature[2]. Application of Reynolds decomposition and averaging as before gives (with constant density)

$$\frac{\partial}{\partial t}(\rho\overline{\phi}) + \nabla.(\rho\overline{\phi\mathbf{v}}) = \nabla.(\Gamma\nabla\overline{\phi} - \rho\overline{\phi'\mathbf{v}'}) + \overline{S}_\phi \tag{2.17}$$

where only the scalar Reynolds stresses appear explicitly and the troublesome extra correlations are lumped together in the turbulent source term $\overline{S_\phi}$.

The Reynolds stress terms in the averaged scalar species mass conservation and energy transport equations may be treated in a similar way to those in the momentum equation using relationships analogous to Fick's law and Fourier's law respectively. Note

[2]The same equation is also used to describe conservation of chemical species x_i. in combusting flows (see, for example, Gosman and Issa [1996] or Jones [1996]).

that, in common with molecular transport, the turbulent proportionality coefficients in these relationships (the turbulent viscosity, diffusion coefficient and conductivity) are related through the dimensionless (turbulent) Prandtl and Schmidt numbers.

Alternative approaches to turbulence modelling, such as stress-equation models and large eddy simulation, are available but at present their use is mainly limited to research applications [Anderson *et al.*, 1984]. It should be noted that the complexity of turbulent flows means that, despite over a century of study, turbulence is still not completely understood and turbulence modelling remains an area of active research.

For compressible flows Reynolds decomposition is inconvenient since it introduces additional terms containing the fluctuating density and therefore the resulting equations do not correspond to the equivalent local instantaneous equations. For example the Reynolds averaged continuity equation with variable density becomes

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla.(\overline{\rho \mathbf{v}} + \overline{\rho' \mathbf{v}'}) = 0 \tag{2.18}$$

and involves the additional averaged cross-product of ρ' and $\mathbf{v}'$.

A more convenient approach is to use Favre or mass-weighted averaging whereby mass-averaged variables and fluctuating components are defined according to (2.19) and (2.20) respectively.

$$\tilde{x} = \frac{\overline{\rho x}}{\overline{\rho}} \tag{2.19}$$

$$x = \tilde{x} + x'' \tag{2.20}$$

In this case the (time or ensemble) average of each mass-averaged variable is equal to the single mass-average (2.21) but note that the average of the mass-averaged fluctuating component is not equal to zero (2.22). However it can be shown that the average of the product of the density and the mass-averaged fluctuating component is equal to zero (2.23).

$$\overline{\tilde{x}} = \tilde{x} \tag{2.21}$$

$$\overline{x''} \neq 0 \tag{2.22}$$

$$\overline{\rho x''} = 0 \tag{2.23}$$

Substituting from (2.20) for the velocity components and thermal variables and from (2.7) for pressure and density in (2.1 - 2.3), in this approach pressure and density are treated as before, and (time or ensemble) averaging using the above relations gives the mass-averaged conservation equations which are the basis for the modelling of (single

phase) compressible flows. An advantage of the mass-averaged formulation is that the energy equation (2.26) is treated in the same way as the mass and momentum equations (2.24 and 2.25). It therefore represents a more theoretically satisfying approach than the definition of somewhat artificial scalar transport equations (*cf.* 2.16 and 2.17).

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla.(\overline{\rho}\tilde{\mathbf{v}}) = 0 \tag{2.24}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{\mathbf{v}}) + \nabla.(\overline{\rho}\tilde{\mathbf{v}}\tilde{\mathbf{v}}) + \nabla\overline{p} - \overline{\rho}\mathbf{g} = \nabla.(\overline{\tau} - \overline{\rho\mathbf{v}''\mathbf{v}''}) \tag{2.25}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{\hat{E}}) + \nabla.(\overline{\rho}\tilde{\mathbf{v}}\tilde{\hat{H}}) + \nabla.\overline{\mathbf{q}} - \nabla.(\overline{\tau}.\tilde{\mathbf{v}} - (\overline{\rho\mathbf{v}''\mathbf{v}''}).\tilde{\mathbf{v}}) - \overline{\rho}(\tilde{\mathbf{v}}.\mathbf{g}) = -\nabla.(\overline{\rho\mathbf{v}''\hat{h}''}) \tag{2.26}$$

The (averaged) total energy and total enthalpy now also include k, *viz.*

$$\tilde{\hat{E}} = \tilde{\hat{U}} + \frac{1}{2}\tilde{\mathbf{v}}.\tilde{\mathbf{v}} + k \qquad\qquad \tilde{\hat{H}} = \tilde{\hat{h}} + \frac{1}{2}\tilde{\mathbf{v}}.\tilde{\mathbf{v}} + k \tag{2.27}$$

$$k = \frac{1}{2}\overline{\mathbf{v}''.\mathbf{v}''} \tag{2.28}$$

It can be seen that the terms in these equations again correspond exactly to those in the equivalent local instantaneous equations (for compressible flows) but in terms of averaged variables. Furthermore additional Reynolds stress terms again appear[3] which can be defined in analogy to the molecular fluxes as before using a form of the Boussinesq approximation generalised for compressible flows (2.29).

$$-\overline{\rho v_i'' v_j''} = (\tilde{\tau}_{ij})_t = -\frac{2}{3}\overline{\rho}k\delta_{ij} + \mu_t\left(\frac{\partial \tilde{v}_i}{\partial z_j} + \frac{\partial \tilde{v}_j}{\partial z_i} - \frac{2}{3}\delta_{ij}\frac{\partial \tilde{v}_k}{\partial z_k}\right) \tag{2.29}$$

Again, this is particularly convenient since it allows the averaged momentum and energy equations (2.25 and 2.26) to be written

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{\mathbf{v}}) + \nabla.(\overline{\rho}\tilde{\mathbf{v}}\tilde{\mathbf{v}}) + \nabla\overline{p} - \overline{\rho}\mathbf{g} = \nabla.(\overline{\tau}_{\mathbf{e}}) \tag{2.30}$$

$$\frac{\partial}{\partial t}(\overline{\rho}\tilde{\hat{E}}) + \nabla.(\overline{\rho}\tilde{\mathbf{v}}\tilde{\hat{H}}) + \nabla.\overline{\mathbf{q}}_e - \nabla.(\overline{\tau}_e.\tilde{\mathbf{v}}) - \overline{\rho}(\tilde{\mathbf{v}}.\mathbf{g}) = 0 \tag{2.31}$$

The relationships for the molecular flux terms in (2.25 - 2.26) are also defined in terms of the mass-averaged variables $\tilde{x}$ and averaged fluctuating components $\overline{x''}$ and therefore represent more complex expressions than their conventionally averaged counterparts. In

[3] A further two terms $\nabla.(\overline{\tau.\mathbf{v}''})$ and $-\overline{\rho\mathbf{v}''\frac{1}{2}\mathbf{v}''\mathbf{v}''}$, corresponding to the molecular diffusion and turbulent transport of the kinetic energy of turbulence respectively, should also appear on the right hand side of the averaged energy equation but these are usually only important in supersonic flows and are omitted in (2.26) (see, for example, Rubesin and Rose [1973] or Wilcox [1993]).

practice, however, terms involving doubly primed fluctuations in the flux relationships are likely to be small and are often neglected [Anderson *et al.*, 1984].

For incompressible flows (2.24 and 2.25) reduce to (2.11 and 2.12) and thus for flows in which density fluctuations can be neglected (and energy conservation is not considered) the formulations based on conventional and mass-weighted averaging become identical.

More details on averaging procedures and turbulence modelling can be found in Rubesin and Rose [1973], Anderson *et al.* [1984], Wilcox [1993] and Jones [1996].

It is interesting to note that for consistency the constitutive and closure laws needed to complete the mathematical description of a given system should be based on the same type of averaging as the averaged conservation equations themselves. In theory, for the general case of unsteady turbulent flow, this should be ensemble averaging as mentioned above. However, most experimental measurements of variables that appear in these laws (for example temperature and pressure) are, in fact, time averaged quantities and in practice, despite the basic inconsistency, conservation equations based on ensemble averaging are frequently used with constitutive and closure laws based on time averaging.

2.2 Multiphase Systems

The characterising features of a multiphase system confined within a defined boundary are the existence of one or more interfaces between the phases and the associated discontinuities of properties across these interfaces. From a theoretical point of view the principal difficulties encountered in modelling multiphase systems arise from the presence of moving interfaces and the fact that both the steady state and dynamic characteristics of the system depend on the internal phase distribution [Ishii, 1975].

2.2.1 Local Instantaneous Formulation

It is theoretically possible to derive a mathematical description of a multiphase system based on continuum mechanics, by considering the system as a field subdivided into single phase regions separated by moving boundaries. Clearly the standard conservation equations (see section 2.1) can be applied over each of these subregions up to an interface but, because of the continuum hypothesis, not across it. Furthermore the conservation equations can also be applied to the interfaces themselves since each one may be considered simply as a singular case of a continuous field. The interfacial balance equations are known as jump conditions and serve to match the solutions of the conservation equations for the single phase regions on either side of an interface.

This approach to multiphase modelling is known as the local instantaneous formulation and results in a moving multi-boundary problem. However, since in general the positions of the interfaces are unknown *a priori* the mathematical difficulties associated

with this approach mean that a solution is only accessible in a few simple cases, such as laminar stratified vapour-liquid flow or in the analysis of a single bubble. The importance of this approach is that it provides the fundamental framework upon which practical multiphase models are built.

A general form of the local instantaneous conservation equations (2.32) and jump conditions (2.33) is given below [Ishii, 1975; Bouré, 1987; Yadigaroglu, 1988]

$$\frac{\partial}{\partial t}(\rho_k \psi_k) + \nabla.(\rho_k \psi_k \mathbf{v_k}) + \nabla.\mathbf{J}_k - \rho_k \chi_k = 0 \tag{2.32}$$

$$\sum_k (\Gamma_k \psi_k + \mathbf{n_k}.\mathbf{J_k}) = 0 \tag{2.33}$$

where the values of ψ_k, $\mathbf{J}_k$ and χ_k are obtained by substituting from table 2.1. In some formulations additional terms are included in the local instantaneous jump conditions for momentum and energy (2.33) to account for the effects of interfacial surface tension (see, for example, Drew and Wallis [1994] or Lahey [1998]).

	ψ_k	$\mathbf{J}_k$	χ_k
Mass Conservation	1	0	0
Momentum Conservation	$\mathbf{v}_k$	$-\mathbf{T}_k$	$\mathbf{F}$
Total Energy Conservation	$\hat{U} + \frac{1}{2}\mathbf{v}_k.\mathbf{v}_k$	$\mathbf{q}_k - \mathbf{T}.\mathbf{v}_k$	$\mathbf{F}.\mathbf{v}_k$

Table 2.1: Local Instantaneous Formulation

2.2.2 Averaged Models

The mathematical intractability of the local instantaneous formulation means that it is of little use for practical engineering purposes. Moreover, the microscopic details of a given process system are rarely needed, rather it is more important to accurately predict the behaviour on a more macroscopic level. Thus, for multiphase systems, the almost universal approach is to use averaged models. These are derived by applying one or more averaging operators to the local instantaneous equations and may be thought of as a 'low pass filtering which eliminates unwanted high frequency signals from local instantaneous fluctuations' [Ishii, 1975]. However, these local fluctuations must clearly affect the macroscopic behaviour and, despite not considering them explicitly, their overall statistical contribution must be taken into account in the constitutive and closure relations. This approach is analogous to turbulent viscosity models for single phase systems where the turbulent motion is accounted for in a semi-empirical manner. The closure problem is a considerable one and is discussed further below. It is worth noting that there is still considerable disagreement between models proposed by different workers not only in

terms of the closure relations but even in terms of the basic two-fluid governing equations [Sinclair, 1999].

2.2.2.1 Model Formulation

In common with averaged single phase models, averaged multiphase three dimensional balance equations may be based either on ensemble or time averaging. Only the latter are presented here. From a practical point of view a time averaged formulation has the advantage of being mathematically consistent when used with constitutive and closure laws which (typically) involve time averaged variables (see section 2.1).

The basis of time averaging is the assumption that a fixed time interval δt can be defined which is large enough to smooth out the local instantaneous fluctuations of the fluid properties yet small compared with the macroscopic time constant of the unsteadiness of the bulk flow [Ishii, 1975]. When this is impossible ensemble averaging must be used. Together with this smoothing the averaging procedure also has the (mathematical) consequence of bringing the single phase regions, which are alternatively occupying a volume element, into a number of continua simultaneously existing at the same point using a statistical local time fraction or local void fraction. This is perhaps the single most important feature of multiphase modelling since it allows the application of continuum mechanics and provides a description which is mathematically consistent with the continuum hypothesis. However, it must be recognised that by its very nature the averaging procedure obscures the instantaneous locations of the interfaces between the phases[4]. An illustration of the derivation of averaged multiphase models from the local instantaneous formulation is given in figure 2.1 [Ishii, 1990].

Time averaged conservation equations (2.34 - 2.36) and (simplified) jump conditions (2.37 - 2.40) derived by Ishii [1975] are given below. A summary of the averaging operators required for this derivation is given in appendix B.

$$\frac{\partial}{\partial t}(\epsilon_k \overline{\overline{\rho}}_k) + \nabla.(\epsilon_k \overline{\overline{\rho}}_k \tilde{\mathbf{v}}_k) = \Gamma_k \tag{2.34}$$

$$\frac{\partial}{\partial t}(\epsilon_k \overline{\overline{\rho}}_k \tilde{\mathbf{v}}_k) + \nabla.(\epsilon_k \overline{\overline{\rho}}_k \tilde{\mathbf{v}}_k \tilde{\mathbf{v}}_k) - \nabla.(\epsilon_k \overline{\overline{\tau}}_e) + \nabla(\epsilon_k \overline{\overline{p}}_k) - \epsilon_k \overline{\overline{\rho}}_k \mathbf{g} = \mathbf{M}_k \tag{2.35}$$

$$\frac{\partial}{\partial t}(\epsilon_k \overline{\overline{\rho}}_k \tilde{\hat{H}}_k) + \nabla.(\epsilon_k \overline{\overline{\rho}}_k \tilde{\mathbf{v}}_k \tilde{\hat{H}}_k) + \nabla.(\epsilon_k \overline{\overline{\mathbf{q}}}_e) - \frac{D_k}{Dt}(\epsilon_k \overline{\overline{p}}_k) - \epsilon_k(\overline{\overline{\tau}}_e : \nabla \tilde{\mathbf{v}}_k) = \Lambda_k \tag{2.36}$$

[4] Wulff [1990] notes that the averaged multiphase conservation equations are commonly misapplied to predict the horizontal liquid or mixture level in a vertical duct, understood in these cases to be a sharp discontinuity in the vapour local void fraction.

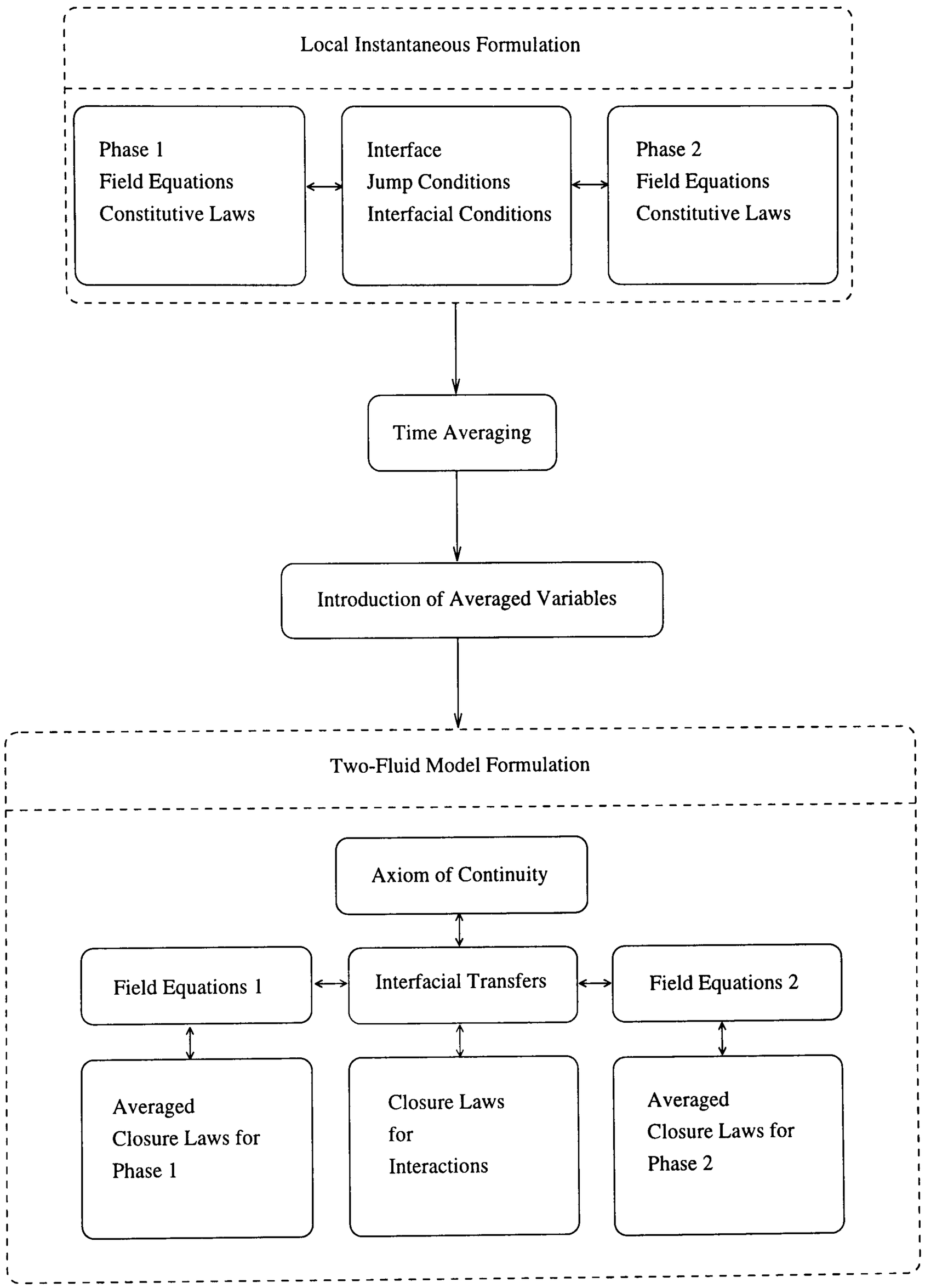


Figure 2.1: Averaged Model Formulation

$$\sum_k \Gamma_k = \sum_k a_i m_k = 0 \tag{2.37}$$

$$\mathbf{M}_k = \mathbf{v}_{ki}\Gamma_k + \overline{\overline{p}}_{ki}\nabla\epsilon_k + \mathbf{M}_{ik} - \nabla\epsilon_k.\tau_i \tag{2.38}$$

$$\sum \mathbf{M}_{ik} = 0 \tag{2.39}$$

$$\sum_k \Lambda_k = \sum_k a_i(m_k\hat{H}_{ki} + q_{ki}) \approx 0 \tag{2.40}$$

Alternative averaged two-fluid (or two-fluid type) formulations are also available. For example, Bouré and Delhaye [1982] give time averaged equations close to those of Ishii [1975] (along with a composite space/time averaged formulation), Drew and Wallis [1994] give a two-fluid model similar in philosophy but based on ensemble averages whilst Lahey [1998] presents a complex four field two-fluid model capable of accurate mechanistic predictions of a variety of vapour/liquid flows by solution with commercial CFD software.

2.2.2.2 Closure

The averaging procedure provides a much simplified equation set but inevitably involves the loss of the local instantaneous information. This must be resupplied in the closure relationships. Three categories of closure relationship may be identified: constitutive laws, transfer laws and topological laws [Bouré, 1987; Roberts, 1997].

Constitutive laws are similar to the constitutive laws of single phase flow and describe the state of a material. They include equations of state for density and enthalpy, thermodynamic and physical relations and also equations defining the (laminar and turbulent) viscous stress and conduction terms (although the latter and their associated derivatives are often neglected). From a theoretical point of view these relationships should be defined in terms of averaged quantities but, in practice, there is often little choice but to use classical laws replacing the instantaneous variables (*e.g.* ρ_k, p_k and h_k) by their (time-)averaged counterparts (*e.g.* $\overline{\overline{\rho}}_k$, $\overline{\overline{p}}_k$ and $\tilde{\hat{H}}_k$) [Bouré, 1987; Yadigaroglu, 1988].

Transfer laws describe the mass, momentum and energy transfer terms arising from fluxes across discontinuity surfaces (interfaces and boundary walls). They are usually obtained by splitting each term into the product of the surface area through which the transfer occurs with an appropriate flux density (although the two are related). The former are then expressed by topological laws (see below) whilst relations for the latter are frequently obtained in a rather *ad hoc* manner and are often empirical [Roberts, 1997].

Topological laws (a term introduced by Bouré) are a specific feature of averaged multiphase models and serve to resupply the information on the interface structure and

phase distribution lost during averaging. In general they would be complicated functions (most likely partial differential equations) of the phase (and interface) distribution and structure, time (because of inertial effects) and nonlocal effects in both space and time caused by the averaging. However, their form is often (usually) unknown and alternative closure methods must be found. Indeed, Bouré [1987] remarks:

> *"A decisive breakthrough in the field of two-phase flow modelling would be the development of a satisfactory void fraction topological law."*

The difficulties associated with formulating appropriate topological closure laws is highlighted by probably the most common simplification in two-fluid models for pipe flows, the single-pressure model. Here a single mixture pressure is assumed which enables the artificial elimination of one dependent variable in the model and, in effect, serves as a substitute for the void fraction topological law for both (or all) phases. However, this assumption, which forces pressure disturbances to propagate at the same velocity within the two phases, is unrealistic in many cases [Bouré, 1987; Roberts, 1997]

To reduce the computational effort, averaged multiphase models are often not used in the full three dimensional form given above but instead are further averaged over the cross-section of the containing duct (or more accurately over an infinitesimal volume slice to ensure continuous derivatives [Banerjee, 1997])(see section 2.4). This additional (composite) averaging eliminates two and three dimensional field distribution information and therefore additional closure laws are again required. These include relationships accounting for the cross-sectional pressure distribution, the so-called apparent or virtual mass effect, and equations relating the product of (cross-sectionally) averaged variables and the corresponding average of the product of local variables. In the latter case the two products are usually assumed equal but this can cause numerical difficulties upon solution unless the averaged fields are carefully chosen to have fairly flat cross-sectional velocity distributions. This has the important consequence that several fields may be needed even for only a two-phase mixture, depending on the flow regime.

Alternative approaches to the closure problem include selectively deriving balance equations for the mixture as a whole, leading to equations such as the Drift-Flux model [Bouré, 1987], or treating the mixture as a pseudo-homogeneous single phase fluid using phase-averaged physical properties.

In summary, it suffices to note that the major problem in the use of multiphase models is in the closure issue. Indeed, Hewitt [1997] states:

> *"The whole difficulty in the two-fluid models is in the closure relationships. Sometimes, one thinks that the only thing that the developers of such models have done is to formalise our ignorance!"*

Despite the difficulties associated with closure for the general case, in some chemical and process engineering applications averaged models are used routinely. For example, fixed bed reactors are commonly modelled using a component continuity and energy balance for each phase (fluid and solid), but often with an empirical pressure drop relation for the mixture rather than the rigorous momentum balances. Fluidized bed reactors can also be modelled using a two-fluid approach, but in this case one dimensional conservation equations are written for imaginary 'bubble' and 'emulsion' phases rather than for the true thermodynamic phases present. Note that in both cases averaged models can be used because relatively simple topological closure laws are known (often both of these quantities are taken as constant). Examples of such models can be found in many undergraduate texts on chemical reaction engineering (see, for example, Froment and Bischoff [1990]).

Further discussion of both the local instantaneous and averaged formulations and the closure issue can be found in Ishii [1975; 1990], Bouré and Delhaye [1982], Bouré [1987], Yadigaroglu [1988], Drew and Wallis [1994] and Roberts [1997].

2.3 Initial and Boundary Conditions

The conservation equations presented in sections 2.1 and 2.2 together with appropriate constitutive laws (equations of state, thermodynamic and physical relations, expressions relating the flux terms to intensive quantities *etc.*) and closure laws (for turbulent and multiphase systems) describe the behaviour of a given material within a general process system. To complete the mathematical description for a particular process system and to distinguish between different instances of that process system requires the specification of an appropriate set of boundary conditions on the edges of the spatial domains and initial conditions on (the initial point of) the time domain respectively. In general, this set must comprise n boundary conditions for each n-th order derivative with respect to a space variable and one initial condition for each time derivative in the model (the number of boundary conditions required for a given problem is discussed further in section 2.6.2).

Mathematically, there is no (conceptual) difference between initial and boundary conditions, since initial conditions are merely boundary conditions imposed on the temporal domain. Physically, however, there is a subtle conceptual difference between the two. Initial conditions define the initial state of the system of interest and, in general, may differ between instances of the same system. Boundary conditions, on the other hand, are a fundamental part of the physical description of the particular system and will be the same in all instances [Oh, 1995]. In terms of process modelling this has the important consequence that the boundary conditions must be determined as part of the physical description, whilst the initial conditions are not fixed and therefore cannot be derived or inferred without additional information. For this reason, the determination of initial conditions is not considered is this thesis. Appropriate boundary conditions for both

single and multiphase fluid flows are guided largely by physical arguments. Accordingly, it is helpful to consider the different types of boundary that may exist on the edges of a flow domain. For internal[5] single and multiphase flows (the latter described by averaged models) the following boundary types are important:

- solid (impermeable) walls
- open fluid interfaces
  - inflow (or inlet) boundaries
  - outflow (or outlet) boundaries
- symmetry boundaries
- cyclic boundaries

These boundaries are illustrated graphically in figure 2.2 and appropriate (physical) conditions for each in viscous flows are discussed below. For averaged models (turbulent and multiphase systems) additional boundary conditions may be required depending on the closure method.

Note that symmetry boundaries and cyclic boundaries do not represent true physical boundaries and indeed are not fundamental at the modelling stage. They are in fact deducible from the other boundary conditions (albeit by solving the model over a larger domain) and may be regarded merely as model simplifications. Nevertheless, they are commonly included as boundary conditions (see, for example, Richardson [1989] or Versteeg and Malalasekera [1995]) and as such are considered alongside the others below. Special forms of boundary condition and other complications arising from the use of simplified models are considered in section 2.5.

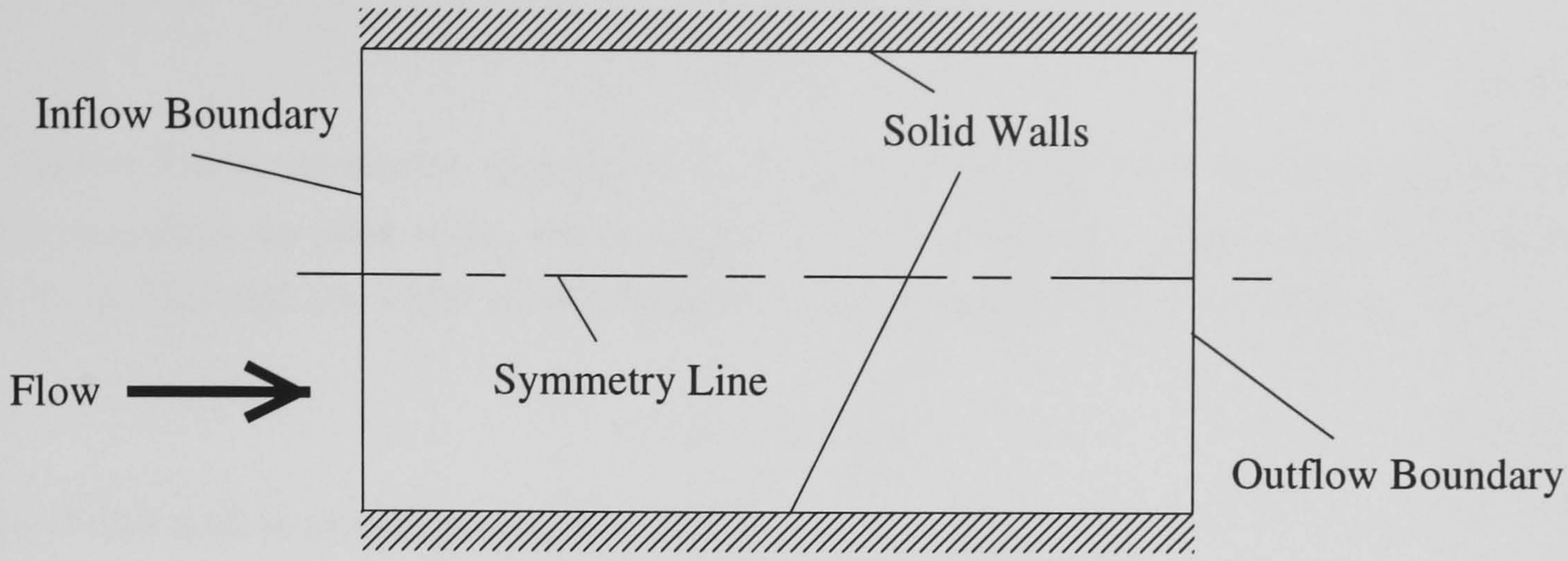
2.3.1 Solid Walls

At solid, impermeable walls kinematic arguments give rise to the no flow-through condition. This simply states that the fluid cannot penetrate the wall nor can an unbounded accumulation of fluid occur there [Richardson, 1989]. Consequently, no flow-through requires that the normal component of the fluid velocity at the wall is equal to the normal component of the velocity of the wall (2.41).

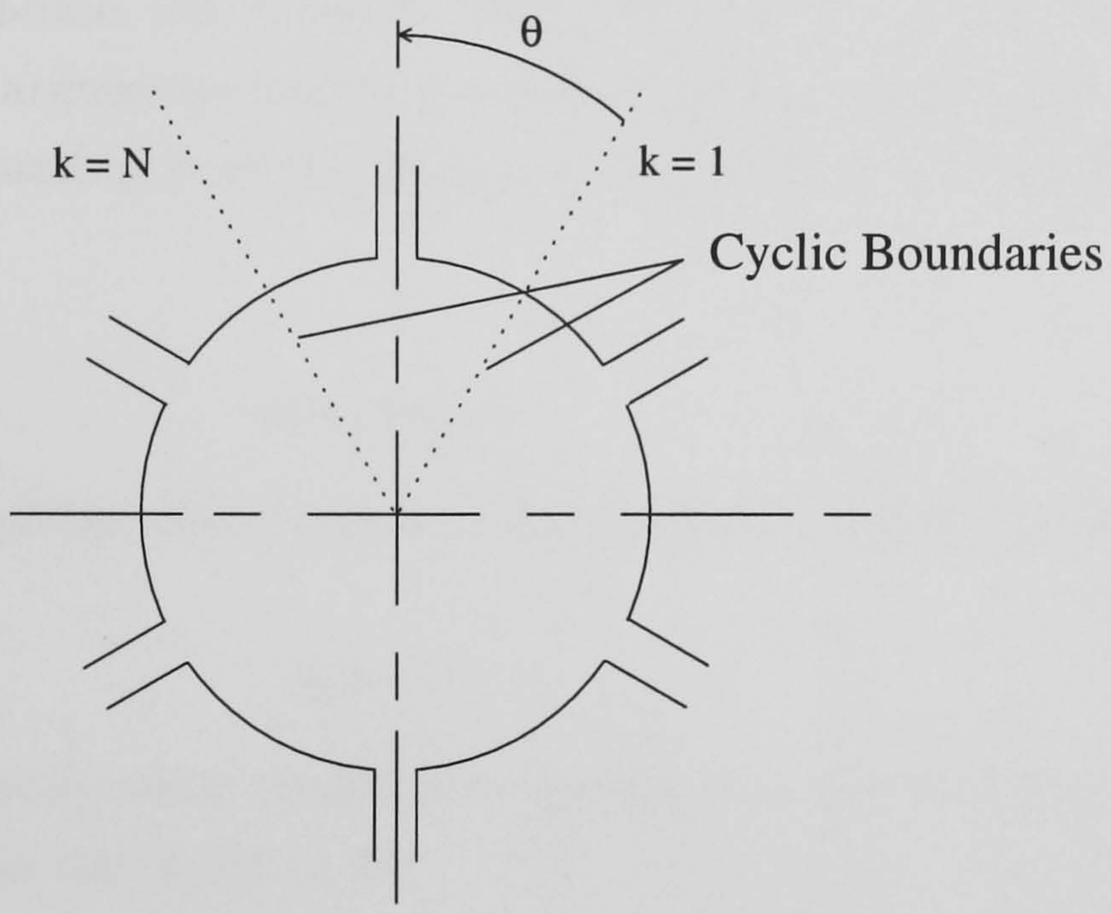
$$\mathbf{v}.\mathbf{n} = \mathbf{v}_w.\mathbf{n} \tag{2.41}$$

If the wall is at rest, then (clearly) the no flow-though condition becomes

[5] External flows are not considered in this thesis.



(a) Boundary Types for Internal Pipe Flow



(b) Cyclic Boundaries in a Burner Nozzle

Figure 2.2: Boundary Types in Internal Flows

$$\mathbf{v}.\mathbf{n} = 0 \tag{2.42}$$

For multicomponent mixtures, where diffusive (or, in turbulent systems, dispersive) fluxes are considered, the conservation arguments leading to the no flow-through condition also give rise to analogous 'no flux-through' conditions on the normal component of the diffusive (or dispersive) flux (2.43)[6].

[6] In some applications, particularly in chemical reaction engineering, diffusive fluxes are sometimes equated to reaction rates or empirical flux expressions at solid walls (see, for example, Himmelblau and Bischoff [1968]).

$$\mathbf{J}_i.\mathbf{n} = 0 \tag{2.43}$$

For viscous fluids, molecular arguments and experimental evidence also give rise to the no-slip condition at solid walls, which states that the tangential components of the fluid velocity at the wall are equal to the tangential components of the wall velocity (2.44).

$$\mathbf{v} \times \mathbf{n} = \mathbf{v}_w \times \mathbf{n} \tag{2.44}$$

Again if the wall is at rest no-slip reduces to

$$\mathbf{v} \times \mathbf{n} = 0 \tag{2.45}$$

Boundary conditions on energy at a solid wall are based on the principles of conservation and thermodynamic equilibrium [Richardson, 1989]. For the former, in analogy to no flow-through, conservation arguments lead to conditions on the normal component of the energy flux at the wall, equating it to the normal component of the energy flux of the wall (2.46).

$$\mathbf{q}.\mathbf{n} = \mathbf{q}_w.\mathbf{n} \tag{2.46}$$

Of course, if the wall is insulated then there is no heat transfer through it (2.47).

$$\mathbf{q}.\mathbf{n} = 0 \tag{2.47}$$

The second notion of thermodynamic equilibrium implies that the fluid at the wall is at the same temperature as the wall itself (2.48)

$$T = T_w \tag{2.48}$$

2.3.2 Open Fluid Interfaces

An open or free fluid interface defines a boundary of a flow domain across which the enclosed fluid phase (or phases) is continuous. It may correspond to a physical boundary, such as the end of a pipe, or it may define an imaginary boundary which delimits a portion of a domain as deemed appropriate in a given application.

2.3.2.1 Single Domains

In studies of fluid mechanics (both traditional and computational), flow domains are normally considered in isolation. Consequently, at inflow boundaries, the spatial distributions of density (in compressible flows), temperature and velocity (or an equivalent set of free variables such as pressure, enthalpy and velocity) must be known or assumed over the

extent of the boundary. Therefore, inflow conditions are usually given in terms of simple equality relations (2.49)

$$x = x_0 \tag{2.49}$$

where x_0 has some given (or assumed) spatial distribution over the boundary.

At outflow boundaries, since no details are known of the flow downstream, a common approach is to use a specified pressure across the entire boundary and set the normal gradients of all other variables to zero (2.50). For scalar quantities (2.50) is often known as the Danckwerts exit condition after Danckwerts [1953] (see section 2.5.1).

$$\frac{\partial x}{\partial n} = 0 \tag{2.50}$$

In CFD applications the positions of outflow boundaries (in particular) must (often) be chosen carefully to ensure the simulation results do not become a function of their location [Versteeg and Malalasekera, 1995]. Placing an outflow boundary in a position where the flow field is still developing, for example, would lead to incorrect solutions.

2.3.2.2 Interfaces Between Domains

In more traditional modelling applications, such as chemical reaction engineering or process synthesis, it is normal for models to consist of a number of interconnected flow domains. An example is given in figure 2.3 which shows an imaginary (in the sense that it is simply an artifact of the model) open fluid interface between two pipes modelled in separate flow domains.

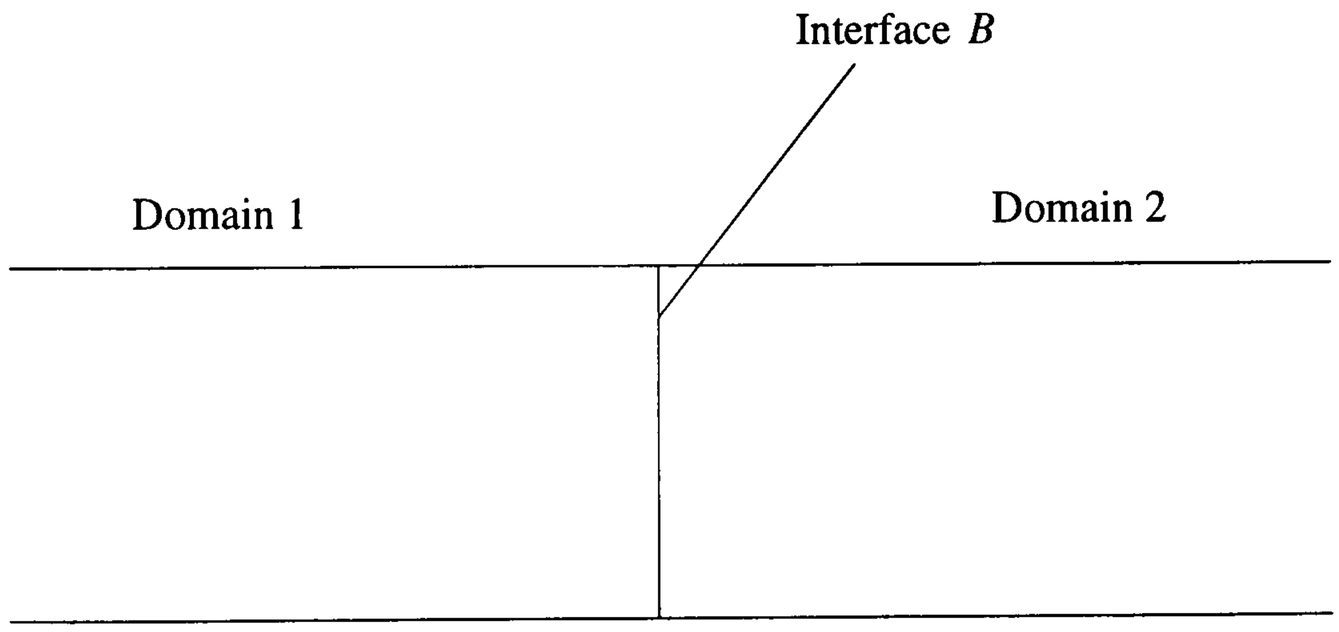


Figure 2.3: An Open Fluid Interface between Two Flow Domains

In such cases, continuity and conservation arguments lead to conditions expressing equality of variables (2.51) and normal gradients (2.52) across the interface, providing there are no discontinuities in the flux term proportionality constants (viscosity, dispersion

coefficient and thermal conductivity)[7].

$$x(B^-) = x(B^+) \tag{2.51}$$

$$\left.\frac{\partial x}{\partial n}\right|_{(B^-)} = \left.\frac{\partial x}{\partial n}\right|_{(B^+)} \tag{2.52}$$

2.3.3 Symmetry Boundaries

In many physical situations, geometrical symmetry in the flow domain is mirrored in the flow itself. Such features lead to boundary conditions which are kinematic in origin. Three cases are of particular importance: symmetry about a point, symmetry about a line (or axis) and symmetry about a plane.

In all three cases the appropriate boundary conditions are based on the principles of conservation and smoothness [Richardson, 1989]. Conservation requires that at a point, line or plane of symmetry the components of the velocity normal to that point, line or plane disappear (2.53), because otherwise an unbounded (positive or negative) accumulation of fluid would occur there.

$$\mathbf{v}.\mathbf{n} = 0 \tag{2.53}$$

The second notion of smoothness dictates that at a point, line or plane of symmetry the normal gradients of both the velocity components tangential to that point, line or plane (2.54) and the scalar dependent variables (2.55) must be zero, because otherwise discontinuities in the normal gradients would exist there.

$$\frac{\partial v_n}{\partial n} = 0 \tag{2.54}$$

$$\frac{\partial \phi}{\partial n} = 0 \tag{2.55}$$

The presence of symmetry boundaries in a flow problem has the effect of reducing the size of the solution domain. For example, in axisymmetric pipe flows it is only necessary to consider the domain from the centreline to the pipe wall (*i.e.* from 0 to R), rather than from one pipe wall to the other (*i.e.* from R^- to R^+).

2.3.4 Cyclic Boundaries

Cyclic or periodic boundaries are used in problems with symmetry in the flow domain and over which the flow is repeated. Figure 2.2(b) gives an example of a physical situation

[7]Boundary conditions for cases where modelling assumptions give rise to discontinuities in proportionality constants are discussed in section 2.5.1.

where cyclic boundary conditions are applied. This shows an end-on view of a burner nozzle taken from Versteeg and Malalasekera [1995]. Here the k boundaries are oriented in the r-z plane (pointing out of the page and orthogonal to θ). In this case the problem can be solved considering only the section of the domain between the boundaries $k = 1$ and $k = N$. The flow in this region is repeated over the other sections that make up the nozzle.

Cyclic boundary conditions are based on the notion that the flow is repeated and express the equality of variables (2.56) and normal gradients (2.57) across the cyclic boundary.

$$x(k = 1) = x(k = N) \tag{2.56}$$

$$\left.\frac{\partial x}{\partial n}\right|_{(k=1)} = \left.\frac{\partial x}{\partial n}\right|_{(k=N)} \tag{2.57}$$

As with symmetry boundaries, the effect of cyclic boundaries is to reduce the size of the solution domain.

2.4 Simplified Models

For many engineering applications the full three dimensional conservation equations (sections 2.1 and 2.2) are too complex and/or contain more detail than is needed for practical purposes and simplified model equations are desirable. Indeed, in many cases, several models of differing mathematical form and complexity are required for the same process system, tailored to suit their intended use (Stephanopoulos *et al.* [1990] discuss as an example the need for models of varying detail and refinement during the development of a process design). From a theoretical point of view, the difference between any two models representing the same process system lies in the assumptions made in their derivation. This implies that all models of a particular process system may be considered as simplifications of a more general model and provides the basis of a consistent methodology for their derivation. This is illustrated below where the derivation of models of reduced dimensionality from the general three dimensional equations given above is considered together with a brief discussion of other common modelling assumptions.

2.4.1 Models of Reduced Dimensionality

One of the most common simplifications used in process modelling is that of reduced spatial dependency or reduced dimensionality. Formal definitions of models of reduced dimensionality are rarely given in either fluid mechanics or process modelling texts. Whitaker [1968] defines a one dimensional flow as one where the velocity vector is a function of only one spatial coordinate. Extending this definition, an n dimensional model (where

$n = 0, 1, 2, 3$) may be defined as one where the model variables are functions only of those n spatial coordinates. It is common to further assume (often implicitly) that the only nonzero components of the velocity vector are also in those n coordinates, although this is not always the case [Richardson, 1989]. For the case where $n = 0$, the model is described as macroscopic or lumped and its variables exhibit no spatial variation over the defined control volume. The derivation of the reduced model equations ($n = 0, 1, 2$) from the general three dimensional (mass-averaged) equations ($n = 3$) given in sections 2.1 and 2.2 is considered below. Note that the averaging overbars and tildes over the various symbols indicating time, ensemble or mass averages are dropped in these derivations to simplify the nomenclature.

2.4.1.1 Two Dimensional Models

Mathematically, it is possible to derive a two dimensional model from a full three dimensional description by spatially averaging over one of the three coordinates. For example, a two dimensional model of a process system exhibiting three dimensional spatial variation of its properties over the coordinates z_1, z_2, z_3 can be derived by averaging over (say) z_1 and approximating the spatial gradients of each quantity with respect to z_1 in terms of the values of those quantities at the endpoints of the z_1 domain.

This can be illustrated simply by considering the derivation of a general two dimensional continuity equation in cartesian coordinates. The procedure is straightforward and involves integrating the full three dimensional equation (2.58) over the spatial coordinate z_1

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial z_1}\rho v_{z_1} + \frac{\partial}{\partial z_2}\rho v_{z_2} + \frac{\partial}{\partial z_3}\rho v_{z_3} = 0 \tag{2.58}$$

to give

$$\int_{z_1} \frac{\partial}{\partial t}\rho \, dz_1 + [\rho v_{z_1}]_{z_1} + \int_{z_1} \frac{\partial}{\partial z_2}\rho v_{z_2} \, dz_1 + \int_{z_1} \frac{\partial}{\partial z_3}\rho v_{z_3} \, dz_1 = 0 \tag{2.59}$$

The spatial average of any variable x over z_1 is defined by

$$\langle x \rangle = \frac{\int_{z_1} x \, dx_1}{\int_{z_1} dz_1} = \frac{1}{L_{z_1}} \int_{z_1} x \, dx_1 \tag{2.60}$$

which, upon substitution in (2.59), gives

$$\langle \frac{\partial}{\partial t}\rho \rangle + \left[\frac{\rho v_{z_1}}{L_{z_1}}\right]_{z_1} + \langle \frac{\partial}{\partial z_2}\rho v_{z_2} \rangle + \langle \frac{\partial}{\partial z_3}\rho v_{z_3} \rangle = 0 \tag{2.61}$$

The integral terms in (2.59) cannot be evaluated exactly unless their distribution over z_1 is known, hence the appearance of averages of complete terms in (2.61). To calculate these spatial distributions would require the solution of (2.58) over z_1, which clearly defeats the

objective of deriving a two dimensional formulation. Instead, the normal approximation is

$$\langle \frac{\partial}{\partial z_2} \rho v_{z_2} \rangle \approx \frac{\partial}{\partial z_2} \langle \rho \rangle \langle v_{z_2} \rangle \tag{2.62}$$

(which is tantamount to assuming that the spatial distributions over z_1 are uniform) so that the integral terms in (2.61), the averages of products, can be replaced by the corresponding products of averages. Making these substitutions in (2.61) results in the general two dimensional equation form shown below (2.63).

$$\frac{\partial}{\partial t} \langle \rho \rangle + \left[\frac{\rho v_{z_1}}{L_{z_1}} \right]_{z_1} + \frac{\partial}{\partial z_2} \langle \rho \rangle \langle v_{z_2} \rangle + \frac{\partial}{\partial z_3} \langle \rho \rangle \langle v_{z_3} \rangle = 0 \tag{2.63}$$

It can be seen that only spatial variation of the (z_1-averaged) mass flux with respect to the z_2 and z_3 coordinates is described explicitly in (2.63) but the overall contribution of the z_1 variation is retained in terms of the difference between its values at the endpoints of the z_1 domain. As with other averaged models, the overall contribution of any distributed information lost during averaging (as a result of 2.62) must be resupplied in the closure laws.

In practice, two dimensional descriptions are usually used in cases where the gradients (and hence the endpoint terms) and velocity component in the averaged coordinate are (assumed to be) negligible (*cf.* 2.64).

$$\frac{\partial}{\partial t} \langle \rho \rangle + \frac{\partial}{\partial z_2} \langle \rho \rangle \langle v_{z_2} \rangle + \frac{\partial}{\partial z_3} \langle \rho \rangle \langle v_{z_3} \rangle = 0 \tag{2.64}$$

Typical examples include planar flows in very wide rectangular channels where the gradients and velocity component in the "wide" direction are ignored or axisymmetric pipe (-like) flows where there are (assumed to be) no angular gradients or angular velocity component (see, for example, Richardson [1989]). For the latter, single phase two dimensional equations can be derived from the general Favre-average equations given above (2.24 - 2.26) simply by setting the angular gradients ($\partial x / \partial \theta$) and velocity component (v_θ) to zero. The results are given below (2.65 - 2.68).

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial z} \rho v_z + \frac{1}{r} \frac{\partial}{\partial r} r \rho v_r = 0 \tag{2.65}$$

$$\frac{\partial}{\partial t} \rho v_z + \frac{\partial}{\partial z} \rho v_z v_z + \frac{1}{r} \frac{\partial}{\partial r} r \rho v_r v_z + \frac{\partial p}{\partial z} - \frac{\partial}{\partial z} \tau^e_{zz} - \frac{1}{r} \frac{\partial}{\partial r} r \tau^e_{rz} - \rho g_z = 0 \tag{2.66}$$

$$\frac{\partial}{\partial t} \rho v_r + \frac{\partial}{\partial z} \rho v_z v_r + \frac{1}{r} \frac{\partial}{\partial r} r \rho v_r v_r + \frac{\partial p}{\partial r} - \frac{\partial}{\partial z} \tau^e_{rz} - \frac{1}{r} \frac{\partial}{\partial r} r \tau^e_{rr} + \frac{\tau^e_{\theta\theta}}{r} = 0 \tag{2.67}$$

$$\frac{\partial}{\partial t}\rho\hat{E} + \frac{\partial}{\partial z}\rho v_z\hat{H} + \frac{1}{r}\frac{\partial}{\partial r}r\rho v_r\hat{H} + \frac{\partial}{\partial}q_z^e + \frac{1}{r}\frac{\partial}{\partial r}rq_r^e$$

$$-\frac{\partial}{\partial z}(\tau_{zz}^e v_z + \tau_{rz}^e v_r) - \frac{1}{r}\frac{\partial}{\partial r}r(\tau_{rz}^e v_z + \tau_{rr}^e v_r) - \rho v_z g_z = 0 \tag{2.68}$$

For axisymmetric multiphase pipe flows, analogous two dimensional differential equations and jump conditions can be derived in a similar manner starting from an appropriate averaged multiphase formulation.

2.4.1.2 One Dimensional Models

The derivation of a general one dimensional model involves integrating the full three dimensional equations over an arbitrary control volume and taking the limit as one spatial coordinate of that control volume tends to zero. Consider the (arbitrary) fixed cylindrical volume shown in figure 2.4. The general prototype balance equation can be written

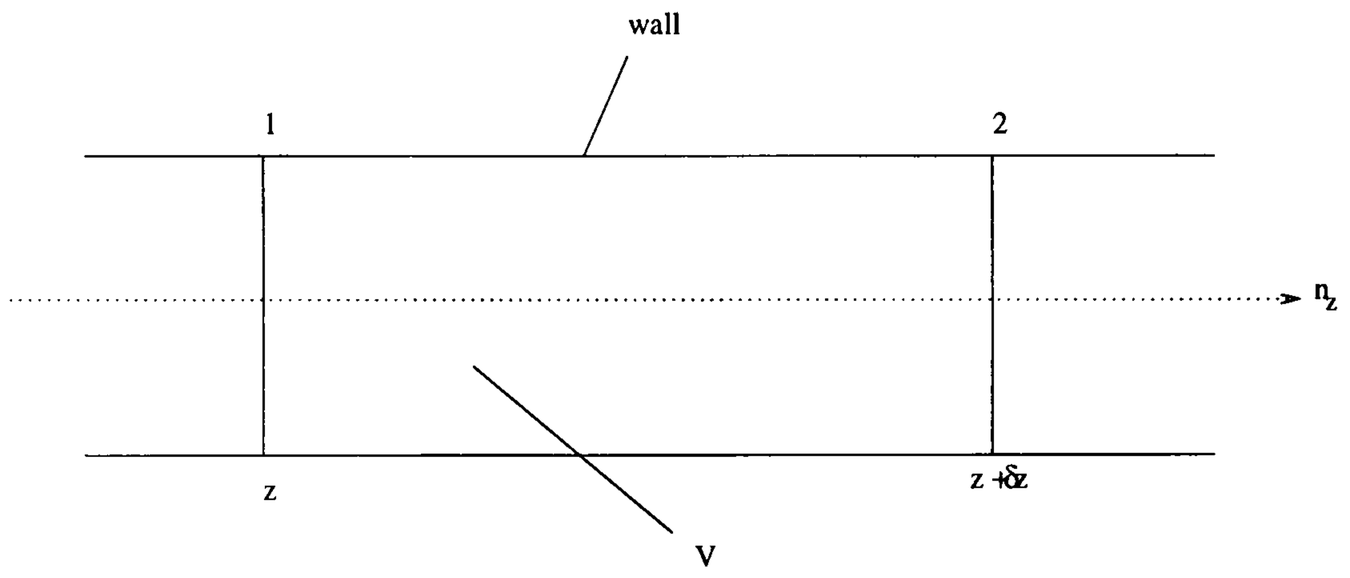


Figure 2.4: Control Volume cut by two Cross-section Planes

$$\frac{\partial}{\partial t}\rho\psi + \nabla.(\rho\psi\mathbf{v}) + \nabla.\mathbf{J} - \rho\chi = 0 \tag{2.69}$$

where the individual mass, momentum and energy equations (2.25 - 2.26) are obtained by substituting the appropriate values of ψ, $\mathbf{J}$ and χ from table 2.2.

	ψ	$\mathbf{J}$	χ
Mass Conservation	1	0	0
Momentum Conservation	$\mathbf{v}$	$-\mathbf{T}$	$\mathbf{F}$
Total Energy Conservation	$\hat{U} + \frac{1}{2}\mathbf{v}.\mathbf{v}$	$\mathbf{q} - \mathbf{T}.\mathbf{v}$	$\mathbf{F}.\mathbf{v}$

Table 2.2: General Balance Equation

Integrating over the fixed cylindrical volume V

$$\int_V \frac{\partial}{\partial t}\rho\psi \, dV + \int_V \nabla.(\rho\psi\mathbf{v}) \, dV + \int_V \nabla.\mathbf{J} \, dV - \int_V \rho\chi \, dV = 0 \tag{2.70}$$

Substituting for the integral terms using the Leibnitz rule (2.71) and Gauss-Ostrogradskii divergence theorem (2.72)

$$\int_{V(t)} \frac{\partial \rho}{\partial t} \, dV = \frac{d}{dt}\int_{V(t)} \rho \, dV - \int_{S(t)} \rho(\mathbf{v_s}.\mathbf{n}) \, dS \tag{2.71}$$

$$\int_{V(t)} \nabla.(\rho\mathbf{v}) \, dV = \int_{S(t)} \rho\mathbf{v}.\mathbf{n} \, dS \tag{2.72}$$

(where in this case $\mathbf{v_s} = 0$ and $S(t)$ $(= S)$ includes both the cross-sectional planes at z and $z + \delta z$ and also the cylindrical wall) gives

$$\frac{\partial}{\partial t}\int_V \rho\psi \, dV + \int_{A_2} \rho\psi\mathbf{v}.\mathbf{n} \, dA_2 - \int_{A_1} \rho\psi\mathbf{v}.\mathbf{n} \, dA_1$$

$$+ \int_{A_2} \mathbf{J}.\mathbf{n} \, dA_2 - \int_{A_1} \mathbf{J}.\mathbf{n} \, dA_1 + \int_{A_w} \mathbf{J}.\mathbf{n}_w \, dA_w - \int_V \rho\chi \, dV = 0 \tag{2.73}$$

where $\mathbf{n}$ is the unit normal in the z direction and $\mathbf{n}_w$ the unit vector normal to the wall. The volume average of any variable x is defined as

$$\langle x \rangle_V = \frac{1}{V}\int_V x \, dV \tag{2.74}$$

Substituting from (2.74), noting $V = A\,\delta z$, rearranging and taking the limit as $\delta z \to 0$

$$\lim_{\delta z \to 0}\left\{\frac{\partial}{\partial t}\langle\rho\psi\rangle_V A\right\} + \lim_{\delta z \to 0}\left\{\frac{\int_{A_2} \rho\psi\mathbf{v}.\mathbf{n} \, dA_2 - \int_{A_1} \rho\psi\mathbf{v}.\mathbf{n} \, dA_1}{\delta z}\right\}$$

$$+ \lim_{\delta z \to 0}\left\{\frac{\int_{A_2} \mathbf{J}.\mathbf{n} \, dA_2 - \int_{A_1} \mathbf{J}.\mathbf{n} \, dA_1}{\delta z}\right\} + \lim_{\delta z \to 0}\left\{\frac{1}{\delta z}\int_{A_w} \mathbf{J}.\mathbf{n}_w \, dA_w\right\} \tag{2.75}$$

$$- \lim_{\delta z \to 0}\{\langle\rho\chi\rangle_V A\} = 0$$

gives

$$\frac{\partial}{\partial t}\langle\rho\psi\rangle_V A + \frac{\partial}{\partial z}\left\{\int_A \rho\psi\mathbf{v}.\mathbf{n} \, dA\right\} + \frac{\partial}{\partial z}\left\{\int_A \mathbf{J}.\mathbf{n} \, dA\right\}$$

$$+ \lim_{\delta z \to 0}\left\{\frac{1}{\delta z}\int_{A_w} \mathbf{J}.\mathbf{n}_w \, dA_w\right\} - \langle\rho\chi\rangle_V A = 0 \tag{2.76}$$

Defining cross-sectional area averages as

$$\langle x \rangle_A = \frac{1}{A}\int_A x \, dA \tag{2.77}$$

and wall area averages as

$$\langle x \rangle_w = \frac{1}{A_w}\int_{A_w} x \, dA_w \tag{2.78}$$

and substituting, noting that in the limit the volume and cross-sectional area averages become identical, gives

$$\frac{\partial}{\partial t}\langle\rho\psi\rangle_A A + \frac{\partial}{\partial z}\langle\rho\psi\mathbf{v}.\mathbf{n}\rangle_A A + \frac{\partial}{\partial z}\langle\mathbf{J}.\mathbf{n}\rangle_A A + \lim_{\delta z\to 0}\left\{\frac{1}{\delta z}\langle\mathbf{J}.\mathbf{n}_w\rangle A_w\right\} - \langle\rho\chi\rangle_A A = 0 \tag{2.79}$$

Since the spatial distributions of the average of products of the variables over the cross-sectional and wall areas are unknown, there is again little choice but to assume

$$\langle x_1 x_2 \ldots\rangle = \langle x_1\rangle\langle x_2\rangle \ldots \tag{2.80}$$

giving

$$\frac{\partial}{\partial t}\langle\rho\rangle\langle\psi\rangle A + \frac{\partial}{\partial z}\langle\rho\rangle\langle\psi\rangle\langle\mathbf{v}.\mathbf{n}\rangle A + \frac{\partial}{\partial z}\langle\mathbf{J}.\mathbf{n}\rangle A + \lim_{\delta z\to 0}\left\{\frac{1}{\delta z}\langle\mathbf{J}.\mathbf{n}_w\rangle_w A_w\right\} - \langle\rho\rangle\langle\chi\rangle A = 0 \tag{2.81}$$

where the angle brackets now denote cross-sectional area averages unless otherwise indicated. The second last term in (2.81) represents the flux of **J** through the wall. In the limit it may be written as

$$\lim_{\delta z\to 0}\left\{\frac{1}{\delta z}\langle\mathbf{J}.\mathbf{n}_w\rangle_w A_w\right\} = P_w\langle\mathbf{J}.\mathbf{n}_w\rangle_w A_w \tag{2.82}$$

Substituting and simplifying for ducts of constant cross-section gives the general form of one dimensional balance equation (2.83),

$$\frac{\partial}{\partial t}\langle\rho\rangle\langle\psi\rangle + \frac{\partial}{\partial z}\langle\rho\rangle\langle\psi\rangle\langle\mathbf{v}.\mathbf{n}\rangle + \frac{\partial}{\partial z}\langle\mathbf{J}.\mathbf{n}\rangle + \frac{\langle\mathbf{J}.\mathbf{n}_w\rangle_w P_w}{A} - \langle\rho\rangle\langle\chi\rangle = 0 \tag{2.83}$$

where the generic one dimensional mass, momentum and energy balance equations (2.84 - 2.86)[8] are obtained by substituting from table 2.2.

$$\frac{\partial\rho}{\partial t} + \frac{\partial}{\partial z}\rho v_z = 0 \tag{2.84}$$

$$\frac{\partial}{\partial t}\rho\mathbf{v} + \frac{\partial}{\partial z}\rho\mathbf{v}v_z + \frac{\partial}{\partial z}p\mathbf{n} - \frac{\partial}{\partial z}\tau_e.\mathbf{n} - \frac{\mathbf{T}.\mathbf{n}_w P_w}{A} - \rho\mathbf{g} = 0 \tag{2.85}$$

$$\frac{\partial}{\partial t}\rho\hat{E} + \frac{\partial}{\partial z}\rho\hat{H}v_z + \frac{\partial}{\partial z}q_z^e + \frac{\mathbf{q}.\mathbf{n}_w P_w}{A} - \frac{\partial}{\partial z}(\tau.\mathbf{n}).\mathbf{v} - \rho v_z g_z = 0 \tag{2.86}$$

Note that although these equations only explicitly describe axial variation of the system properties, the radial and angular gradients are retained in the closure laws and approximations used for the wall terms. Notice also that in the form given above the generic

[8] Angled brackets indicating spatial averages are omitted for conciseness.

momentum balance (2.85) is a vector equation with three orthogonal components and as such is valid for more general situations than that shown in figure 2.4. However, in this case, the r and θ components are identically zero so that (2.85) and (2.86) reduce directly to (the scalar) (2.87) and (2.88) respectively.

$$\frac{\partial}{\partial t}\rho v_z + \frac{\partial}{\partial z}\rho v_z v_z + \frac{\partial}{\partial z}p - \frac{\partial}{\partial z}\tau^e_{zz} + \frac{\tau_w P_w}{A} - \rho g_z = 0 \tag{2.87}$$

$$\frac{\partial}{\partial t}\rho \hat{E} + \frac{\partial}{\partial z}\rho \hat{H} v_z + \frac{\partial}{\partial z}q^e_z - \frac{Q_w P_w}{A} - \frac{\partial}{\partial z}\tau^e_{zz} v_z - \rho v_z g_z = 0 \tag{2.88}$$

Note that, following Bird *et al.* [1960] and Whitaker [1968], the sign of the wall flux terms ($\tau_w P_w/A$ and $Q_w P_w/A$) in (2.87) and (2.88) is reversed from that in (2.85) and (2.86) to define the frictional force in terms of the drag by the wall and the wall heat transfer in terms of energy added to the fluid (for consistency with the sign convention usually used in thermodynamics).

For multiphase systems an analogous derivation can be performed starting from an appropriate time or ensemble averaged three dimensional formulation (*cf.* 2.34 - 2.40). In this case the situation is complicated by the existence of the multiphase mixture and its associated single phase regions and interfaces within the control volume V. Following the same procedure as for single phase systems but using a generalised form of the Leibnitz rule and divergence theorem leads to the composite (space and time/ensemble) averaged balance equations and jump conditions (2.89 and 2.90) given below [Roberts, 1997].

$$\frac{\partial}{\partial t}\epsilon_k \rho_k \psi_k A + \frac{\partial}{\partial z}\epsilon_k \rho_k \psi_k \mathbf{v}_k.\mathbf{n} A + \frac{\partial}{\partial z}\epsilon_k \mathbf{J}_k.\mathbf{n} A +$$

$$\lim_{\delta z \to 0} \frac{1}{\delta z}\left\{\int_{A_I} (m_k \psi_k + \mathbf{J}_k.\mathbf{n}_k)\, dA_I + \int_{A_k w} \mathbf{J}_k.\mathbf{n}_k\, dA_{kw}\right\} - \epsilon \rho_k \chi_k A = 0 \tag{2.89}$$

$$\sum_k \left(\lim_{\delta z \to 0} \frac{1}{\delta z}\left\{\int_{A_I} m_k \psi_k\, dA_I + \int_{A_I} \mathbf{J}_k.\mathbf{n}_k\, dA_I\right\}\right) = 0 \tag{2.90}$$

Again, the individual mass, momentum and energy equations for the k-th phase are obtained by substituting the appropriate values of ψ_k, $\mathbf{J}_k$ and χ_k from table 2.2.

The formulation of appropriate closure relationships for the model represented by equations (2.89) and (2.90), which in this case also include equations for the limiting wall transfer terms, is again a considerable problem in practice.

2.4.1.3 Macroscopic Models

Macroscopic or lumped parameter mathematical models represent the simplest mathematical description of a process system based on continuum mechanics. In this approach all detail within a given system is ignored and as a consequence the governing mathematical equations involve no spatial gradients. A rigorous derivation of a macroscopic model

BIL. LONDN. UNIV.

involves integration of the governing single or multiphase three dimensional differential equations, which may be ensemble or time averaged (see sections 2.1 and 2.2.2), over the system volume. For single phase systems the results of this integration are as follows [Bird, 1957]

$$\frac{d}{dt}M_i = -\Delta w_i + R_i \qquad i = 1, 2, ..., n \tag{2.91}$$

$$\frac{d}{dt}\mathbf{P} = -\Delta\left(\frac{\langle v^2\rangle}{\langle v\rangle}\mathbf{w} + p\mathbf{S}\right) - \mathbf{F} + M\mathbf{g} \tag{2.92}$$

$$\frac{d}{dt}E = -\Delta\left[\left(\hat{U} + p\hat{V} + \frac{1}{2}\frac{\langle v^3\rangle}{\langle v\rangle} + \hat{\Phi}\right)w\right] + Q - W \tag{2.93}$$

Notice that the momentum equation (2.92) is again in vector form with three orthogonal components. However, in practical applications it is often replaced by a more convenient scalar form (2.94) suitable for the calculation of flow rates and pressure drops (but not force balances) using the assumption of quasi-steady state behaviour [Whitaker, 1968].

$$-\Delta\left(\frac{\langle v^2\rangle}{\langle v\rangle}w + pS\right) - F = 0 \tag{2.94}$$

In essence (2.94) forms an equivalent of the macroscopic mechanical energy balance but suitable for both single and multicomponent systems (see, for example, Bird [1957] or Whitaker [1968]) and in which errors arising from the scalar approximation are accounted for (implicitly) in the empirical equations for the wall friction (usually expressed in terms of friction factors).

For many engineering calculations it is common to make additional simplifying assumptions. These include setting various terms that are considered negligible to zero, replacing the holdup terms by the product of the system volume and a volume averaged intensity (*cf.* 2.95 for mass holdup)

$$M_i = \int_V \rho_i \, dV = \langle\rho_i\rangle_V V \tag{2.95}$$

and assuming flat velocity profiles (*cf.* 2.80) over the inlet and outlet surfaces so that

$$\frac{\langle v^3\rangle}{\langle v\rangle} \approx \langle v\rangle^2 \qquad\qquad \frac{\langle v^2\rangle}{\langle v\rangle} \approx \langle v\rangle \tag{2.96}$$

Note also that the macroscopic balance equations do not imply perfect mixing but merely relate the holdup of each conserved quantity to the fluxes across the open and closed system boundaries.

Rigorous derivations of macroscopic multiphase models are rarely used. However, it can be shown that such a derivation must be performed with care to obtain a

description that is consistent with the continuum hypothesis. An illustration of the complications that arise from the existence of internal interphase boundaries is given below for the case of a hypothetical two phase vessel (perhaps a flash unit) under two different operating conditions (figure 2.5). For the first case in figure 2.5 a macroscopic model (for simplicity considering only mass conservation) can be derived from the local instantaneous formulation by integration over the separate phase volumes[9]. Starting with the continuity equation for phase A

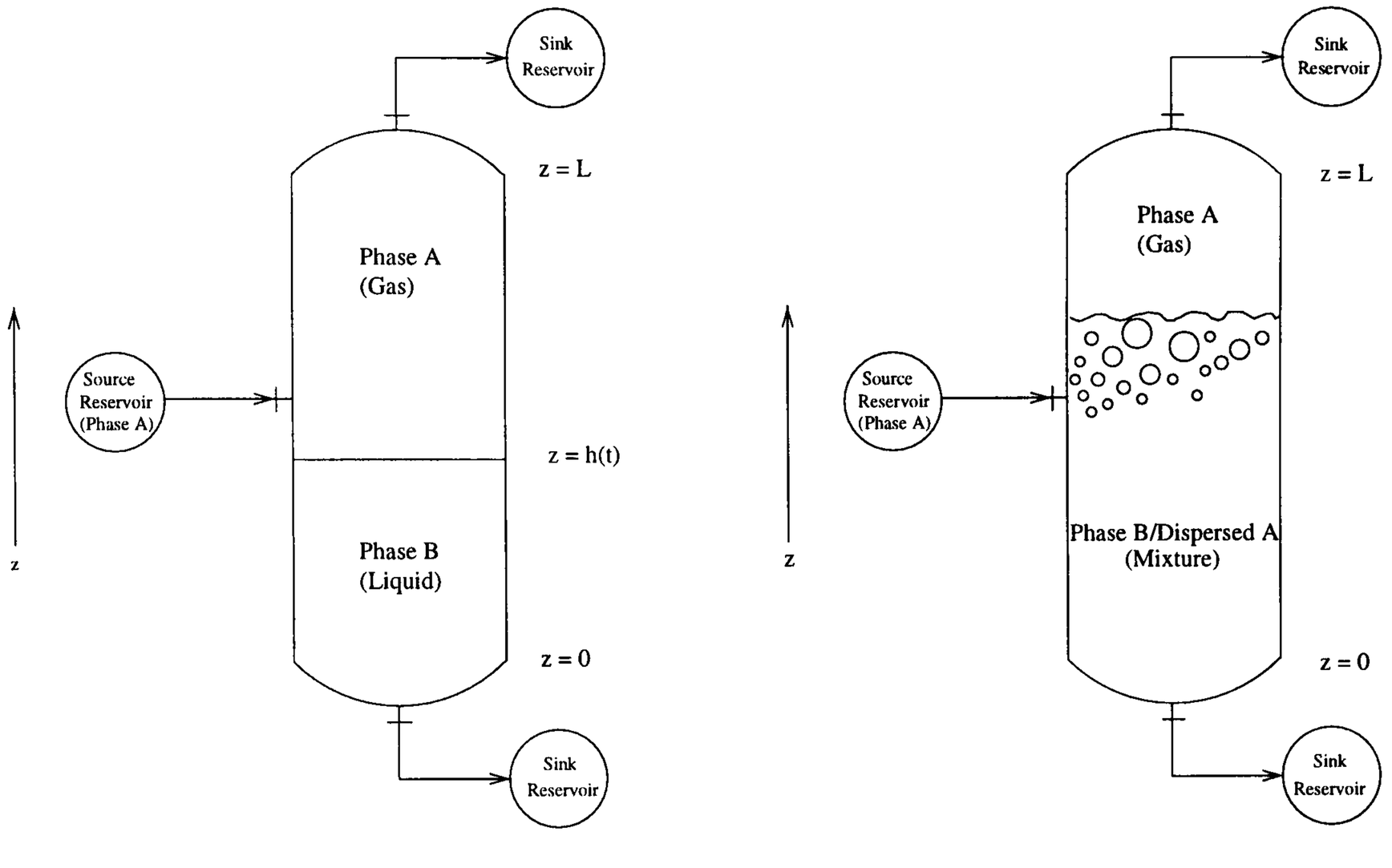


Figure 2.5: Hypothetical Two Phase Vessel

$$\frac{\partial}{\partial t}(\rho_A) + \nabla.(\rho_A \mathbf{v}_A) = 0 \tag{2.97}$$

and integrating over the (variable) volume of phase A

$$\int_{V(t)} \frac{\partial \rho_A}{\partial t}\, dV + \int_{V(t)} \nabla.(\rho_A \mathbf{v}_A)\, dV = 0 \tag{2.98}$$

Substituting for the two integral terms using the Leibnitz formula (2.71) and Gauss-Ostrogradskii divergence theorem (2.72) gives

$$\frac{d}{dt}\int_{V(t)} \rho_A\, dV = -\int_{V(t)} \nabla.\left(\rho_A(\mathbf{v}_A - \mathbf{v}_s)\right)\, dV \tag{2.99}$$

Using a one dimensional description of phase A for a vessel of constant cross-sectional area this can be written

[9] A similar derivation for a heterogeneous system contained within an arbitrary volume has been given by Standart [1964].

$$\frac{d}{dt}\int_{h(t)}^{L} A_x \rho_A \, dz = -\int_{h(t)}^{L} \frac{\partial}{\partial z}\left(A_x \rho_A (v_{zA} - v_{zs})\right) \, dz \tag{2.100}$$

giving

$$\frac{d}{dt} M_A = A_x \rho_A (v_{zA} - v_{zs})|_{h(t)} - A_x \rho_A (v_{zA} - v_{zs})|_L \tag{2.101}$$

or

$$\frac{d}{dt} M_A = \Upsilon_A - w_A \tag{2.102}$$

where

$$\Upsilon_A = A_x m_A = A_x \rho_A (v_{zA} - v_{zs})|_{h(t)} \tag{2.103}$$

and

$$w_A = A_x \rho_A (v_{zA} - v_{zs})|_L \tag{2.104}$$

where the directionality of w_A is from vessel to reservoir and Υ_A is interphase transfer to A. Similarly for phase B

$$\frac{d}{dt} M_B = \Upsilon_B - w_B \tag{2.105}$$

The sum of (2.102) and (2.105) gives the total mass balance for the vessel (2.106) since, from the jump condition, $\sum_{k=A}^{B} m_k = 0$.

$$\frac{d}{dt} M + w_A + w_B = 0 \tag{2.106}$$

Furthermore, if phase B is assumed incompressible (*i.e.* a liquid phase) a relationship between the total mass holdup of B and the liquid level can be found (for a constant cross-sectional area) from

$$M_B = \int_0^{h(t)} A_x \rho_B \, dz = \rho_B V_B \tag{2.107}$$

where

$$V_B = \int_0^{h(t)} A_x \, dz \tag{2.108}$$

This approach is possible because the phases in the vessel are separated into clearly defined continuous regions where the location of the interface can be specified (in terms of $h(t)$).

However, if the level of phase B in the vessel rises, as in case 2 (figure 2.5), so that phase A becomes dispersed in phase B, such an approach is not possible because the location of the interfaces in the two phase mixture is not known *a priori*. Consequently

the local instantaneous equations cannot be integrated in the normal sense without violating the continuum hypothesis. A rigorous macroscopic model can only be derived by integrating an averaged three dimensional model over the volume of the vessel. Recall that in this description phases A and B are considered as being in two continua simultaneously existing at the same point with a statistically determined local void fraction (see section 2.2.2). For case 2 the equivalent derivation therefore involves integrating the one dimensional form of the averaged mass conservation equation (*cf.* 2.89) over the total (fixed) height of the vessel, *viz.*

$$\int_0^L \frac{\partial}{\partial t}\left(A_x \epsilon_A \rho_A\right)\, dz + \int_0^L \frac{\partial}{\partial z}\left(A_x \epsilon_A \rho_A v_{zA}\right)\, dz = \int_0^L \left(A_x \Gamma_A\right)\, dz \tag{2.109}$$

giving

$$\int_0^L \frac{\partial}{\partial t}\left(A_x \epsilon_A \rho_A\right)\, dz = A_x \epsilon_A \rho_A v_{zA}|_0 - A_x \epsilon_A \rho_A v_{zA}|_L + \Upsilon_A \tag{2.110}$$

or

$$\frac{d}{dt} M_A = w_{in,A} - w_{out,A} + \Upsilon_A \tag{2.111}$$

where

$$\frac{d}{dt} M_A = \int_0^L \frac{\partial}{\partial t}\left(A_x \epsilon_A \rho_A\right)\, dz \tag{2.112}$$

and

$$w_{in,A} = A_x \epsilon_A \rho_A v_{zA}|_0 \tag{2.113}$$

$$w_{out,A} = A_x \epsilon_A \rho_A v_{zA}|_L \tag{2.114}$$

$$\Upsilon_A = \int_0^L \left(A_x \Gamma_A\right)\, dz \tag{2.115}$$

Notice that in this case (and in general) no relation can be found between the mixture level and holdup since the internal interfaces are not defined explicitly and the phases are considered as existing in two continua extending over the entire volume of the vessel.

In practice derivations of macroscopic models by integration are not widely used and instead they are commonly obtained by applying a statement of the form

net		net		net		net		net
accumulation =	net	transport	-	net	transport	+	net	generation -	net	consumption
in system		in through			out through			in system		in system
		system surfaces			system surfaces					

to extensive quantities over the entire volume of the process system (this is the approach normally given in undergraduate texts on process modelling, for example Luyben [1990]).

It is interesting to note that many process discontinuities, which have been a key focus of recent work in computer-aided process modelling (for example, Barton [1992], Pantelides and Barton [1993], Pantelides [1995] and Cho [1997]) arise in fact from moving boundary problems based on the local instantaneous formulation.

2.4.2 Modelling Assumptions

In addition to the model dimensionality, further simplifications are often possible for given situations. These assumptions typically describe some aspect(s) of the operation or physico-chemical behaviour of the process system under consideration and serve to reduce the model equations to a more simplified mathematical form. Common examples include the assumption of constant physical properties (such as incompressible fluids), assumptions regarding the operating conditions (for example, steady state behaviour or isothermal operation), simplifications to the flow regime (such as plug flow) and the omission of terms or transport mechanisms that are considered negligible (such as radiative heat transfer or viscous work terms). Further examples can be found in nearly all texts on process modelling (see, for example, Bird *et al.* [1960], Himmelblau and Bischoff [1968] or Luyben [1990]). Note that valid assumptions for a given situation cannot be stated generally and typically depend on both the nature of the system under consideration and also the modelling application. Assumptions may also have different meanings depending on the context. For example, the assumption of isothermal behaviour can be viewed in the sense of constant temperature such that energy conservation need not be considered or in a perfect control sense where the aim is to calculate the heat flux required to maintain a constant temperature. Finally, it is important to recognise that simplifications and assumptions impose limitations on the model that must be considered when evaluating its predicted results.

2.5 Connections Between Simplified Models

The use of simplified models allows for the possibility of connections (and hence interfaces or boundaries) between two flow domains described by models of either different dimensionality or derived under different assumptions (or both). For the purposes of the model generation approach developed in this thesis two cases are of particular importance: discontinuities in mixing characteristics across open fluid interfaces and boundaries across which the dimensionality of the model equations changes. The formulation of appropriate boundary conditions and approximations for each case is considered below.

2.5.1 Discontinuities in Mixing Characteristics Across Open Interfaces

Simplifications to the flow regime and/or model dimensionality lead to the possibility of open fluid boundaries across which the flux term proportionality constants (dispersion coefficient, thermal conductivity and viscosity) exhibit discontinuities. The most common examples occur in models of packed bed tubular reactors where the dispersion coefficient is assumed to vary discontinuously across the entrance and exit regions (see, for example, Froment and Bischoff [1990]).

The theoretical consideration of appropriate boundary conditions for flows with discontinuities in mixing characteristics is usually based on the analysis of reactor models. In a seminal paper, Danckwerts [1953] put forward (now famous) boundary conditions for flows with discontinuities in dispersion coefficient. He considered the case of constant density, steady-state one dimensional dispersed flow in a tubular reactor with plug flow upstream and downstream and used physical arguments to establish the conditions for mass conservation. More recently, Barber *et al.* [1998] used a rigorous mathematical analysis to deduce boundary conditions for mass conservation in more general situations with all combinations of zero, finite and infinite dispersion upstream and downstream of an open fluid interface B (as in figure 2.3). Their results show that mass balance dictates continuity of total molar flux across an open fluid interface (2.116) (although the dispersive flux is undefined if the dispersion coefficient is infinite)

$$N_i(B^-) = N_i(B^+) \tag{2.116}$$

and that, in addition, the concentration is continuous (2.117) in all cases except where plug flow occurs upstream of the interface

$$c_i(B^-) = c_i(B^+) \tag{2.117}$$

Continuity of the dispersive flux (2.118) also follows from (2.116) and (2.117)

$$J_i(B^-) = J_i(B^+) \tag{2.118}$$

which can in turn give a relation between concentration gradients when the upstream and downstream dispersion coefficients are finite and nonzero, and reduces to the Danckwerts exit condition (2.119) for plug flow downstream

$$J_i(B^-) = 0 \tag{2.119}$$

For the special case of plug flow upstream there is a discontinuity in concentration across the interface and (2.117) and (2.118) are no longer valid. However, the total molar flux is continuous as required by mass balance and thus from (2.116) the Danckwerts entrance condition (2.120) is recovered

$$v_z(B^-)c_i(B^-) = v_z(B^+)c_i(B^+) + J_i(B^+) \tag{2.120}$$

These results are based on a one dimensional analysis and are limited to linear cases (constant density mass and energy conservation). However, in the absence of a more complete analytical treatment it is not unreasonable to assume that they can be extended to include more general, nonlinear cases (compressible mass, energy and momentum conservation) and hence that similar conditions to above apply in general. Making this assumption, the boundary conditions for flows where the mixing characteristics exhibit discontinuities across an open fluid interface are in general given by

$$x(B^-) = x(B^+) \tag{2.121}$$

$$J_n(B^-) = J_n(B^+) \tag{2.122}$$

except where plug flow occurs upstream of the interface in which case

$$\rho v_n(B^-)x(B^-) = \rho v_n(B^+)x(B^+) + J_n(B^+) \tag{2.123}$$

where x represents component (molar or mass) density, enthalpy (and hence temperature in 2.121) or velocity and J_n the normal components of the corresponding (component) mass, energy and momentum fluxes.

In cases where modelling assumptions/simplifications lead to both discontinuities in the mixing characteristics and changes in the model dimensionality across an open fluid interface the boundary conditions represented by (2.121 - 2.123) are sometimes inconvenient and alternative conditions or approximations may be desirable (see below).

2.5.2 Model Dimensionality

The use of simplified models allows for the possibility of connections between models of different dimensionality and hence the existence of boundaries across which the dimensionality of the model equations (not to be confused with the physical dimensions) changes. This situation, which is clearly non-physical in origin, has important implications regarding the specification of boundary conditions. Consider the case of an interface between two control volumes where one is described by a two dimensional model and the other by a one dimensional model (figure 2.6).

If the problem is set up in such a way that the information propagates solely in the direction of decreasing dimensionality (from left to right in figure 2.6), the values of the model variables on the one dimensional side (*i.e.* at $z_1 = 0^+$) can be calculated simply by averaging their two dimensional counterparts over z_2 (2.124).

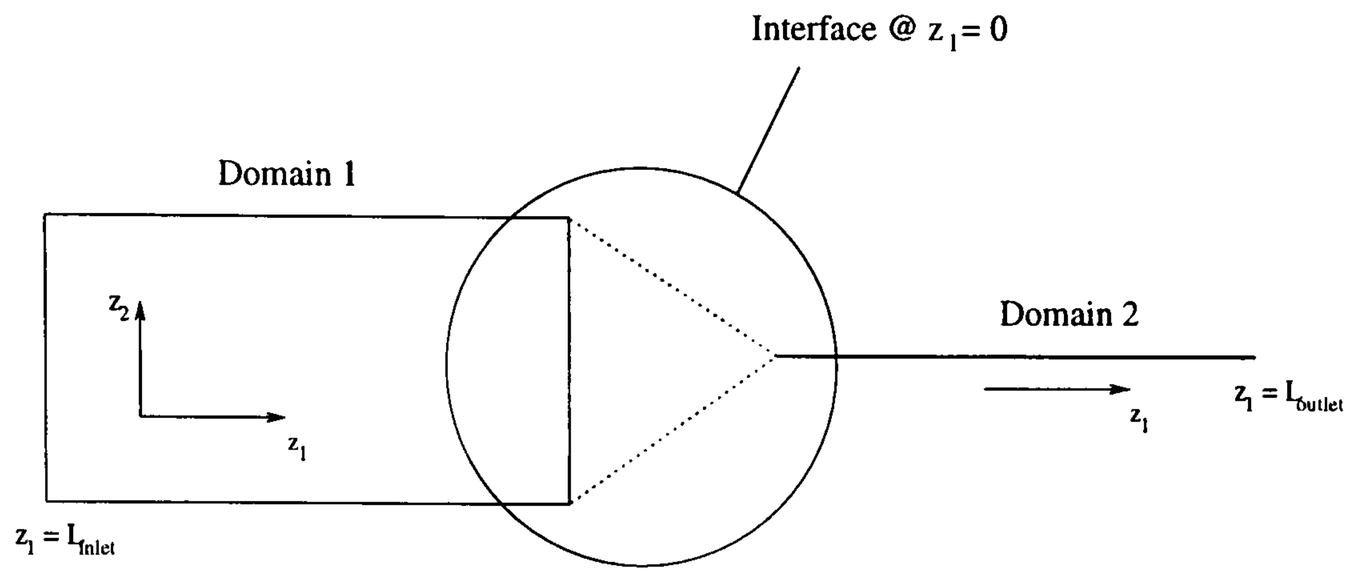


Figure 2.6: Interface Between Domains of Different Dimensionality

$$x(0^+) = \frac{\int_{z_2} x(0^-, z_2)\, dz_2}{\int_{z_2} dz_2} \qquad @\ z_1 = 0 \tag{2.124}$$

On the other hand, if the information propagates in the direction of increasing dimensionality (from right to left) only one dimensional information is available at the interface and this is insufficient to fully specify the corresponding information on the two dimensional side. Instead, at the interface, the spatial distributions of the two dimensional model variables over z_2 must be assumed, in a manner which is consistent with the conservation principles. The simplest method is to assume flat distributions (2.125).

$$x(0^-, z_2) = x(0^+) \qquad \forall\ z_2 \qquad @\ z_1 = 0 \tag{2.125}$$

In some physical situations, information can propagate in both directions. If the boundary shown in figure 2.6 represents an open fluid interface with bulk flow from left to right, it may be desirable to specify the boundary conditions for velocity upstream on the two dimensional side (at $z_1 = L_{\text{inlet}}$) and the pressure downstream on the one dimensional side (at $z_1 = L_{\text{outlet}}$). Consequently, at the interface, the equations matching the velocity variables can take the form of averages (*cf.* 2.124) whilst the pressure distribution over z_2 must be assumed (2.125). Flows with diffusion or dispersion also propagate information in both directions, meaning that to complete the boundary conditions (using the rules given in section 2.5.1) would require the formulation of some interface equations in terms of averaged quantities (*cf.* 2.124) whilst assuming flat distributions for others (*cf.* 2.125).

The above has important consequences in terms of model formulation since it implies that the boundary conditions depend not only on the physical situation but also on how the model is to be solved (for example, the specification of upstream versus downstream pressure boundary conditions). For the purposes of the model generation approach developed in this thesis, only unidirectional information propagation is supported. Two cases are important: open interfaces and closed interfaces.

For open interfaces the information is assumed to propagate only in the direction of bulk flow. Thus, boundary conditions for distributed parameter models are always

formulated in terms of upstream pressure (or density) and use zero fluxes and normal velocity gradients at downstream boundaries. Then, if the model dimensionality is reduced downstream, the boundary conditions for the downstream model at the interface are calculated from the average of the upstream values (*cf.* 2.124). Alternatively, if the model dimensionality increases from a macroscopic or one dimensional model upstream to a two dimensional model downstream, flat distributions are assumed at the interface (*cf.* 2.125). In this way, consistent (if not always the most convenient) boundary conditions for distributed parameter (especially two dimensional) models are guaranteed, irrespective of the nature of the models upstream and downstream. It could be argued that this approach is rather crude, especially given the previous discussion of discontinuities in mixing characteristics which might also occur (see section 2.5.1). However, it is justified by the fact that any errors introduced are significant only in the neighbourhood of the interface and has the advantages of satisfying overall conservation and being both straightforward to implement and consistent with the use of zero gradient outlet conditions in most CFD codes.

For closed boundaries the problem is the same in principle but in this case it is normally possible to calculate the heat flux on the side of higher dimensionality using a film heat transfer model and then obtain the corresponding heat flux on the side of lower dimensionality by spatial averaging (*cf.* 2.124).

2.6 Mathematical Classification of Model Equations

In the previous sections the formulation of mathematical models of process systems has been considered from a largely physical perspective. In this section consideration is given to the mathematical classification of the differential equations and boundary conditions that form these process models.

2.6.1 Differential Equations

Single partial differential equations (PDEs) are normally classified by comparison with the standard quasi-linear two dimensional equation (2.126)

$$a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial x \partial y} + c\frac{\partial^2 u}{\partial y^2} + d\frac{\partial u}{\partial x} + e\frac{\partial u}{\partial y} + fu = g(x,y) \qquad (2.126)$$

where a, b, c, d, e and f may be functions of x or y, the dependent variable u or its first (but not second) order derivatives. The classification involves searching for possible simple wave solutions and hence examining the roots of the characteristic equation (2.127).

$$a\left(\frac{dy}{dx}\right)^2 - b\left(\frac{dy}{dx}\right) + c = 0 \qquad (2.127)$$

The existence or otherwise of roots of (2.127) depends on the value of the discriminant $(b^2 - 4ac)$. The three possible cases are outlined in table 2.3. The terminology is taken by analogy with the general second-order equation in analytical geometry where the type of conic section also depends on the value of the discriminant.

$b^2 - 4ac$	Equation type	Characteristics
> 0	Hyperbolic	Two real characteristics
$= 0$	Parabolic	One real characteristic
< 0	Elliptic	No characteristics

Table 2.3: PDE Classifications

For PDEs where the coefficients in (2.126) are constant these classifications apply over the entire solution domain. If the coefficients are functions of x and y or if the equation is nonlinear the equation type may vary depending on the local values. Such equations are said to be of mixed type.

Elliptic equations govern equilibrium or jury problems. These are boundary-value problems where the solution depends on the boundary conditions over the entire extent of the boundary. Two important features of elliptic equations are that disturbances propagate in all directions through the solution domain and also that the solution is always smooth, even if the boundary conditions are discontinuous [Versteeg and Malalasekera, 1995]. The prototype elliptic equation is Laplace's equation

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{2.128}$$

which describes steady state conductive heat transfer and requires the specification of boundary conditions over the entire boundary $(x_l \leq x \leq x_u,\, y_l \leq y \leq y_u)$.

Parabolic and hyperbolic equations govern marching problems. These are transient or transient-like problems where the solution is sought on some open domain. Information travels through the solution (or some part of it) in a single direction and consequently can be computed by marching forward in this direction from the initial data surface.

Parabolic equations describe time-dependent problems with significant amounts of dissipation. An example is the diffusion equation (2.129) describing transient conductive heat transfer in one spatial dimension.

$$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2} \tag{2.129}$$

This is an example of an initial-boundary-value problem which requires the specification of boundary conditions at both ends of x for all t and initial conditions for all x at $t = 0$.

The solution moves forward in time and diffuses in space. Consequently a disturbance at time t_0 can only affect the solution at later times $t > 0$ and the solution can be obtained by marching forward in time from the initial point. The spatial diffusion ensures the solution is always smooth in the interior at times $t > 0$ even if the initial conditions contain discontinuities. In this case steady state is reached as $t \to \infty$ and at steady state the behaviour becomes elliptic.

Hyperbolic equations are distinguished from the other types by the finite speed at which information travels through the problem and the possible existence of discontinuities in the interior of the solution [Versteeg and Malalasekera, 1995]. Shockwave discontinuities exhibited in supersonic flows of inviscid fluids are a manifestation of hyperbolic behaviour. The prototype hyperbolic equation is the wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{2.130}$$

whose solution can be shown to be a linear combination of functions of two characteristics lines $x - ct = \text{const}$ and $x + ct = \text{const}$ which run through the solution[10]. It can also be shown that the solution has a limited domain of dependence and is influenced only by boundary conditions between the two characteristics. Data outside this domain has no effect. Physically this is caused by the limited speed at which disturbances can travel through the problem. This is demonstrated in the example of a supersonic flow of an inviscid fluid mentioned above where the magnitude of the flow velocity is such that the downstream pressure is unable to propagate disturbances upstream. The limited zone of influence is a feature of hyperbolic equations.

The classification of systems of PDEs is more complicated and involves expressing the equations as a set of first-order equations in matrix form and finding eigenvalues of the resulting coefficient matrices. In general, the governing differential equations for fluid flow behave in a complex way and may differ in character in different parts of the flow domain. A more complete discussion is given by Anderson *et al.* [1984].

The importance of these classifications is that they categorise the behaviour of the solution and also its dependence on the boundary conditions. They are therefore important when considering numerical methods for the solution of PDEs and also in the selection and imposition of appropriate boundary conditions. In practice, however, the complex behaviour of the fluid flow equations means few theoretical results are directly applicable and the modelling of such systems is often guided by physical arguments and the success of previous simulations [Versteeg and Malalasekera, 1995]. An example of this is given by Schiesser [1991] who departs from the traditional classification and instead uses what he describes as a 'rather loose' definition to categorise systems of PDEs by

[10]This is the basis of the method of characteristics for hyperbolic problems where the solution is obtained by transforming a partial differential equation into a set of ordinary differential equations.

inspection for selection of appropriate numerical schemes based on the method of lines.

2.6.2 Boundary Conditions

For the purposes of the model generation approach developed in this thesis two main issues are important with regard to the mathematical classification of boundary conditions: the required number of conditions and their mathematical form. These are discussed below.

2.6.2.1 Number of Boundary Conditions

In general the solution of a system of ordinary or partial differential and algebraic equations (DAEs or PDAEs) involves integrating one or more times with respect to each independent variable appearing in the equation set. With each integration a new constant (or function) of integration is introduced. To evaluate these constants (or functions) requires the specification of a set of n boundary conditions (usually) on the edges of the space-time region defined by the independent variables.

The simplest method of determining n is to consider only first order systems. If necessary, higher order systems can be reduced to first order systems by defining extra variables. Consider, for example, the third order ordinary differential equation

$$a\frac{d^3u_1}{dx^3} + b\frac{d^2u_1}{dx^2} + c\frac{du_1}{dx} = f(x, u_1) \tag{2.131}$$

Defining the extra variables $u_2 = du_1/dx$ and $u_3 = du_2/dx$, the system becomes:

$$\frac{du_1}{dx} = u_2 \tag{2.132}$$

$$\frac{du_2}{dx} = u_3 \tag{2.133}$$

$$a\frac{du_3}{dx} + bu_3 + cu_2 = f(x, u_1) \tag{2.134}$$

For first order systems comprising solely ordinary or partial differential equations (with no algebraics) the required number of boundary conditions n is simply equal to the number of dependent variables (u_1, u_2, u_3).

For DAEs there may be an additional complication. The algebraic equations must be satisfied over the entire extent of the independent variable and hence derivatives of all orders of the algebraic equations must also be satisfied. These differential equations may or may not be independent of the original system. The index of the system is defined as the highest order differential equation which is required to provide an independent set. The boundary conditions must be consistent with the original algebraic equations and also any new equations generated by differentiation. For systems of index two and above, the additional differential constraints suggested by the algebraic equations reduce the number

of conditions which can be specified arbitrarily at a boundary (and also have implications regarding numerical solution). Further discussion is given by Pantelides *et al.* [1988].

An equivalent complication may arise in systems of PDAEs. In this case the algebraic equations and all orders of their derivatives must be satisfied over the entire space of the independent variables. Again the boundary conditions must be consistent with the original algebraic equations and also any new equations generated by differentiation. Although there is no formal definition of the index for PDAEs, by analogy with DAE systems of index 2 and above, it is clear that differential constraints implied by the algebraic equations may reduce the number of boundary conditions which can be specified arbitrarily. The system of equations describing compressible fluid flow, for example, contains spatial derivatives of temperature, pressure and density. However, it is clear that all three cannot be specified independently at a boundary since they remain constrained by an (algebraic) equation of state (such as the ideal gas law). For incompressible fluids, on the other hand, the pressure and density are not related and so each may be given arbitrary values.

2.6.2.2 Types of Boundary Conditions

In the mathematical literature it is common to classify boundary conditions into three types: Dirichlet conditions, Neumann conditions and Robin boundary conditions (also known as mixed or third-type conditions) which are a combination of the other two.

The terminology is based on the names associated with the solution of Laplace's equation (2.135) subject to the three specific types of boundary condition [Anderson *et al.*, 1984].

$$\nabla^2 u = 0 \tag{2.135}$$

The Dirichlet problem refers to the solution of (2.135) in a closed domain D subject to boundary conditions which require the solution to take on prescribed values on the enclosing boundary B. Thus Dirichlet conditions are boundary conditions on the dependent variable u of the form

$$u = f(x) \tag{2.136}$$

The Neumann problem also refers to the solution in D but with the boundary conditions given in terms of the normal derivative of u on B rather than the dependent variable. Consequently, Neumann boundary conditions are of the form

$$\frac{\partial u}{\partial n} = g(x) \tag{2.137}$$

Lastly, the mixed problem refers to the solution of Laplace's equation in D subject to boundary conditions on B which are a combination of the dependent variable and its normal derivative

$$a_1(x)u + a_2(x)\frac{\partial u}{\partial n} = h(x) \tag{2.138}$$

The assignment of the names Dirichlet, Neumann and Mixed or Robin is generally used to define types of initial and boundary conditions for any partial differential equation.

In fluid flows, examples of Dirichlet conditions include inflow conditions (2.49), no-slip (2.44) and no flow-through (2.41), Neumann conditions include outflow (2.50) and symmetry conditions (2.54 and 2.55) and Mixed or Robin conditions (which generally arise in simplified models) include the Danckwerts inlet condition (2.120) and (though not shown above) may also include conditions equating the reaction rate or heat flux at a boundary to an averaged flux expressed in terms of a film transfer model.

In practice, boundary conditions for single and multiphase flows are usually based on physical arguments (see section 2.3) and the above classifications are of minor importance.

2.6.3 Differential Equation and Boundary Condition Domain Validity

In general terms, the solution of a differential equation in some region D of the space of the independent variables is a function that has all the partial derivatives appearing in the equation in some domain containing D and satisfies the equation everywhere in D [Kreyszig, 1993]. Particular solutions for given situations are then obtained by the application of additional constraints (boundary conditions) on the boundary (B) of D which the solution must also satisfy. The formulation of appropriate boundary conditions for fluid flows based on the boundary type has been considered above (see section 2.3).

However, in some circumstances discontinuities in the boundary conditions give rise to singularities such that the differential equation is only satisfied arbitrarily close to the boundary but not at the boundary itself. Consider the simple ODE (2.139)

$$\frac{du}{dx} = u \qquad 0 \leq x \leq \infty \tag{2.139}$$

with the boundary (initial) condition

$$u(0) = 1 \tag{2.140}$$

The system represented by (2.139 and 2.140) yields the analytical solution

$$u = \exp x \qquad 0 \leq x \leq \infty \tag{2.141}$$

If the system has a step change from $u = 0$ to $u = 1$ at $x = 0$, then $u(0)$ is not defined. However, a solution of an ODE is defined as a continuous function, so $u(x)$ is defined for $x > 0$ and by continuity can converge to $u = 1$ as $x \to 0$. This is written as $u(0^+) = 1$ and yields the solution above (2.141). Thus, the differential equation (2.139) need not be satisfied at $x = 0$ (or indeed at any specific point in the domain) and the solution at these points must be determined by continuity arguments.

In fluid flow models, examples of boundary conditions with discontinuities include interfaces between phases in models based on the local instantaneous formulation (see section 2.2.1), boundaries between simplified models with discontinuities in mixing characteristics (see section 2.5.1) and may also include conditions arising from assumptions made in the model analysis, such as the determination of the system response to a feed step change or impulse (see, for example, Froment and Bischoff [1990]).

2.7 Scope of Work

A truly general purpose model generation tool would support the automatic generation of mathematical models of process systems at any level of complexity, from single phase laminar systems described by simple steady-state macroscopic balances through to three dimensional dynamic models of turbulent multiphase systems in arbitrary geometries. However, the complex nature of process modelling means that such a tool is unlikely to be realised at present and any practical approach must therefore give consideration to its intended scope. The main theoretical difficulties lie in the formulation of appropriate closure relationships for turbulent single phase modelling and, more notably, in both the model formulation and closure issue in general for multiphase systems (see section 2.2). Furthermore, the mathematical description of systems involving complex geometries, whilst introducing no new theoretical difficulties, constitutes a considerable problem in practice since detailed surface modelling is required to allow the accurate imposition of boundary conditions and the spatial subdivision of the flow domain (mesh generation) needed for numerical solution of the model equations.

The main objective of the work presented in this thesis is the development of a (prototype) software tool for the automatic generation of combined lumped and distributed parameter process models from purely physical descriptions. The generated models must provide a complete and consistent mathematical description and therefore include conservation equations, constitutive equations, boundary conditions and, for averaged models, closure relationships. An exception to this rule is the generation of equations for physical property calculation/prediction for which the existence of an interface to an appropriate software package is assumed. However, for the reasons outlined above, this requires a restriction on the class of process systems under consideration and hence a reduction in the scope of the model generation tool.

The principal restriction imposed in this thesis is that the modelling of multiphase systems is not considered. At present, existing approaches to multiphase modelling are limited in the class of systems they can describe and the closure problem in particular is a considerable one. Moreover, there is still significant disagreement between models proposed by different workers not only in terms of the closure relations but even in terms of the basic two-fluid governing equations. The uncertainty in the physics and the consequent lack of general purpose theoretical approaches to multiphase modelling mean that the generation or even formulation of general predictive multiphase models is not possible at this time. Note that, although multiphase systems are not considered explicitly, they may be represented and modelled as pseudo-homogeneous single phase systems using mean properties.

The other main restriction is that complex geometrical descriptions of process units are not supported at present. Thus, for distributed parameter process models, each control volume is approximated by a single, regular cylinder or annulus and only axisymmetric flows are considered. This approach permits the generation of process models describing spatial variation of the dependent variables that is useful and easy to visualise whilst minimising the effort required by the user for problem specification. This has the advantage that a textual input format may be (reasonably) used and that the generated models can be solved in existing modelling environments. To this end the generated model output will be in the format of the *gPROMS* modelling language [Barton and Pantelides, 1994; Oh and Pantelides, 1996] for direct use in the *gPROMS* process modelling environment. Note that no geometrical restrictions are placed on process systems described by macroscopic models.

In addition to the above, a number of minor restrictions are also imposed. These include using scalar forms of the one dimensional and quasi-steady state macroscopic momentum conservation equations (2.87 and 2.94 respectively), considering only Newtonian fluids and turbulence models based on the Boussinesq approximation (2.13 or 2.29) (note that the generation of equations for the prediction of the turbulent proportionality constants, such as the k-ϵ turbulence model, is not considered at present) and, following the approach used in commercial CFD codes, using simple Fickian-type linear equations for the dispersive mass flux rather than more rigorous nonlinear models such as the Stefan-Maxwell equations.

The restrictions outlined above provide a feasible and practical framework both for the development of a model generation approach and for its implementation in a software tool. They support the generation of process models suitable for the description of the dynamic behaviour of (potentially) entire process plants in a manner which is consistent with the underlying fluid mechanics, rather than focusing on the detailed description of single and multiphase fluid flows in complex geometries which forms the essence of CFD.

2.8 Summary

In this chapter the mathematical modelling of single and multiphase systems was considered using a rigorous theoretical approach based on continuum mechanics. Attention focused on the formulation of models appropriate for the description of turbulent and multiphase systems and the importance of averaging operations in the derivation of mathematical descriptions which are consistent with the continuum hypothesis was discussed. Consideration was then given to the specification of appropriate boundary conditions and the derivation of simplified models. The mathematical classification of the model equations and boundary conditions was also briefly discussed. Finally, based on this material, the scope of the model generation approach developed in this thesis was set out.

In the next two chapters a process representation and model building procedure are defined based on the theoretical approach to process modelling presented in this chapter and which together support the automatic generation of mathematical models of process systems from purely physical descriptions.

Chapter 3

Process Representation

In chapter 1 the background, motivation and objectives for this work were presented. The main aim was identified as the development of a software tool for the automatic generation of combined lumped and distributed parameter process models from physical descriptions.

In this chapter a process representation is defined permitting the description of a process system in physical terms and allowing the application of the theoretical approach to process modelling presented in chapter 2.

The representation consists of, firstly, a general framework supporting the decomposition and structuring of a process system into basic conceptual elements and, secondly, a process modelling language encapsulating this framework and providing a formal means of describing a process system in terms of it. Both of these are discussed below.

3.1 Conceptual Framework

The conceptual framework presented here is a refinement of that developed in earlier work at Imperial College by Vázquez-Román [1992] and later by Perkins *et al.* [1996] and Cho [1997]. The validity of these previous representations has been proven by their successful implementation in two prototype computer programs for the automatic generation of lumped parameter process models (see section 1.1).

The approach is based on the premise that all material undergoing processing is instantaneously in a stable thermodynamic state, implying that all material is present in well-defined thermodynamic phases which may be described in terms of a finite set of state variables [Perkins *et al.*, 1996]. Therefore a *process* system may be ultimately represented as a collection of interacting *phases*, each containing a number of inert and/or reacting chemical *components*, contained within *vessels*. In this context a vessel is defined simply as a containing surface, providing volume constraints and restricting the dynamic and kinematic behaviour of the internal fluid flow.

Each vessel may be linked to one or more other vessels by idealised topological *connections*, considered as being of (essentially) zero extent such that they contain no

material holdup. The connections are made via *ports* on vessel surfaces, which may be either open or closed to mass transport and may therefore represent either free fluid interfaces or solid walls.

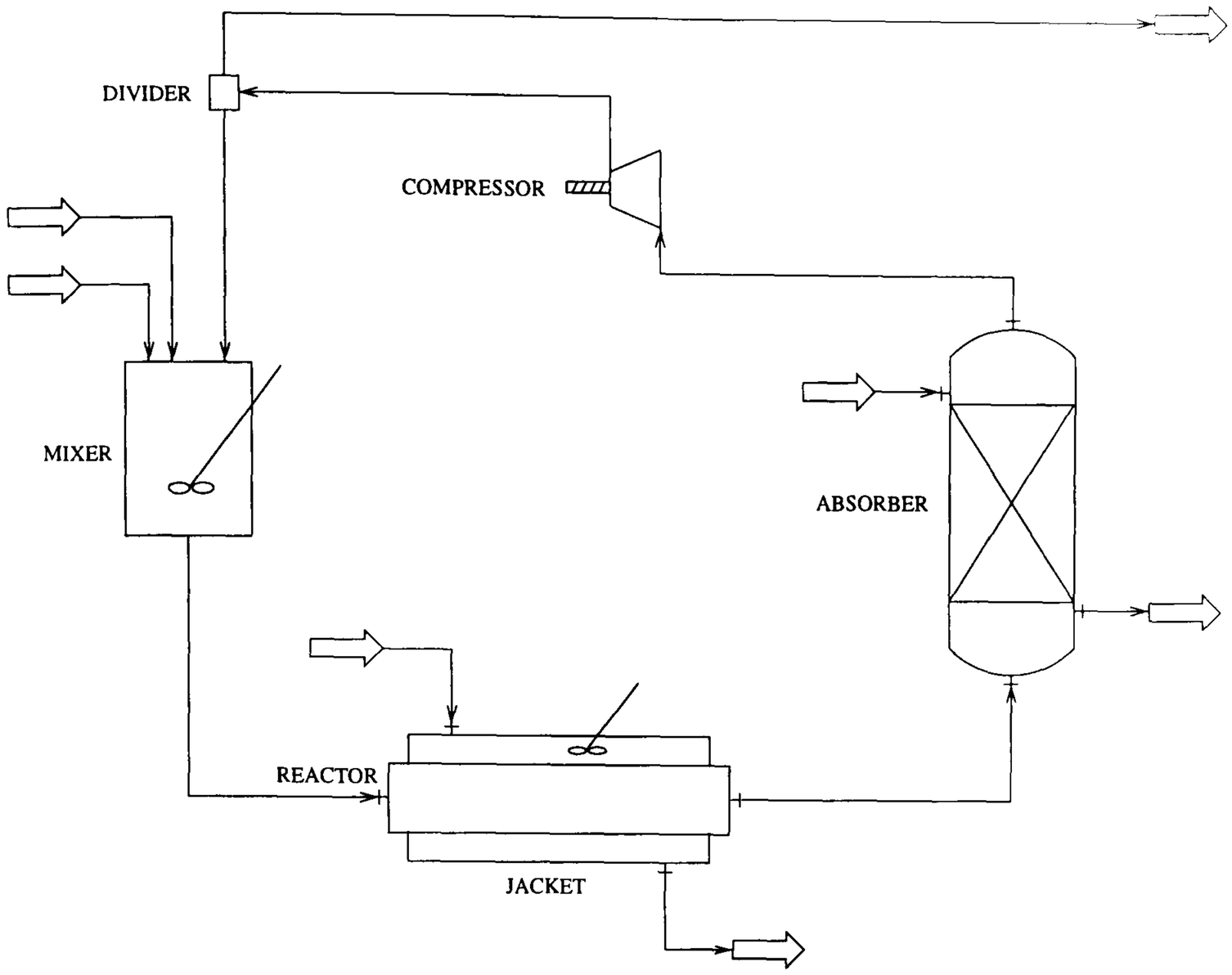
Inter-phase mass, energy and momentum interactions, either within a single vessel or between two vessels through a connection, are described by physico-chemical *transfer laws*, usually defined in terms of the states of the two connected phases (note however that transfer laws are not needed to describe inter-vessel interactions within a phase that is continuous across a free fluid interface).

Clearly a process system does not exist in isolation but receives feeds and delivers products and may use or produce process utilities such as steam or cooling water. There may also be interactions with the environment (both intentional and otherwise) through, for example, heat losses or leaky valves. These interactions are described by exchanges with process *reservoirs* which are again linked to vessels via connections. The concept of a reservoir provides the means of delimiting the process system under consideration from its surroundings.

This basic framework is sufficiently general to support the description of a process system of arbitrary complexity in a purely physical manner. Moreover the decomposition of a process system into a collection of interacting phases provides a description that is consistent with the continuum hypothesis and therefore allows the application of continuum mechanics and the theoretical approach to process modelling presented in chapter 2.

An illustration of this approach for the physical representation of a process system is given in figure 3.1, using an example adapted from Heydweiller *et al.* [1977]. This represents a typical chemical process plant, consisting of a reaction section, a separation section and a recycle. The reaction section comprises a gas phase tubular reactor with temperature control provided by heat exchange with a heating/cooling medium. The reaction products are recovered in a gas-liquid contacting column whilst unused reactants are recycled to the reactor feed. A purge prevents build up of contaminants and/or inerts. The flow through the process is driven by a single compressor.

The process representation (in terms of the conceptual framework) for this example consists of six interconnected vessels. Five of these, the mixer, reactor, jacket, compressor and divider, contain a single gas or liquid phase whilst the absorber contains an interacting multiphase gas-liquid mixture. The vessels are linked together through connections via ports on their surfaces whilst feeds, products and process utilities are drawn from or delivered to upstream and downstream reservoirs.



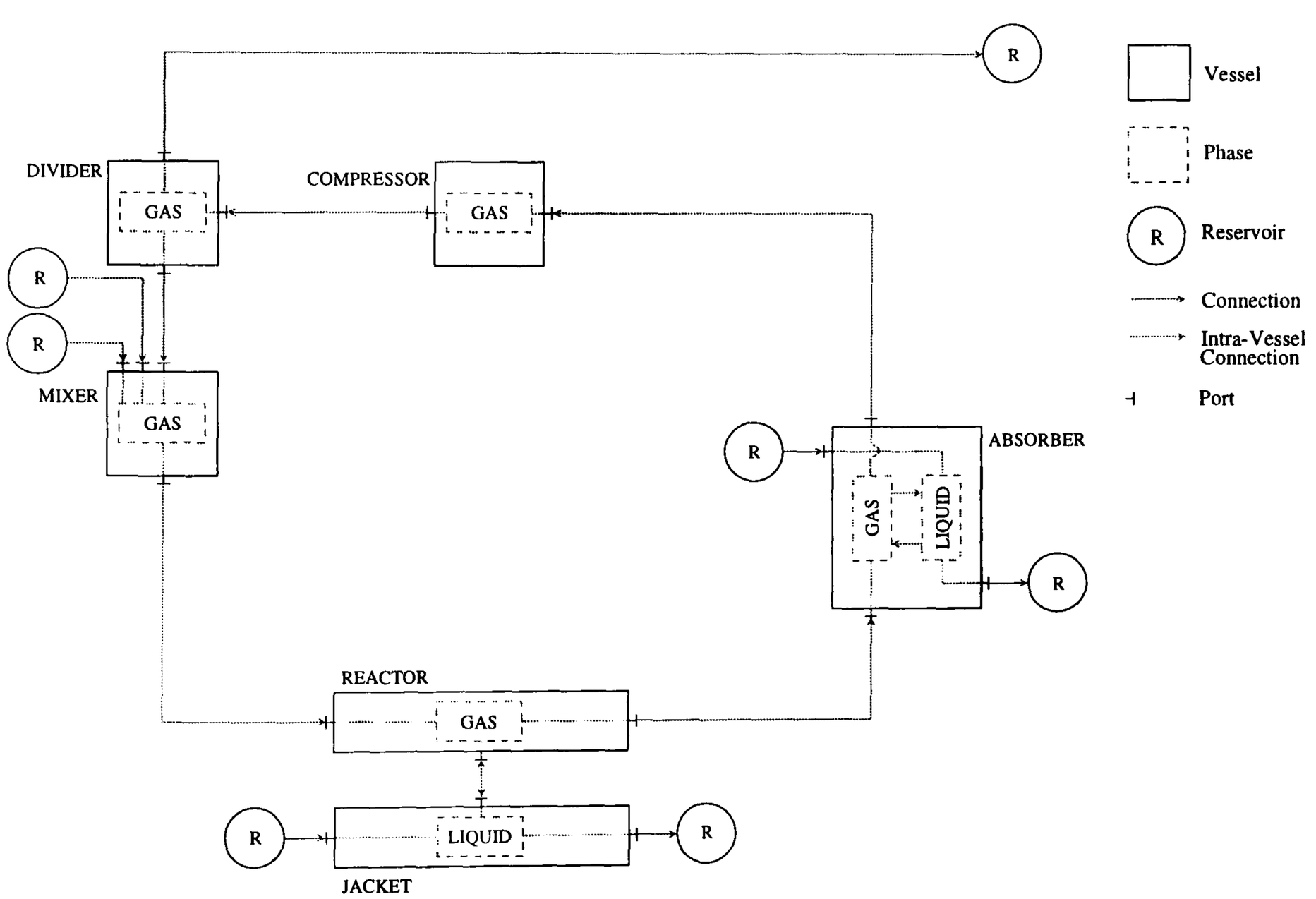


Figure 3.1: Conceptual Representation of a Hypothetical Process System

3.2 Process Modelling Language

In the previous section a general framework was presented which permits the physical representation of a process system in terms of fundamental conceptual elements. In this section a process modelling language is defined which encapsulates this framework and provides a formal means of describing a process system in terms of it. Although the framework is fully general and makes no restriction on the type of process system it can be used to represent, the language incorporates only those parts needed for the restricted scope of systems considered in this work (see section 2.7).

The language is based on the declaration of keywords and is structured closely around the conceptual framework. Accordingly, it consists of three main sections: the vessels section, the reservoirs section and the connections section. Together these provide the means of declaring both the basic conceptual elements that comprise the physical representation and also any additional information needed to complete the process description. The latter may include, for example, the specification of geometrical attributes and/or modelling assumptions. The syntax and semantics of each section are presented below. A formal definition of the language is also given in terms of lexical tokens and a context-free grammar in appendices C and D respectively.

3.2.1 Vessel Section

A vessel is defined simply as a containing surface enclosing one or more phases. By definition, a vessel cannot be empty (either completely or in part) and the phases it contains must fill its volume completely. In essence a vessel defines a local control volume (and its associated surfaces and boundaries) over which the conservation principles can be applied.

The formal declaration of a vessel comprises the specification of not only the attributes of the vessel itself (such as its name, size, shape, orientation *etc.*) but also a number of other conceptual elements (and their associated attributes) which are contained within, or form part of, the vessel. These include the *phases* (and chemical *components* of which they are comprised) enclosed by the vessel's surfaces, the *transfer laws* which describe interactions between pairs of phases contained in the vessel and the *ports* on the vessel's surfaces which represent its interfaces with the environment (other vessels and reservoirs).

The formal syntax of a vessel declaration is given below (definition 3.1)[1]. It consists of the keyword **VESSEL** followed by two attributes: a user-specified name and a keyword indicating the vessel type. The remainder of the declaration is decomposed into four subsections, each of which is formally defined in sections 3.2.1.1 - 3.2.1.4. The

[1] Language keywords are written in **bold** whilst subsections (formally defined elsewhere) are denoted by *italics*. User specified identifiers are enclosed within angled brackets.

vessel name is a unique identifier by which the vessel can be referenced globally. It may be any combination of upper case or lower case text, numbers and underscores but must begin with a letter. The vessel type is a keyword (**MACROSCOPIC**, **1-D** or **2-D**) used to specify the degree of spatial distribution (macroscopic, one-dimensional or two-dimensional respectively) required for the mathematical model of that vessel. Since only axisymmetric cylindrical flows are considered in this thesis (see section 2.7), three dimensional models are not included at present although the necessary extension to the language is trivial.

In all language sections and subsections colons are used to separate the attributes and a semi-colon indicates the end of the section. Compound attributes (attributes containing a number of other attributes) are enclosed within square brackets.

Language Definition 3.1 *Vessel Declaration.*

VESSEL: <name>: *VesselType*;

Geometry Subsection	(Language Definition 3.2)
Phase Subsection	(Language Definitions 3.5 and 3.6)
Port Subsection	(Language Definitions 3.7 and 3.8)
Assumptions Subsection	(Language Definition 3.11)

Note that, since only single phase systems are considered in this work (see section 2.7), the language includes no *transfer law* subsection for the description of inter-phase interactions within a vessel.

Example vessel declarations are given in figure 3.2 (examples of each language subsection are given later).

```
VESSEL: pipe: 1-D;
  ...

VESSEL: cstr_R101: MACROSCOPIC;
  ...
```

Figure 3.2: Example Vessel Declarations

3.2.1.1 Geometry Subsection

The generation of distributed parameter process models for vessels of type **1-D** and **2-D** necessarily requires the specification of the geometrical attributes of the control volume and enclosing surfaces defined by the vessel (see section 2.7). If no spatial distribution is assumed then the geometry subsection is omitted.

The formal syntax of the geometry subsection is given below (definition 3.2). It consists of the keyword **GEOMETRY** followed by three attributes: a keyword describing the vessel shape, a compound size attribute defining its dimensions and a keyword specifying the orientation. The shape keyword in conjunction with the size declaration (discussed further below) describes completely the volume enclosed by the vessel and the dimensions of each bounding surface. It also defines an appropriate coordinate system with which to span the volume. Since consideration is restricted to simple cylindrical geometries in this work (see section 2.7), only a single shape keyword (**CYLINDER**) is supported at present. The formal syntax of the size declaration consists simply of three keywords representing the domain axes (**AXIAL**, **RADIAL** and **ANGULAR** for cylindrical coordinate systems) each followed by two real-valued numerical constants specifying the lower and upper limits of the domain. In fact, since only two-dimensional models are considered at present, the specification of the third (**ANGULAR**) domain is not strictly necessary (definition 3.3) but is included in the language for completeness (definition 3.4). Finally the vessel orientation indicates the inclination of the vessel (**UP**, **DOWN**, **LEFT** or **RIGHT**) with respect to a common inertial reference frame. The use of **UP** versus **DOWN** or **LEFT** versus **RIGHT** defines the positive direction of the axial coordinate and therefore supports the description of vessels connected in a countercurrent flow arrangement (see chapter 6). Note that if the orientation is omitted, the vessel orientation is taken as **LEFT**.

Language Definition 3.2 *Geometry Subsection Declaration.*

GEOMETRY: *VesselShape* [*VesselSize*]: *Orientation*;

Language Definition 3.3 *Vessel Size Declaration (i).*

AXIAL=< z_1 >,< z_2 >: **RADIAL**=< r_1 >,< r_2 >

Language Definition 3.4 *Vessel Size Declaration (ii).*

AXIAL=< z_1 >,< z_2 >: **RADIAL**=< r_1 >,< r_2 >: **ANGULAR**=< θ_1 >,< θ_2 >

Two example geometry subsection declarations are given in figure 3.3.

```
GEOMETRY: CYLINDER [AXIAL = 0, 0.5: RADIAL = 0, 0.025:
                    ANGULAR = 0, 360]: LEFT;

GEOMETRY: CYLINDER [AXIAL = 0, 10: RADIAL = 0,0.5]: UP;
```

Figure 3.3: Example Geometry Attribute Declarations

3.2.1.2 Phase Subsection

The basic conceptual framework presented above is based on the premise that all material undergoing processing is instantaneously in a stable thermodynamic state, implying that it exists in well-defined thermodynamic phases. A phase therefore represents a material medium which is continuous (or can be treated as continuous in the case of multiphase systems) within the control volume defined by the vessel and its corresponding set of conserved quantities (total mass, chemical components, energy and momentum).

The formal syntax of the phase subsection is given below (definition 3.5). It consists of the keyword **PHASES** followed by two attributes: a user-specified name followed by a compound attribute defining a list of chemical components. The phase name is a locally unique identifier by which the phase may be referenced within the vessel (phases in different vessels may share a common name). As with vessels, the name may consist of any combination of text, numbers and underscores but must start with a letter. The component list is simply a list of locally unique identifiers (subject to the same rules as the vessel and phase names) defining the chemical species which comprise the phase. For pure (single component) phases this attribute is optional and may be omitted (definition 3.6). A useful extension to the phase subsection, though not implemented at present, would be to include language enabling the user to specify a set of possible reactions which could occur. Perhaps the most likely form for this extension would be a library of chemical reactions containing the stoichiometry and kinetics for both individual reactions and sets of reactions in particular kinetic pathways.

Note that, at present, only single phase systems are considered in this work (see section 2.7) and consequently no phase type attributes are required for inter-vessel phase associations [Cho, 1997].

Language Definition 3.5 *Phase Subsection Declaration (i).*

PHASES: <name>: [<a list of components>];

Language Definition 3.6 *Phase Subsection Declaration (ii).*

PHASES: <name>;

Two example phase subsection declarations are given in figure 3.4.

```
PHASES: reactingGas: [N2 O2];

PHASES: coolingWater;
```

Figure 3.4: Example Phase Attribute Declarations

3.2.1.3 Port Subsection

A port represents a vessel's interface with its environment and is defined simply as a portion of a vessel's surface through which it may be connected to another vessel or reservoir.

The formal syntax of the port subsection is given below (definition 3.7). It consists of the keyword **PORT** followed by three attributes: a user-specified name, a compound attribute defining both the size of the port and its location on one of the vessel's surfaces and a keyword indicating the port type. The port name is a locally unique identifier by which the port may be referenced within the vessel (ports in different vessels may share a common name). As before, the name must begin with a letter but may then consist of any combination of text, numbers and underscores. The port geometry (discussed further below) defines both the dimensions of the port and also its position on one of the bounding vessel surfaces. For vessels where no spatial distribution is assumed the port geometry attributes are omitted (definition 3.8). Finally the port type is a keyword defining the port as either **OPEN** or **CLOSED** to mass transport.

Language Definition 3.7 *Port Subsection Declaration (i)*

PORTS: <name>: [*PortGeometry*]: *PortType*;

Language Definition 3.8 *Port Subsection Declaration (ii)*

PORTS: <name>: *PortType*;

The syntax of the port geometry declaration follows closely that of the vessel size declaration described above (definitions 3.3 and 3.4). It again consists of three keywords representing the domain axes (**AXIAL**, **RADIAL** and **ANGULAR**) each followed by up to two real-valued numerical constants. Together these specify both the size of port and also its location on one or other of the vessel surfaces. For example, the port size declarations given in figure 3.5 define the port `inlet` on the vessel surface at $z = 0$ of size $\pi(0.025)^2$ and the port `wall` on the surface at $r = 0.025$ of size $2\pi(0.025)(10)$. As before, the specification of the third (**ANGULAR**) domain is not strictly necessary (definition 3.9) but is included in the language for completeness (definition 3.10).

Language Definition 3.9 *Port Geometry Declaration (i).*

AXIAL=< z_1 >,< z_2 >: **RADIAL**=< r_1 >,< r_2 >

Language Definition 3.10 *Port Geometry Declaration (ii).*

AXIAL= < z_1 >,< z_2 >: **RADIAL**=< r_1 >,< r_2 >: **ANGULAR**=< θ_1 >,< θ_2 >

Two example port subsection declarations are given in figure 3.5.

```
PORTS: inlet [AXIAL = 0: RADIAL = 0, 0.025:
              ANGULAR = 0, 360]: OPEN;

PORTS: wall [AXIAL = 0, 10: RADIAL = 0.5]: CLOSED;
```

Figure 3.5: Example Port Attribute Declarations

3.2.1.4 Assumption Subsection

Although not part of the basic conceptual framework the assumption subsection provides a formal mechanism for the declaration of common engineering (knowledge-based) modelling assumptions that are fundamentally physical (rather than mathematical) in nature. This allows the user both to explore their effect on the model form and also to generate alternative, consistent models of the same process system (perhaps tailored to suit a specific task).

The formal syntax of the assumption subsection is given below (definition 3.11). It consists simply of the keyword **ASSUMPTIONS** followed by a list of keywords specifying assumption types. Each of these types refers to a specific aspect of either the physicochemical behaviour or operational circumstance of the vessel being defined. Typical examples would include the declaration of **ISOTHERMAL** or **STEADY_STATE** operation or the assumption of an **INCOMPRESSIBLE** fluid phase or **FRICTIONLESS** vessel walls.

Language Definition 3.11 *Assumption Subsection Declaration*

ASSUMPTIONS: *AssumptionTypes*;

An example assumption subsection declaration is given in figure 3.6.

```
ASSUMPTIONS: INCOMPRESSIBLE;
             STEADY_STATE;
```

Figure 3.6: Example Assumption Declarations

3.2.2 Reservoir Section

Reservoirs provide a means of delimiting the system under consideration from its surroundings and also permit the description of exchanges and interactions with entities that are not modelled explicitly, such as process utilities.

The formal syntax of a reservoir declaration is given below (definition 3.12). It consists simply of the keyword **RESERVOIRS** followed by a list of user-specified

names. As before, each name may be any combination of text, numbers and underscores but must begin with a letter. No further specifications are needed since any additional information required by the model generation code is deduced from the connectivity (see section 3.2.3).

Language Definition 3.12 *Reservoir Declaration*

RESERVOIRS: <a list of reservoirs>;

An example reservoir declaration is given in figure 3.7.

```
RESERVOIRS: InletR1;
            InletR2;
            R3;
            OutletR;
```

Figure 3.7: Example Reservoir Declarations

3.2.3 Connection Section

A connection is defined as an idealised topological link either between two vessels or between a reservoir and a vessel. Each is considered to be of zero extent and therefore contains no material or energy holdup. In essence, a connection provides a matching of conserved quantities between connected entities.

The formal syntax of a connection declaration is given below (definition 3.13). It consists simply of the keyword **CONNECTIONS** followed by a list of individual connections. Each of these may be one of two types: an inter-vessel connection or a vessel-reservoir (or *vice versa*) connection. The former consists of a (previously declared) vessel name and port name followed by a right arrow (implemented as '–' and a '>') and the names of a second vessel and port (definition 3.14). The size and types of the two ports must be the same. For the latter the vessel and port names on one side are replaced by a reservoir name (definition 3.15 or 3.16). Inter-reservoir connections have no meaning and are illegal.

For open connections (*i.e.* those between open ports or between a reservoir and an open port) the directionality must reflect the direction of bulk flow (see section 2.3). For closed connections the directionality is arbitrary and simply defines a sign convention for the connection in the generated mathematical model (see chapter 4).

Language Definition 3.13 *Connection Declaration*

CONNECTIONS: *ConnectionList*;

Language Definition 3.14 *Connection - type (i)*

<vessel>.<port> → <vessel>.<port>

Language Definition 3.15 *Connection - type (ii)*

<reservoir> → <vessel>.<port>

Language Definition 3.16 *Connection - type (iii)*

<vessel>.<port> → <reservoir>

Example connection declarations are given in figure 3.8.

```
CONNECTIONS: R1->pipe.inlet;
             pipe.wall->R3;
             pipe.outlet->absorber.inlet;
```

Figure 3.8: Example Connection Declarations

3.3 Process Modelling Language Example

An example of the use of the process modelling language for the description of the reaction section of the process system in figure 3.1 is given below. It declares the vessels `reactorTube` and `coolingJacket` with typical physical attributes and modelling assumptions. The limits of the sub-system are represented by reservoirs `R1` and `R2`, the cooling water feed and product, and `R3` and `R4`, the upstream feed from the mixer and downstream product to the absorber respectively.

Further examples of complete language declarations are given as part of the case studies presented in chapter 6.

3.4 Summary

In this chapter a process representation has been presented which permits the description of a process system in physical terms. The representation consists of two parts: a general framework supporting the decomposition of a process system into basic conceptual elements and a process modelling language which encapsulates the framework and provides a formal means of describing a process system in terms of it.

The conceptual framework is based on the premise that all material undergoing processing is instantaneously in a stable thermodynamic state, implying that it exists in well-defined thermodynamic phases. Thus a process is ultimately represented as a

```
VESSEL: reactorTube: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0, 10: RADIAL = 0, 0.037]: LEFT;

PHASES: gas: [O2 CO2 C2H4 C2H6 VAc H20 HAc] ;

PORTS: inlet  [AXIAL = 0: RADIAL = 0, 0.037]: OPEN ;
       outlet [AXIAL = 10: RADIAL = 0, 0.037]: OPEN ;
       wall   [AXIAL = 0, 10: RADIAL = 0.037]: CLOSED ;

ASSUMPTIONS: QUASI_STEADY_STATE;
             PLUG_FLOW;
#-----------------------------
VESSEL: coolingJacket: MACROSCOPIC;

PHASES: water;

PORTS: inlet: OPEN;
       outlet: OPEN;
       wall: CLOSED;

ASSUMPTIONS: STEADY_STATE;
             INCOMPRESSIBLE;
#-----------------------------
RESERVOIRS:

R1;
R2;
R3;
R4;
#-----------------------------
CONNECTIONS:

R1->coolingJacket.inlet;
coolingJacket.outlet->R2;
R3->reactorTube.inlet;
reactorTube.outlet->R4;
reactorTube.wall->coolingJacket.wall;
```

Figure 3.9: Complete Language Example

collection of interacting phases contained within vessels, which are linked together via connections.

The process modelling language is based upon the declaration of keywords and is structured closely around the conceptual framework. It allows the formal declaration of both the basic conceptual elements that comprise the physical representation and also any additional information required to complete the process description, such as geometrical attributes and modelling assumptions.

In the next chapter a methodology for generating mathematical models from the process representation defined in this chapter is developed, using the theoretical approach to process modelling presented in chapter 2.

Chapter 4

Model Building Procedure

In the previous chapter a process representation and modelling language were defined permitting the description of a process system in purely physical terms.

In this chapter a systematic model building procedure is presented which supports the automatic generation of combined lumped and distributed parameter process models from the physical description, based on the theoretical approach to process modelling given in chapter 2. The chapter begins with an overview of the basic model building strategy and then considers the fundamental model building algorithms in detail.

4.1 Basic Strategy

The process representation and modelling language defined in chapter 3 permit the description of a process system in purely physical terms. Moreover, the decomposition of a process system into a set of reservoirs, phases, vessels and connections provides a representation which is consistent with the continuum hypothesis and which therefore allows the application of continuum mechanics and the theoretical approach to process modelling presented in chapter 2. The significance of this representation is that it completely decouples the physical description of a process system from the formulation of the mathematical equations describing its behaviour. Consequently, the model building procedure is concerned only with the generation of the mathematical description of a process system from the physical representation declared in terms of the process modelling language.

The strategy for the generation of the mathematical model equations from the process description is based on the systematic application of model generation algorithms to each of the conceptual elements that comprise the process representation. The procedure begins with the construction of models for each declared vessel. This involves the generation of mass, momentum and energy conservation equations and constitutive relationships in a general form for each phase[1] contained within the vessel, followed by simplification of these equations to a specific form based on the physical description and declared modelling assumptions. The second stage in the procedure is the generation of

[1] In actual fact 'for each phase' is not strictly necessary since only single phase vessels are considered at present but it is included in the model building procedure for completeness (see section 2.7).

models of each reservoir, which serve to define the states of the system boundaries. These consist of equations of state and physical and thermodynamic constitutive relationships but, unlike vessel models, contain no balance equations. Finally, an overall connection model is constructed consisting of equations describing the reservoir-vessel and inter-vessel connections based on the defined process topology and interface types. The overall model building procedure is summarised in algorithmic form in figure 4.1.

Note that whilst an attempt has been made to define the model building procedure in a general form, it can only be considered valid for the (restricted) class of process system for which it has been developed (see section 2.7). Extensions to the scope would certainly necessitate extensions to the model generation algorithms and, depending on their nature, may also require extensions to the overall model building procedure.

- FOR each VESSEL DO
  - FOR each PHASE DO
    * Generate conservation equations.
    * Generate equations of state.
    * Generate physical and thermodynamic relationships.
    * Simplify model according to specified assumptions.

    END

  END

- FOR each RESERVOIR DO
  - Generate equations of state.
  - Generate physical and thermodynamic relationships.

  END

- FOR each CONNECTION DO
  - Generate connection equations.

  END

Figure 4.1: Summary of Model Building Procedure

In the next section the fundamental model generation algorithms which together form the overall model building procedure are presented in detail using a top-down description.

4.2 Model Generation Algorithms

The model generation algorithms are structured around the process modelling language and comprise three sets on two hierarchical levels. The bottom level contains two sets of algorithms for the construction of the individual vessel and reservoir submodels, whilst the set at the top level is responsible for the generation of the overall connection submodel describing the process topology. The formal algorithm for invoking the model building procedure is given below.

Algorithm 4.1 *Generate Mathematical Model.*

1. Generate vessel submodels. (Algorithm 4.2)
2. Generate reservoir submodels. (Algorithm 4.23)
3. Generate connection submodel. (Algorithm 4.24)

The three constituent parts of algorithm 4.1 are discussed below. Note that the creation of the model variables is not included in the model generation algorithms given here (4.3 - 4.24) but is merely inferred by the variables required to build each equation (or part thereof). In the software implementation of the model building procedure (see chapter 5) each new variable is created in the algorithm where it is first used and added to a lookup table from where it is referenced by pointers. This avoids duplication and reduces the potential for errors. The convention for the variable labelling, typing and referencing used in gMaGiC is given in appendix A.2.

4.2.1 Vessel Submodels

Vessel submodels are generated for each vessel declared in the process description. They are based around the construction of mathematical statements describing the principles of conservation of mass, energy and momentum over the control volumes defined by the vessels, using the theoretical approach to process modelling given in chapter 2.

The basic strategy is to construct each submodel in a general form and then reduce it to the specific form required for the particular vessel instance under consideration according to the physical specification. In this way consistency between different models of the same system is ensured by construction. In principle, this approach would involve the generation of general (mass-averaged) three dimensional equations for each vessel submodel followed by simplification to an appropriate form. However, there are a number of problems. Firstly, it is usually a complex procedure and often the derivation involves several steps, with specific assumptions and simplifications required at each step (see section 2.4). These may not be merely analytical or mathematical in nature but require engineering and physical knowledge, for example, to recognise the importance of certain

terms (such as wall fluxes) or to identify terms which are negligible as compared with others, and also to direct the derivation to ensure that the end result is in the required form. In addition, the complexity of the derivation means that more information may be required than is available. For example, the derivation of a lumped parameter model of a continuous stirred tank reactor in this manner would require a full three dimensional geometrical description of the reactor surfaces. However this information is usually unavailable and instead all that is known is the overall vessel volume and the cross-section areas of the inlet and outlet ducts. A further consequence is that the implementation of this approach is likely to be computationally expensive, particularly if the ultimate goal is the generation of a fairly simple model.

For the reasons discussed above, rather than defining a single set of generic three dimensional model equations from which all other models are derived, the approach used here defines three sets of generic equations based on the vessel dimensionality (`MACROSCOPIC`, `1-D` or `2-D`)[2], using the results of the derivations given in section 2.4. Simplified forms of each of these types can then be derived in a more straightforward manner based on the specifications given in the physical description. The generation of each vessel submodel therefore begins by identifying its type (algorithm 4.2).

Algorithm 4.2 *Generate Vessel Submodels.*

1. FOR each `VESSEL` DO

   (a) SWITCH: *VesselType*

      i. CASE: `MACROSCOPIC`

         A. Generate macroscopic model. (Algorithm 4.3)

      ii. CASE: `1-D`

         A. Generate one dimensional model. (Algorithm 4.10)

      iii. CASE: `2-D`

         A. Generate two dimensional model. (Algorithm 4.15)

      END

   END

The algorithms for the generation of each type of vessel submodel are described below.

4.2.1.1 Macroscopic Models

The formal procedure for the generation of a macroscopic vessel submodel is given by algorithm 4.3.

---

[2]Due to the restricted geometrical descriptions employed here, three dimensional models are not considered in this work (see section 2.7).

Algorithm 4.3 *Generate Macroscopic SubModel.*

1. FOR each PHASE DO

   (a) Generate macroscopic mass balance. (Algorithm 4.4)

   (b) Generate macroscopic momentum balance. (Algorithm 4.5)

   (c) Generate macroscopic energy balance. (Algorithm 4.6)

   (d) Generate equations of state. (Algorithm 4.7)

   (e) Generate constitutive equations. (Algorithm 4.8)

   (f) Simplify macroscopic model. (Algorithm 4.9)

END

For each phase contained within the vessel the procedure involves calls to three algorithms for the construction of the individual macroscopic mass, momentum and energy conservation equations (respectively) followed by calls to two shared algorithms (that is, algorithms that are common to models of all types) for the construction of the equations of state and constitutive relationships. The result of steps 1.(a) - 1.(e) is the generation of a macroscopic model in a general form, subject to the phase and port attributes of the declared vessel. The final step 1.(f) of algorithm 4.3 is the application of the individual modelling assumptions declared in the process description. These assumptions, which are fundamentally physical in nature, describe one or more aspects of the physico-chemical behaviour or operation circumstance of the vessel under consideration and serve to reduce the generated macroscopic model and/or individual equations to a more simplified mathematical form.

The first step in algorithm 4.3 is the generation of the macroscopic mass balance. This is based on the prototype macroscopic mass conservation equation (4.1) (or 4.2 for multicomponent phases) given in section 2.4.1.3.

$$\frac{d}{dt}\rho V = -\Delta\rho A v \tag{4.1}$$

$$\frac{d}{dt}\rho_i V = -\Delta\rho_i A v + R_i \qquad i = 1, 2, \ldots, n \tag{4.2}$$

The procedure for its construction is given by algorithm 4.4. Note that for single component phases the application of this procedure results in a single equation (*cf.* 4.1) whilst for multicomponent phases the equation is constructed in vector form of size equal to the number of components n (*cf.* 4.2).

Algorithm 4.4 *Generate Macroscopic Mass Balance.*

1. Add mass holdup term.

2. Write equal sign

3. FOR each PORT DO

   (a) IF *PortType* = OPEN

      i. IF PORT = inlet

         A. Add convective mass flow rate term.

      ii. ELSE PORT = outlet

         A. Subtract convective mass flow rate term.

      END IF

   END IF

END

4. IF PHASE = multicomponent

   (a) Add reaction term.

END IF

The first step in algorithm 4.4 is the generation of the mass holdup term. Since only single phase vessels are considered here and a basic requirement for the process representation is that the phases must fill their enclosing vessel, this is written as the time derivative of the product of the total vessel volume and either the total density or density of component i (where $i = 1, 2, \ldots, n$), depending on whether the phase contains a single component or multicomponent mixture (respectively). The time rate of change of the holdup is equated to the net mass flow rate entering the vessel plus any generation by chemical reaction(s). According to the process representation, a flow may only enter or exit a vessel via the ports on its surfaces. Therefore, for mass conservation, the net mass flow rate entering the phase is given by the difference between that entering through each of the open inlet ports and that leaving through each of the open outlet ports. The terms inlet and outlet merely define the appropriate sign convention in the balance equation and are deduced from the declared connections. The inlet and outlet mass flow rates are written as the product of the total density or density of component i (as appropriate), the port area and the average flow velocity across the port through which the flow is passing (dispersive contributions are ignored). The final step in algorithm 4.4 is the generation of the reaction term. This is only carried out for multicomponent phases and consists simply of a variable representing the net rate of generation of component i by chemical reactions. As noted in section 3.2.1.2, the process modelling language includes no facilities for describing chemical reactions and, consequently, individual equations describing the reaction term for each component are not generated at present.

The second step in algorithm 4.3 is the construction of the macroscopic momentum balance. As discussed in section 2.4.1.3, the prototype macroscopic momentum conservation equation is based on the scalar quasi-steady state form (4.3).

$$-\Delta(\rho A v^2 + pA) - F = 0 \tag{4.3}$$

The formal procedure for its construction is given by algorithm 4.5.

Algorithm 4.5 *Generate Macroscopic Momentum Balance.*

1. FOR each PORT DO

   (a) IF *PortType* = OPEN

      i. IF PORT = inlet

         A. Add convective momentum flow rate term.

         B. Add pressure term.

      ii. ELSE PORT = outlet

         A. Subtract convective momentum flow rate term.

         B. Subtract pressure term.

         END IF

      END IF

   END

2. Subtract friction term.

3. Equate to zero.

It begins with the generation of the net convective momentum flow rate and pressure terms. The former are given by the difference between the flows entering and leaving the phase via the open ports by bulk motion and are written as the product of the total mass flow rate and averaged velocity. The latter account for the difference between the pressure forces acting on the open ports and these are written as the product of the average pressure and the port area. The final two steps simply subtract a term accounting for the frictional drag on the walls of the enclosing vessel and equate the sum of the generated terms to zero. Since no geometrical information is used for the generation of a macroscopic model, no term accounting for the gravitational force is built explicitly in algorithm 4.3 but instead must be included with the pressure term if needed.

The third step of algorithm 4.3 is the generation of the energy balance. This is based on the prototype macroscopic energy conservation equation (4.4), which is based on that given in section 2.4.1.3.

$$\frac{d}{dt}\rho\hat{U}V = -\Delta\left[\rho Av\left(\hat{h}+\frac{1}{2}v^2\right)\right]+Q-W \tag{4.4}$$

The formal procedure for its construction is given by algorithm 4.6.

Algorithm 4.6 *Generate Macroscopic Energy Balance.*

1. Add energy holdup term.
2. Write equal sign.
3. FOR each PORT DO
   (a) IF *PortType* = OPEN
      i. IF PORT = inlet
         A. Add convective energy flow rate term.
      ii. ELSE PORT = outlet
         A. Subtract convective energy flow rate term.
      END IF
   (b) ELSE *PortType* = CLOSED
      i. Add wall heat transfer term.
   END IF
   END
4. Subtract work term.

The first step in algorithm 4.6 is the generation of the holdup term. In general, this consists of the sum of the holdups of the internal, kinetic and potential energies. However, if the potential energy is assumed to be independent of time (*i.e.* due only to gravitational acceleration) and if the holdup of kinetic energy is ignored (for consistency with the quasi-steady state assumption used to derive the scalar form of macroscopic momentum conservation equation 4.3), the total energy holdup is comprised solely of internal energy and hence is written simply as the time derivative of the product of the density, specific internal energy and vessel volume. The holdup is equated to the net convective energy transport to the phase by bulk flow (dispersive contributions are again ignored) and heat transfer minus any (shaft) work done by the phase. The net convective energy flow rate is given by the difference between that entering and leaving the phase via the open ports. Each of these consists of the product of the total mass flow rate and the sum of the specific enthalpy and kinetic energy of the fluid passing through the port. However, since no geometrical information is used in the generation of a macroscopic model, the convective

flow rates include no potential energy contribution which instead must be added as a work term and lumped with the shaft work (below) if needed. Step 3.(b) of algorithm 4.6 adds terms accounting for the heat transfer through any closed ports. These are always written as positive for consistency with the sign convention used in thermodynamics and the directionality is then taken into account in the corresponding film heat transfer model (see section 4.2.3). Finally a negative work term is added (again for consistency with the sign convention used in thermodynamics) accounting primarily for mechanical work done by the phase from its action of moving parts within the vessel (such as impellers) but, as mentioned above, may include a gravitational work term if necessary. No explicit heat of reaction term is generated in the energy balance and this is instead included inherently in the internal energy by defining the energy datum relative to elemental species (see, for example, Bird *et al.* [1960] or Pantelides [1998]).

The fourth step of algorithm 4.3 is the generation of the equations of state for density and enthalpy. In practice, equations of state are available in many different forms and are usually empirical or semi-empirical in nature. Often the selection of appropriate equations for a given process system cannot be made arbitrarily since particular forms are suited to particular applications, such as specific groups of chemical components or certain ranges of operating conditions (see, for example, Reid *et al.* [1987] or Rogers and Mayhew [1992]). For these reasons, the model building procedure is not based on prototype equations of state but instead simply constructs relationships for density and enthalpy which indicate their functional dependence (*cf.* 4.5 and 4.6 or 4.7 and 4.8 for single or multicomponent phases respectively). It is left to the user (or a suitable physical property package) to construct (or invoke) the specific equations themselves.

$$\rho = \rho(p, T) \tag{4.5}$$

$$\hat{h} = \hat{h}(p, T) \tag{4.6}$$

$$\rho = \rho(p, T, \mathbf{x}) \tag{4.7}$$

$$\hat{h} = \hat{h}(p, T, \mathbf{x}) \tag{4.8}$$

The procedure for generating these relationships is straightforward and given by algorithm 4.7, which is common to models of all types.

Algorithm 4.7 *Generate Equations of State*

1. Generate functional form of equation of state for density.

2. Generate functional form of equation of state for enthalpy.

Note that macroscopic models may contain several sets of density and enthalpy variables (arising from the holdup and port transfer terms) and therefore several sets of equations of state may be generated for each macroscopic model.

The fourth step in algorithm 4.3 (and the final step in the construction of the general form of the macroscopic model) is the generation of the physical and thermodynamic constitutive relationships. These equations define relationships between existing model variables and also introduce a new (square) set of equations and variables for additional common modelling quantities (such as molar concentrations and mole fractions). The formal procedure for constructing the constitutive equations is given by algorithm 4.8, which is common to models of all types (although step 4 is not carried out for macroscopic models).

Algorithm 4.8 *Generate Constitutive Equations*

1. Generate enthalpy/internal energy relation.
2. Generate concentration definition equation.
3. IF phase = multicomponent
   (a) Generate density sum equation.
   (b) Generate concentration sum equation.
   (c) Generate mole fraction definition equation.
   (d) Generate mass fraction definition equation.

   END IF
4. IF *VesselType* $\neq$ `MACROSCOPIC`
   (a) Generate total enthalpy equation.
   (b) Generate total energy equation.
   (c) IF phase = multicomponent
      i. Generate Ficks law equation.

      END IF
   (d) Generate Fourier law equation.
   (e) Generate Newtonian shear stress equations.

   END IF

The construction of each individual equation in algorithm 4.8 is straightforward and not expanded further here. For macroscopic models the constitutive relationships consist of

concentration definitions (4.11), an equation relating the enthalpy to the internal energy (4.14) and, for multicomponent phases, mass and mole fraction definitions (4.9 and 4.10) and total density and total concentration summations (4.12 and 4.13).

$$\rho_i = x_i \rho \qquad i = 1, 2, \ldots, n \tag{4.9}$$

$$c_i = y_i c \qquad i = 1, 2, \ldots, n \tag{4.10}$$

$$\rho_i = M_w c_i \qquad i = 1, 2, \ldots, n \tag{4.11}$$

$$\sum_i \rho_i = \rho \tag{4.12}$$

$$\sum_i c_i = c \tag{4.13}$$

$$\rho \hat{h} = \rho \hat{U} + p \tag{4.14}$$

Note that, as with equations of state, macroscopic models may contain several sets of constitutive equations arising from the fact that several variables of each type (such as density) may appear in the model in the holdup and port transfer terms.

The final step in the generation of a macroscopic model is the simplification of the general model to a specific form based on the declared modelling assumptions. The formal simplification procedure is given by algorithm 4.9.

Algorithm 4.9 *Simplify Macroscopic Model*

1. SWITCH: *AssumptionType*
   (a) FOR each `ASSUMPTION` DO
      i. CASE: `FRICTIONLESS`
         A. Remove friction term from momentum balance.
      ii. CASE: `NO_SHAFT_WORK`
         A. Remove work term from energy equation.
      iii. CASE: `WELL_MIXED`
         A. Generate well-mixed equations.
      iv. CASE: `STEADY_STATE`
         A. Remove holdup terms from balance equations.
      v. CASE: `ISOTHERMAL`
         A. Remove energy balance.

vi. CASE: INCOMPRESSIBLE

A. Remove equation of state for density.

B. IF PHASE $\neq$ multicomponent

I. Remove holdup terms in mass balance equation.

END IF

END

END

The first two assumptions simply involve removing terms from the momentum and energy balances respectively. The assumption of well-mixed flow involves the generation of additional relationships equating the temperature and component densities of the holdup to the corresponding values at the outlet ports. Steady state behaviour implies that there is no variation of the system properties with time and hence the holdup terms in the mass and energy equations are removed (steady state behaviour is implicitly assumed for momentum conservation by the form of equation generated). Isothermal behaviour states that the temperature of the phase remains constant and also assumes that the calculation of the heat flux required to maintain this temperature is not needed. In effect, this reduces the number of unknown variables in the model by one and therefore one less equation is required. Since the temperature and hence, in effect, the enthalpy is known, no energy balance is required and it is removed from the model. Note that the assumption of isothermal behaviour is not permitted if the vessel contains closed ports (implying heat transfer) or if it is connected to another, nonisothermal vessel. Finally, if the phase is assumed to be incompressible (implying the density is independent of both pressure and temperature) the equation of state for density is removed and, for single component phases, the density holdup term, which is identically zero, is removed from the mass balance.

4.2.1.2 One Dimensional Models

One dimensional vessel submodels are generated in an exactly analogous manner to macroscopic models. The formal procedure is given by algorithm 4.10.

Algorithm 4.10 *Generate One Dimensional SubModel.*

1. FOR each PHASE DO

(a) Generate one dimensional mass balance. (Algorithm 4.11)

(b) Generate one dimensional momentum balance. (Algorithm 4.12)

(c) Generate one dimensional energy balance. (Algorithm 4.13)

(d) Generate equations of state. (Algorithm 4.7)

(e) Generate constitutive equations. (Algorithm 4.8)

(f) Simplify one dimensional model. (Algorithm 4.14)

END

It again consists of calls to three algorithms for the construction of the individual conservation equations and two calls to the shared algorithms already described above for the generation of the equations of state and physical and thermodynamic constitutive relationships. In this case, the generated equations describe the one dimensional variation of the system properties over the entire extent of the (single) phase filling the cylindrical volume of the containing vessel. As with algorithm 4.3 for macroscopic models, the final step in algorithm 4.10 is the application of the individual modelling assumptions declared in the physical description which serve to simplify the generated equations to a problem-specific form.

The first step in algorithm 4.10 is the generation of the mass balance. This is based on the prototype mass conservation equation (4.15) (or 4.16 for multicomponent systems) derived in section 2.4.1.2.

$$\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial z}\rho v_z = 0 \tag{4.15}$$

$$\frac{\partial}{\partial t}\rho_i + \frac{\partial}{\partial z}\rho_i v_z + \frac{\partial}{\partial z}J^e_{iz} + R_i = 0 \qquad i = 1, 2, \ldots, n \tag{4.16}$$

The formal procedure for its construction is given by algorithm 4.11. As before, for single component phases the application of this procedure results in a single equation (*cf.* 4.15) whilst for multicomponent phases the equation is constructed in vector form (*cf.* 4.16).

Algorithm 4.11 *Generate One Dimensional Mass Balance*

1. Add mass holdup term.

2. Add convective mass flux term.

3. IF `PHASE` = multicomponent

   (a) Add dispersive flux term.

   (b) Add reaction term.

   END IF

4. Equate to zero.

The algorithm is relatively straightforward. It begins with the construction of the holdup term which consists simply of the time derivative of the total or component density, depending on whether the phase contains a single component or multicomponent mixture. The next step is the addition of the convective flux term which comprises the axial spatial derivative of the product of the axial velocity and the appropriate (total or component) density. For multicomponent phases the final two steps add terms accounting for the spatial rate of change of the dispersive flux and generation of each component by chemical reaction(s). Finally, the sum of the generated terms is equated to zero.

The second step of algorithm 4.10 is the generation of the one dimensional momentum balance. The prototype equation is based on the scalar form (4.17) which, for regular cylindrical geometries, follows directly from the general (vector) form of the one dimensional momentum conservation equation (see section 2.4.1.2).

$$\frac{\partial}{\partial t}\rho v_z + \frac{\partial}{\partial z}\rho v_z v_z + \frac{\partial p}{\partial z} - \frac{\partial}{\partial z}\tau^e_{zz} + \frac{\tau_w P}{A} - \rho g_z = 0 \tag{4.17}$$

The procedure for its construction is given by algorithm 4.12.

Algorithm 4.12 *Generate One Dimensional Momentum Balance*

1. Add momentum holdup term.
2. Add convective momentum flux term.
3. Add pressure gradient term.
4. Subtract shear stress term.
5. Add wall friction term.
6. IF *Orientation* = vertical
   (a) Subtract gravitational term.
   END IF
7. Equate to zero.

It begins with the construction of the holdup term, which consists of the time derivative of the product of the total density and axial velocity. The next step adds the convective momentum flux, which comprises the axial spatial derivative of the product of the density and the square of the axial velocity. The next two steps are responsible for the construction of the spatial derivative of the total stress tensor, which is decomposed into the sum of the pressure gradient and the negation of the effective deviatoric (pseudo) stress tensor. This sign convention is used for consistency with most texts on fluid dynamics (but,

incidently, not Bird *et al.* [1960]). Step 5 adds a term accounting for the frictional drag on the cylindrical vessel walls per unit length and for vertically inclined vessels a term accounting for the gravitational acceleration is subtracted. Finally, the sum of the generated terms is equated to zero.

The third step in algorithm 4.10 is the generation of the one dimensional energy balance. This is based on the prototype one dimensional energy conservation equation (4.18) derived in section 2.4.1.2.

$$\frac{\partial}{\partial t}\rho\hat{E} + \frac{\partial}{\partial z}\rho v_z\hat{H} + \frac{\partial}{\partial}q_z^e - \frac{Q_w P}{A} - \frac{\partial}{\partial z}\tau_{zz}^e v_z - \rho v_z g_z = 0 \qquad (4.18)$$

The formal procedure for its construction is given by algorithm 4.13.

Algorithm 4.13 *Generate One Dimensional Energy Balance*

1. Add energy holdup term.
2. Add convective energy flux term.
3. Add dispersive energy flux term.
4. IF VESSEL wall is a CLOSED PORT
   (a) Add wall heat transfer term.

   END IF
5. Subtract shear work term.
6. IF *Orientation* = vertical
   (a) Add gravitational work term.

   END IF
7. Equate to zero.

It begins with the construction of the holdup term, which consists of the time derivative of the product of the total density and specific total energy, where the latter includes both the specific internal and kinetic energy. The next step adds the convective energy flux, which consists of the axial spatial derivative of the product of the total density, axial velocity and specific total enthalpy, where the latter includes both the specific enthalpy and kinetic energy. The next step adds a term accounting for the axial dispersive heat flux and is written simply as the axial spatial derivative of the axial heat flux. If the cylindrical vessel wall is declared as a closed port then step 4 subtracts the wall heat flux per unit length of the vessel, where the negative sign convention is adopted for consistency

with that used in thermodynamics. Steps 5 and 6 subtract the shear and gravitational work terms respectively. The shear work consists of the axial spatial derivative of the product of the deviatoric pseudo stress tensor and the axial velocity and, for vertically oriented vessels, a gravitational work term is subtracted comprising the product of the total density, velocity and vertical component of gravitational acceleration. The final step equates the sum of the generated terms to zero.

The next two steps of algorithm 4.10 are responsible for the generation of the equations of state and physical and thermodynamic constitutive relationships. The equations of state are generated by algorithm 4.7 (described above) in the same way as for macroscopic vessel submodels. The formal procedure for the generation of the constitutive equations is given by algorithm 4.8 (also described above). For one dimensional models the constitutive relationships comprise not only those equations generated for macroscopic vessel submodels (4.9 - 4.14) but also additional equations for the effective dispersive mass (for multicomponent systems) and heat flux terms (4.19 and 4.20 respectively), the pseudo Newtonian shear stress equation (4.21) and the total energy and total enthalpy relations (4.22 and 4.23 respectively).

$$J^e_{zi} = -D^e_i \frac{\partial c_i}{\partial z} \qquad i = 1, 2, \ldots, n \tag{4.19}$$

$$q^e_z = -k_e \frac{\partial T}{\partial z} \tag{4.20}$$

$$\tau^e_{zz} = C\mu \frac{\partial v_z}{\partial z} \tag{4.21}$$

$$\hat{E} = \hat{U} + \frac{1}{2} v_z^2 \tag{4.22}$$

$$\hat{H} = \hat{h} + \frac{1}{2} v_z^2 \tag{4.23}$$

Note that for simplicity, following the approach used in commercial CFD codes, the constitutive relationship for the effective dispersive mass flux is based on a Fickian-type linear equation rather than a more rigorous nonlinear model such as the Stefan-Maxwell equations.

The final step in the generation of the one dimensional vessel submodel is the simplification of the general model to a specific form based on the declared modelling assumptions. The formal procedure is given by algorithm 4.14.

Algorithm 4.14 *Simplify One Dimensional Model*

1. FOR each ASSUMPTION DO

(a) SWITCH: AssumptionType

i. CASE: `FRICTIONLESS`

A. Remove friction term from momentum balance.

ii. CASE: `PLUG_FLOW`

A. Remove dispersive flux terms from balance equations.

B. Remove shear work terms from balance equations.

C. Remove constitutive laws for dispersive flux terms.

iii. CASE: `QUASI_STEADY_STATE`

A. Remove momentum holdup term from momentum balance.

iv. CASE: `STEADY_STATE`

A. Remove holdup terms from balance equations.

v. CASE: `ISOTHERMAL`

A. Remove energy balance.

vi. CASE: `INCOMPRESSIBLE`

A. Remove equation of state for density.

B. Modify constant in pseudo Newtonian shear stress equation.

C. IF `PHASE` $\neq$ multicomponent

I. Remove holdup terms in mass balance equation.

END IF

END

END

The first assumption simply involves removing the wall friction terms from the momentum conservation equation. The second assumption of plug-flow implies that there is no back-mixing and hence that there are no dispersive flux contributions to the overall transport of each conserved quantity. Therefore, with this assumption, all dispersive terms and their associated constitutive equations are removed from the model. The final four assumptions all correspond exactly to the equivalent assumptions in macroscopic vessel submodels. Quasi-steady state behaviour assumes that the momentum holdup is negligible whilst steady state indicates that the system properties do not vary with time and hence that the mass, energy and momentum holdups are all zero. Isothermal behaviour states that the temperature of the phase remains constant. Again this reduces the number of variables in the model by one and therefore, with one less equation required, the energy balance is removed. As before, the assumption of isothermal behaviour is not permitted if the vessel contains closed ports (implying heat transfer) or if it is connected to another nonisothermal vessel. Finally, if the phase is assumed to be incompressible

the equation of state for density is removed and, for single component phases, the density holdup term is removed from the mass balance. In addition, the numerical constant in the constitutive equation for the pseudo shear stress (4.21) is changed (from 4/3 to 2) since for incompressible fluids there is no contribution from the divergence of velocity.

4.2.1.3 Two Dimensional Models

Two dimensional vessel submodels are generated in a similar manner to macroscopic and one dimensional models. The formal procedure is given by algorithm 4.15.

Algorithm 4.15 *Generate Two Dimensional SubModel.*

1. FOR each PHASE DO
   (a) Generate two dimensional mass balance. (Algorithm 4.16)
   (b) Generate two dimensional linear momentum balances. (Algorithm 4.17)
   (c) Generate two dimensional energy balance. (Algorithm 4.20)
   (d) Generate equations of state. (Algorithm 4.7)
   (e) Generate constitutive equations. (Algorithm 4.8)
   (f) Simplify two dimensional model. (Algorithm 4.21)
   (g) Generate internal boundary conditions. (Algorithm 4.22)

   END

It again consists of calls to three algorithms for the construction of the individual conservation equations and two calls to the shared algorithms already described above for the generation of the equations of state and physical and thermodynamic constitutive relationships. As with one dimensional vessel submodels, these equations apply over the entire extent of the phase and hence, since only single phase systems are considered here, over the entire volume of the containing vessel. Following construction of the general form of the model equations the next step is the application of the individual modelling assumptions declared in the physical description. The final step in algorithm 4.15 is the generation of the internal boundary conditions which, for the cylindrical geometries considered here, comprise conditions at symmetry lines and vessel walls. This step is carried out after the model simplification to ensure that only those conditions that are required are added to the model.

The first step in algorithm 4.15 is the generation of the mass balance. This is based on the prototype two dimensional conservation equation (4.24) (or 4.25 for multi-component phases) given in section 2.4.1.1.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial z}\rho v_z + \frac{1}{r}\frac{\partial}{\partial r} r \rho v_r = 0 \tag{4.24}$$

$$\frac{\partial \rho_i}{\partial t} + \frac{\partial}{\partial z}\rho_i v_z + \frac{1}{r}\frac{\partial}{\partial r} r \rho_i v_r + \frac{\partial}{\partial z} J^e_{zi} + \frac{1}{r}\frac{\partial}{\partial r} r J^e_{ri} + R_i = 0 \qquad i = 1, 2, \ldots, n \qquad (4.25)$$

The formal procedure for its construction is given by algorithm 4.16. As before, for single component phases the application of this procedure results in a single equation (*cf.* 4.24) whilst for multicomponent phases the equation is constructed in vector form (*cf.* 4.25).

Algorithm 4.16 *Generate Two Dimensional Mass Balance*

1. Add mass holdup term.
2. Add convective mass flux terms.
3. IF `PHASE` = multicomponent
   (a) Add dispersive flux terms.
   (b) Add reaction term.

   END IF
4. Equate to zero.

It begins with the construction of the holdup term which consists simply of the time derivative of either the total or component density, depending on whether the phase contains a single component or multicomponent mixture. The next step is the addition of the two convective flux terms which describe the two dimensional divergence of the product of the density and velocity vector. The first consists of the axial spatial derivative of the product of the axial velocity and the (total or component) density whilst the second consists of the radial spatial derivative of the product of the radial velocity and the density together with the appropriate radial coordinate multipliers which account for the cylindrical coordinate system. For multicomponent phases the final two steps add terms accounting for the axial and radial spatial rate of change of the dispersive flux and the generation of each component by chemical reaction(s). Finally, the sum of the generated terms is equated to zero.

The next step of algorithm 4.15 is the generation of the two orthogonal components of the two dimensional linear momentum balances. These are constructed separately as described by algorithm 4.17.

Algorithm 4.17 *Generate Two Dimensional Linear Momentum Balances*

1. Generate axial component of linear momentum balance. (Algorithm 4.18)
2. Generate radial component of linear momentum balance. (Algorithm 4.19)

The first step of algorithm 4.17 is the generation of the axial component of the linear momentum equation. This is based on the prototype axial momentum conservation equation (4.26) given in section 2.4.1.1.

$$\frac{\partial}{\partial t}\rho v_z + \frac{\partial}{\partial z}\rho v_z v_z + \frac{1}{r}\frac{\partial}{\partial r} r \rho v_r v_z + \frac{\partial p}{\partial z} - \frac{\partial}{\partial z}\tau^e_{zz} - \frac{1}{r}\frac{\partial}{\partial r} r \tau^e_{rz} - \rho g_z = 0 \qquad (4.26)$$

The formal procedure for its construction is given by algorithm 4.18.

Algorithm 4.18 *Generate Axial Component of Two Dimensional Momentum Balance*

1. Add axial component of momentum holdup term.
2. Add axial component of convective momentum flux terms.
3. Add axial component of pressure gradient term.
4. Subtract axial component of shear stress terms.
5. IF *Orientation* = vertical
   (a) Subtract gravitational term.

   END IF
6. Equate to zero.

It begins with the construction of the holdup term, which consists of the time derivative of the product of the density and axial velocity component. The next step is the addition of the two convective terms, which describe the divergence of the product of the total density, the axial velocity and the velocity vector. The first term consists of the axial spatial derivative of the product of the density and square of the axial velocity component whilst the second comprises the radial spatial derivative of the product of the density and axial and radial velocity components together with the appropriate radial coordinate multipliers. The third and fourth steps add the divergence of the total shear stress terms. These are decomposed into the sum of the axial pressure gradient and the negation of the axial and radial gradients of the axial components of the two dimensional deviatoric stress tensor. For vertically oriented vessels the fifth step subtracts the gravitational term which consists of the product of the total density and axial component of gravitational acceleration. Finally the sum of the generated terms is equated to zero.

The second step of algorithm 4.17 is the generation of the radial component of the linear momentum equation. This is based on the prototype radial momentum conservation equation (4.27) given in section 2.4.1.1.

$$\frac{\partial}{\partial t}\rho v_r + \frac{\partial}{\partial z}\rho v_z v_r + \frac{1}{r}\frac{\partial}{\partial r}r\rho v_r v_r + \frac{\partial p}{\partial r} - \frac{\partial}{\partial z}\tau^e_{rz} - \frac{1}{r}\frac{\partial}{\partial r}r\tau^e_{rr} + \frac{\tau^e_{\theta\theta}}{r} = 0 \tag{4.27}$$

The formal procedure for its construction is given by algorithm 4.19.

Algorithm 4.19 *Generate Radial Component of Two Dimensional Momentum Balance*

1. Add radial component of momentum holdup term.
2. Add radial component of convective momentum flux terms.
3. Add radial component of pressure gradient term.
4. Subtract radial component of shear stress terms.
5. Equate to zero.

The procedure is very similar to that for the generation of the axial momentum balance. It begins with the construction of the holdup term, which consists of the time derivative of the product of the density and radial velocity component. The next step adds the two convective terms, which describe the divergence of the product of the total density, radial velocity and the two dimensional velocity vector. The first consists of the axial spatial derivative of the product of the density and axial and radial velocity components whilst the second comprises the radial spatial derivative of the product of the density and the square of the radial velocity components, together with the appropriate radial coordinate multipliers. The third and fourth steps add the divergence of the total shear stress terms. These are decomposed into the sum of the radial pressure gradient and the negation of the axial and radial gradients of the radial components of the two dimensional deviatoric stress tensor (which include contributions from $\tau^e_{\theta\theta}$). Finally, the sum of the generated terms is equated to zero.

The third step of algorithm 4.15 is the construction of the energy balance. This is based on the prototype two dimensional energy conservation equation (4.28) given in section 2.4.1.1.

$$\frac{\partial}{\partial t}\rho\hat{E} + \frac{\partial}{\partial z}\rho v_z\hat{H} + \frac{1}{r}\frac{\partial}{\partial r}r\rho v_r\hat{H} + \frac{\partial}{\partial}q^e_z + \frac{1}{r}\frac{\partial}{\partial r}rq^e_r$$
$$-\frac{\partial}{\partial z}(\tau^e_{zz}v_z + \tau^e_{rz}v_r) - \frac{1}{r}\frac{\partial}{\partial r}r(\tau^e_{rz}v_z + \tau^e_{rr}v_r) - \rho v_z g_z = 0 \tag{4.28}$$

The formal procedure for its construction is given by algorithm 4.20.

Algorithm 4.20 *Generate Two Dimensional Energy Balance*

1. Add energy holdup term.

2. Add convective energy flux terms.

3. Add dispersive energy flux terms.

4. Subtract shear work term.

5. IF *Orientation* = vertical

   (a) Subtract gravitational work term.

   END IF

6. Equate to zero.

It begins with the construction of the energy holdup term, which consists of the time derivative of the product of the total density and the specific total energy, where the latter comprises the sum of the specific internal and kinetic energies. The next step adds the two convective energy flux terms, which describe the two dimensional divergence of the product of the density, total enthalpy and velocity vector. The first consists of the axial spatial derivative of the product of the total density, axial velocity and total enthalpy whilst the second consists of the radial spatial derivative of the product of the density, radial velocity and total enthalpy, together with the appropriate radial coordinate multipliers which account for the cylindrical coordinate system. The third step is responsible for the construction of two terms accounting for the divergence of the dispersive energy flux. These consist of the axial and radial gradients of the axial and radial heat flux components respectively and for the latter again include the radial coordinate multipliers which account for the cylindrical coordinate system. The fourth step adds two shear work terms which describe the divergence of the dot product of the effective deviatoric stress tensor with the velocity vector and, for vertically oriented vessels, the fifth step also adds a gravitational work term consisting of the product of the density and axial components of velocity and gravitational acceleration. The final step equates the sum of the generated terms to zero.

The next two steps of algorithm 4.15 are responsible for the generation of the equations of state and physical and thermodynamic constitutive relationships. The equations of state are generated by algorithm 4.7 (described above) in the same way as for macroscopic and one dimensional vessel submodels. The formal procedure for the generation of the constitutive equations is given by algorithm 4.8 (also described above). For two dimensional models the constitutive relationships consist of the equations generated for macroscopic vessel submodels (4.9 - 4.14) and in addition equations for the two components of the dispersive mass (4.29 and 4.30) (for multicomponent phases) and heat flux terms (4.31 and 4.32), the effective Newtonian shear stress equations (4.33 - 4.36) and the two dimensional total energy and total enthalpy relations (4.37 and 4.38 respectively).

$$J_{iz}^e = -D_i^e \frac{\partial c_i}{\partial z} \qquad i = 1, 2, \ldots, n \tag{4.29}$$

$$J_{ir}^e = -D_i^e \frac{\partial c_i}{\partial r} \qquad i = 1, 2, \ldots, n \tag{4.30}$$

$$q_z^e = -k_e \frac{\partial T}{\partial z} \tag{4.31}$$

$$q_z^e = -k_e \frac{\partial T}{\partial r} \tag{4.32}$$

$$\tau_{zz}^e = \mu_e \left[2\frac{\partial v_z}{\partial z} - \frac{2}{3}\left(\frac{\partial v_z}{\partial z} + \frac{1}{r}\frac{\partial}{\partial r} r v_r \right) \right] - \frac{2}{3}\rho k \tag{4.33}$$

$$\tau_{rz}^e = \mu_e \left[\frac{\partial v_z}{\partial r} + \frac{\partial v_r}{\partial z} \right] \tag{4.34}$$

$$\tau_{rr}^e = \mu_e \left[2\frac{\partial v_r}{\partial r} - \frac{2}{3}\left(\frac{\partial v_z}{\partial z} + \frac{1}{r}\frac{\partial}{\partial r} r v_r \right) \right] - \frac{2}{3}\rho k \tag{4.35}$$

$$\tau_{\theta\theta}^e = \mu_e \left[\frac{v_r}{r} - \frac{2}{3}\left(\frac{\partial v_z}{\partial z} + \frac{1}{r}\frac{\partial}{\partial r} r v_r \right) \right] - \frac{2}{3}\rho k \tag{4.36}$$

$$\hat{E} = \hat{U} + \frac{1}{2}(v_z v_z + v_r v_r) + k \tag{4.37}$$

$$\hat{H} = \hat{h} + \frac{1}{2}(v_z v_z + v_r v_r) + k \tag{4.38}$$

Note that, as with one dimensional submodels, linear Fickian-type equations are used for the effective dispersive mass flux components rather than more rigorous nonlinear forms. Also, the effective shear stress equations include both laminar and turbulent contributions and, accordingly, are based on the generalised compressible form of the Boussinesq approximation (see chapter 2), whilst the total energy and total enthalpy relations include the kinetic energy of turbulence.

The sixth step of algorithm 4.15 is responsible for the simplification of the general two dimensional model generated by the first five steps to a specific form based on the declared model assumptions. The formal simplification procedure is described by algorithm 4.21.

Algorithm 4.21 *Simplify Two Dimensional Model*

1. FOR each `ASSUMPTION` DO
   (a) SWITCH: AssumptionType

i. CASE: LAMINAR

A. Replace effective flux terms in balance equations by their laminar counterparts.

B. Remove constitutive equations for turbulent flux terms.

C. Remove kinetic energy of turbulence from total energy and total enthalpy constitutive equations.

ii. CASE: QUASI_STEADY_STATE

A. Remove momentum holdup term from momentum balance.

iii. CASE: STEADY_STATE

A. Remove holdup terms from balance equations.

iv. CASE: ISOTHERMAL

A. Remove energy balance.

v. CASE: INCOMPRESSIBLE

A. Remove equation of state for density.

B. Simplify Newtonian normal shear stress equations.

C. IF PHASE $\neq$ multicomponent

I. Remove holdup terms in mass balance equation.

END IF

END

END

The first assumption of laminar flow involves removing the terms and variables which describe turbulent fluxes. This simplification requires replacing the effective flux and shear stresses in the momentum and energy balance equations with their (exact) laminar counterparts and removing both their associated constitutive equations and also the kinetic energy of turbulence from the total energy and total enthalpy equations. Quasi-steady state behaviour again assumes that the momentum holdup is negligible whilst the steady state assumption, as with the other types of vessel submodels, indicates that the system properties do not vary with time and hence that the mass, energy and momentum holdup terms are all zero. Isothermal behaviour states that the temperature of the phase remains constant and again the energy balance is removed from the model. As before, the assumption of isothermal behaviour is not permitted if the vessel contains closed ports (implying heat transfer) or if it is connected to another nonisothermal vessel. Finally, again as before, if the phase is assumed to be incompressible the equation of state for density is removed and, for single component phases, the density holdup term is removed from the mass balance. In addition, the constitutive equations for the Newtonian normal shear

stresses are simplified by removing the contribution from the divergence of the velocity vector which is identically zero.

The final step of algorithm 4.15 is the generation of the internal boundary conditions at symmetry lines and vessel walls. Unlike the other two dimensional equations which apply over the entire extent of the containing vessel, the boundary conditions apply only over specific portions of the vessel surfaces as given by the declared vessel and/or port geometries. This step is carried out after the application of the modelling assumption to ensure that only the required conditions are generated (for example, the symmetry condition for density is replaced by a symmetry condition for pressure in incompressible models).

Algorithm 4.22 *Generate Internal Boundary Conditions*

1. Add symmetry conditions.

2. Add wall boundary conditions.

For the cylindrical vessels considered here symmetry conditions simply equate the radial velocity and radial gradients of temperature, axial velocity and either density or pressure (for compressible or incompressible phases respectively) to zero at the centreline (*cf.* 4.39 and 4.40).

$$v_r = 0 \qquad @r = 0,\ L_l \leq z \leq L_u \tag{4.39}$$

$$\frac{\partial x}{\partial r} = 0 \qquad @r = 0,\ L_l \leq z \leq L_u \tag{4.40}$$

Wall boundary conditions describe the principles of no-slip at the vessel walls (4.41), no flow through (4.42) and no dispersive mass flux through the vessel walls (4.43) and, if insulated (*i.e.* with no heat transfer), no heat flux through the vessel walls (4.44) (for walls where there is heat transfer the relevant equations are generated in the connection submodel). Note that with the simple geometrical descriptions considered here, closed vessels walls are only permitted on the cylindrical (curved) surfaces (rather than at the endpoints of the axial domain).

$$v_z = 0 \qquad @r = R,\ L_l \leq z \leq L_u \tag{4.41}$$

$$v_r = 0 \qquad @r = R,\ L_l \leq z \leq L_u \tag{4.42}$$

$$J^e_{ri} = 0 \qquad i = 1, 2, \ldots, n \qquad @r = R,\ L_l \leq z \leq L_u \tag{4.43}$$

$$q^e_r = 0 \qquad @r = R,\ L_l \leq z \leq L_u \tag{4.44}$$

4.2.2 Reservoir Submodels

Reservoir submodels are generated for each reservoir declared in the process description. They are based on a reduced form of macroscopic vessel submodel and consist simply of equations of state and physical and thermodynamic relationships which serve to define the state of either the system boundaries (such as feed or product streams) or other entities which are not modelled explicitly (such as the environment or process utilities). The formal procedure for the construction of the reservoir submodels is given by algorithm 4.23.

Algorithm 4.23 *Generate Reservoir Submodels*

1. FOR each `RESERVOIR` DO

   (a) IF connected to a `VESSEL` via an `OPEN PORT`

      i. Generate Equations of State. (Algorithm 4.7)

      ii. Generate constitutive equations. (Algorithm 4.8)

   (b) ELSE connected to a `VESSEL` via a `CLOSED PORT`

      i. (Temperature variable only.)

   END IF

END

The procedure constructs two types of reservoir submodels: those for reservoirs connected to vessels via open ports and those for reservoirs connected via closed ports. In general, the former consist of the same set of equations of state and constitutive relationships as for macroscopic submodels described above (*cf.* 4.5 - 4.8 and 4.9 - 4.14 respectively) and are generated using the same algorithms (4.7 and 4.8). The specific form of these equations may be simplified somewhat depending on the assumptions declared for the vessels to which they are connected. For example, reservoirs connected to vessels containing incompressible fluids include no equation of state for density and those connected to vessels containing single component phases include no multicomponent equations (such as mass and mole fractions). The latter contain no equations but merely comprise a single (temperature) variable which defines the reservoir temperature and is used in the construction of the film heat transfer equations in the connection submodel (below).

4.2.3 Connection Submodel

The final step in the model building procedure (algorithm 4.1) is the generation of the overall connection submodel. In general, this consists of one or more equations for each declared reservoir-vessel or inter-vessel connection, where the number and form of the

equations depend on the interface type (open or closed), the bulk flow direction and the upstream and downstream dimensionality and flow characteristics. The formal procedure for the construction of the connection submodel is given by algorithm 4.24. Note that this procedure is again only valid for the restricted scope of process system (particularly with regard to its geometry) considered here (see section 2.7).

Algorithm 4.24 *Generate Connection Submodel*

1. FOR each `CONNECTION` DO
   (a) Get upstream node.
   (b) Get downstream node.
   (c) IF `OPEN CONNECTION`
      i. Generate `OPEN CONNECTION` equation(s). (Lookup table 4.1)
   (d) ELSE
      i. Generate `CLOSED CONNECTION` equation(s). (Lookup table 4.2)
   END IF

END

For each connection the procedure consists of four main steps. Steps 1.(a) and 1.(b) simply retrieve the submodels associated with the individual reservoirs and/or vessels (nodes) between which the connection is made. Steps 1.(c) and 1.(d) then classify the connection into open or closed types, deduced from the corresponding port types, and construct an appropriate set of equations based primarily on this classification.

The equations for open connections are summarised in lookup table 4.1. The upstream nodes (node 1) are shown in the rows and the downstream nodes (node 2) in the columns. For open connections five key types of node are important: reservoir submodels (R), macroscopic vessel submodels (M), one dimensional vessel submodels with plug flow (1-D (PF)), one dimensional vessel submodels with dispersed flow (1-D (DF)) and two dimensional vessel submodels (2-D) (which always describe dispersed flows). The table then gives the appropriate number and form of prototype equations for each possible interconnection.

For reservoir submodels or macroscopic vessel submodels upstream (node 1), these equations simply equate intensities x across the connection (cells aA - aE and bA-bE), where x represents total or component density, temperature, velocity and, for incompressible systems, pressure. Note that for two dimensional vessel submodels downstream, due to the jump in dimensionality across the connection, flat radial distributions of each intensity and zero radial velocity are assumed at the interface (cells aE and bE). For one dimensional vessel submodels with plug flow upstream the same equations also apply in all

cases, except for one dimensional dispersed flow downstream (cell cD) where Danckwerts inlet-type boundary conditions are generated for mass, energy and momentum describing discontinuities in each intensity across the interface. Note however that, for incompressible systems, the upstream and downstream pressures are equated. For one dimensional vessel submodels with dispersed flow upstream (cells dA - dE) the same equations are generated as for reservoir and macroscopic submodels upstream. In addition the mass, energy and momentum fluxes are set to zero at the interface in all cases except for another one dimensional dispersed flow vessel submodel downstream where equations describing the continuity of the fluxes are generated (cell dD). The zero flux equations are, in effect, equivalent to the Danckwerts exit (or outflow) boundary conditions since the (effective) proportionality constants (dispersion coefficient, thermal conductivity and viscosity) are all nonzero. Finally, for two dimensional vessel submodels upstream the equations again describe continuity of each intensity (though in this case the downstream values are calculated as the average of the upstream values) and use the two dimensional equivalent of Danckwerts exit conditions for all cases (cells eA - eD) except for a two dimensional vessel submodel downstream (cell eE), where equations describing continuity of intensity, flux and velocity gradients are generated.

The equations for closed connections are summarised in lookup table 4.2. Again the upstream nodes (node 1) are shown in the rows and the downstream nodes (node 2) in the columns. Note however that for closed connections there is no bulk flow and hence the direction of the connection (*i.e.* which side is node 1 and which is node 2) does not alter the form of the boundary equation but merely defines the sign convention. For closed connections four key types of node are important: reservoir submodels (R), macroscopic vessel submodels (M), one dimensional vessel submodels (1-D) and two dimensional vessel submodels (2-D). The presence or absence of dispersion (in one dimensional vessel submodels) does not affect the equation form. Again the table gives the appropriate number and form of prototype equations for each possible (closed) interconnection.

All equations are based primarily around the notion of conservation of heat flux across the connection but where additional relations are needed may also include conditions describing thermodynamic equilibrium (continuity of temperature). Note that for connections involving macroscopic or one dimensional vessel submodels the (average) heat flux is described in terms of a film or overall linear heat transfer model. Also, for connections between interfaces with a jump in dimensionality (such as between a one or two dimensional vessel submodel and either a reservoir submodel or a macroscopic vessel submodel) the equations describing continuity of flux are expressed in terms of averaged quantities to ensure that conservation is satisfied.

The equations in tables 4.1 and 4.2 reflect the theoretical discussion of the required number and type of boundary conditions given in chapter 2 and show their dependence on the interface type, bulk flow direction and upstream/downstream dimensionality

and mixing characteristics. Note that these equations may be identified as either simple matching equations which relate quantities between vessels or boundary conditions on the edges of the spatial domains defined by the vessels. Thus, for the latter, the tables show not only the equation form but also the boundaries over which they apply, based on the simple geometrical descriptions employed here (see section 2.7).

4.3 Summary

In this chapter a model building procedure has been presented which permits the automatic generation of mathematical models of process systems from purely physical descriptions. The procedure is based on the rigorous theoretical approach to process modelling presented in chapter 2 and involves the systematic application of model generation algorithms to the physical description of a process system declared in terms of the process representation and modelling language defined in chapter 3.

In the next chapter details of the implementation of the process representation and the model building procedure in the computer program gMaGiC (*g*eneral *M*odel *G*eneration *C*ode) are given.

Node 2 (Downstream)

Node 1 (Upstream)		A: R_2	B: M_2	C: 1-D_2 (PF)	D: 1-D_2 (DF)	E: 2-D_2
a	R_1	–	$x_1 = x_2$	$x_1 = x_2$ †	$x_1 = x_2$ †	$x_1 = x_2(L_l, r)$ ‡ $v_{r2}(L_l, r) = 0$ ‡
b	M_1	$x_1 = x_2$	$x_1 = x_2$	$x_1 = x_2$ †	$x_1 = x_2$ †	$x_1 = x_2(L_l, r)$ ‡ $v_{r2}(L_l, r) = 0$ ‡
c	1-D_1 (PF)	$x_1 = x_2$	$x_1 = x_2$	$x_1 = x_2$ †	$(p_1 = p_2)$ † $v_1\rho_{i1} = v_2\rho_{i2} + J^e_{i2}$ † $\rho_1 v_1 h_1 = \rho_2 v_2 h_2 + q^e_2$ † $\rho_1 v_1 v_1 = \rho_2 v_2 v_2 - \tau^e_{zz2}$ †	$x_1 = x_2(L_l, r)$ ‡ $v_{r2}(L_l, r) = 0$ ‡
d	1-D_1 (DF)	$x_1 = x_2$ $J^e_{i1} = 0$ † $\tau^e_{zz1} = 0$ † $q^e_1 = 0$ †	$x_1 = x_2$ $J^e_{i1} = 0$ † $\tau^e_{zz1} = 0$ † $q^e_1 = 0$ †	$x_1 = x_2$ † $J^e_{i1} = 0$ † $\tau^e_{zz1} = 0$ † $q^e_1 = 0$ †	$x_1 = x_2$ † $J^e_{i1} = J^e_{i2}$ † $\tau^e_{zz1} = \tau^e_{zz2}$ † $q^e_1 = q^e_2$ †	$x_1 = x_2(L_l, r)$ ‡ $J^e_{i1} = 0$ † $\tau^e_{zz1} = 0$ † $q^e_1 = 0$ † $v_{r2}(L_l, r) = 0$ ‡
e	2-D_2	$\frac{2}{R^2}\int_R rx(L_u, r)\,dr = x$ $J^e_{i1}(L_u, r) = 0$ ‡ $q^e_1(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{r1}(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{z1}(L_u, r) = 0$ ‡	$\frac{2}{R^2}\int_R rx(L_u, r)\,dr = x$ $J^e_{i1}(L_u, r) = 0$ ‡ $q^e_1(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{r1}(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{z1}(L_u, r) = 0$ ‡	$\frac{2}{R^2}\int_R rx(L_u, r)\,dr = x$ † $J^e_{i1}(L_u, r) = 0$ ‡ $q^e_1(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{r1}(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{z1}(L_u, r) = 0$ ‡	$\frac{2}{R^2}\int_R rx(L_u, r)\,dr = x$ † $J^e_{i1}(L_u, r) = 0$ ‡ $q^e_1(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{r1}(L_u, r) = 0$ ‡ $\frac{\partial}{\partial z}v_{z1}(L_u, r) = 0$ ‡	$x_1(L_u, r) = x_2(L_l, r)$ ‡ $J^e_{i1}(L_u, r) = J^e_{i2}(L_l, r)$ ‡ $q^e_1(L_u, r) = q^e_2(L_l, r)$ ‡ $\frac{\partial}{\partial z}v_{r1}(L_u, r) = \frac{\partial}{\partial z}v_{r2}(L_l, r)$ ‡ $\frac{\partial}{\partial z}v_{z1}(L_u, r) = \frac{\partial}{\partial z}v_{z2}(L_l, r)$ ‡

† @$z = L_l$ and/or $z = L_u$

‡ @$z = L_l$ and/or $z = L_u$, $R_l \leq r \leq R_u$

Table 4.1: Lookup Table for Connections via Open Ports

Node 2

Node 1		A: R_2	B: M_2	C: 1-D_2	D: 2-D_2
a	R_1	–	$Q_2 = h_2(T_1 - T_2)$	$Q_2(z) = h_2(z)(T_1 - T_2(z))$	$T_1 = T_2(z, R)$ †
b	M_1	$Q_1 = h_1(T_2 - T_1)$	$1/h = 1/h_1 + 1/h_2$ $Q_1 = h(T_2 - T_1)$ $Q_1 = -Q_2$	$1/h(z) = 1/h_1 + 1/h_2(z)$ $Q_2(z) = h(z)(T_1 - T_2(z))$ $Q_1 = -\frac{1}{L}\int_L Q_2(z)\,dz$	$q^e_{r2}(z, R) = h_1(T_1 - T_2(z, R))$ † $Q_1 = -\frac{1}{L}\int_L q^e_{r2}(z, R)\,dz$
c	1-D_1	$Q_1(z) = h_1(z)(T_2 - T_1(z))$	$1/h(z) = 1/h_1(z) + 1/h_2$ $Q_1(z) = h(z)(T_2 - T_1(z))$ $\frac{1}{L}\int_L Q_1(z)\,dz = -Q_2$	$1/h(z) = 1/h_1(z) + 1/h_2$ $Q_2(z) = h(z)(T_2(z) - T_1(z))$ $Q_1(z) = -Q_2(z)$	$Q_1(z) = h_1(z)(T_2(z, R) - T_1(z))$ $q^e_{r2}(z, R) = -Q_1(z)$ †
d	2-D_1	$T_1(z, R) = T_2$	$q^e_{r1}(z, R) = h_2(T_2 - T_1(z, R))$ † $\frac{1}{L}\int_L q^e_{r1}(z, R)\,dz = -Q_2$	$q^e_{r1}(z, R) = h_2(z)(T_2(z) - T_1(z, R))$ † $q^e_{r1}(z, R) = -Q_2(z)$	$T_1(z, R) = T_2(z, R)$ † $q^e_{r1}(z, R) = -q^e_{r2}(z, R)$ †

† @$r = R_l$ and/or $r = R_u$, $L_l \leq x \leq L_u$

Table 4.2: Lookup Table for Connections via Closed Ports

Chapter 5

Implementation

In the previous two chapters a formal process representation and modelling language and a systematic model building procedure have been defined, supporting the generation of mathematical models of process systems from a purely physical description.

This chapter presents details of the implementation of this approach in gMaGiC (*g*eneral *M*odel *G*eneration *C*ode), a computer program written in ISO C (see, for example, Sexton [1996]). gMaGiC takes a text file containing the physical description of a process system declared in terms of the modelling language and generates an appropriate mathematical model in the format of a specific target language. The current version has been ported to both UNIX and IBM-PC platforms and generates mathematical models in the format of the *gPROMS* modelling language [Barton and Pantelides, 1994; Oh and Pantelides, 1996].

The chapter begins with an overview of the software architecture used in gMaGiC and then discusses each of the constituent parts in detail.

5.1 Software Architecture

The software architecture used in gMaGiC is based on the structure of a compiler and comprises three main parts: the translator, the model generation engine and the code generator (see, for example, Aho and Ullman [1977]) . Intermediate storage is provided by the dynamic allocation of physical and mathematical symbol tables, defined as abstract data structures in external libraries. A schematic diagram of the software architecture, which is close to that used by Cho [1997], is given in figure 5.1.

The translator takes an input file containing a physical description of a process system declared in terms of the process modelling language defined in chapter 3 and checks for syntactical and semantic correctness. If the translation is successful, the physical description is stored in the physical symbol table. If not, errors in the input are reported and execution of the model generation procedure is aborted. The physical symbol table is used by the model generation engine to construct the mathematical model, based on the model building algorithms developed in chapter 4, and an internal representation of the generated data is stored in the mathematical symbol table. Finally the mathematical

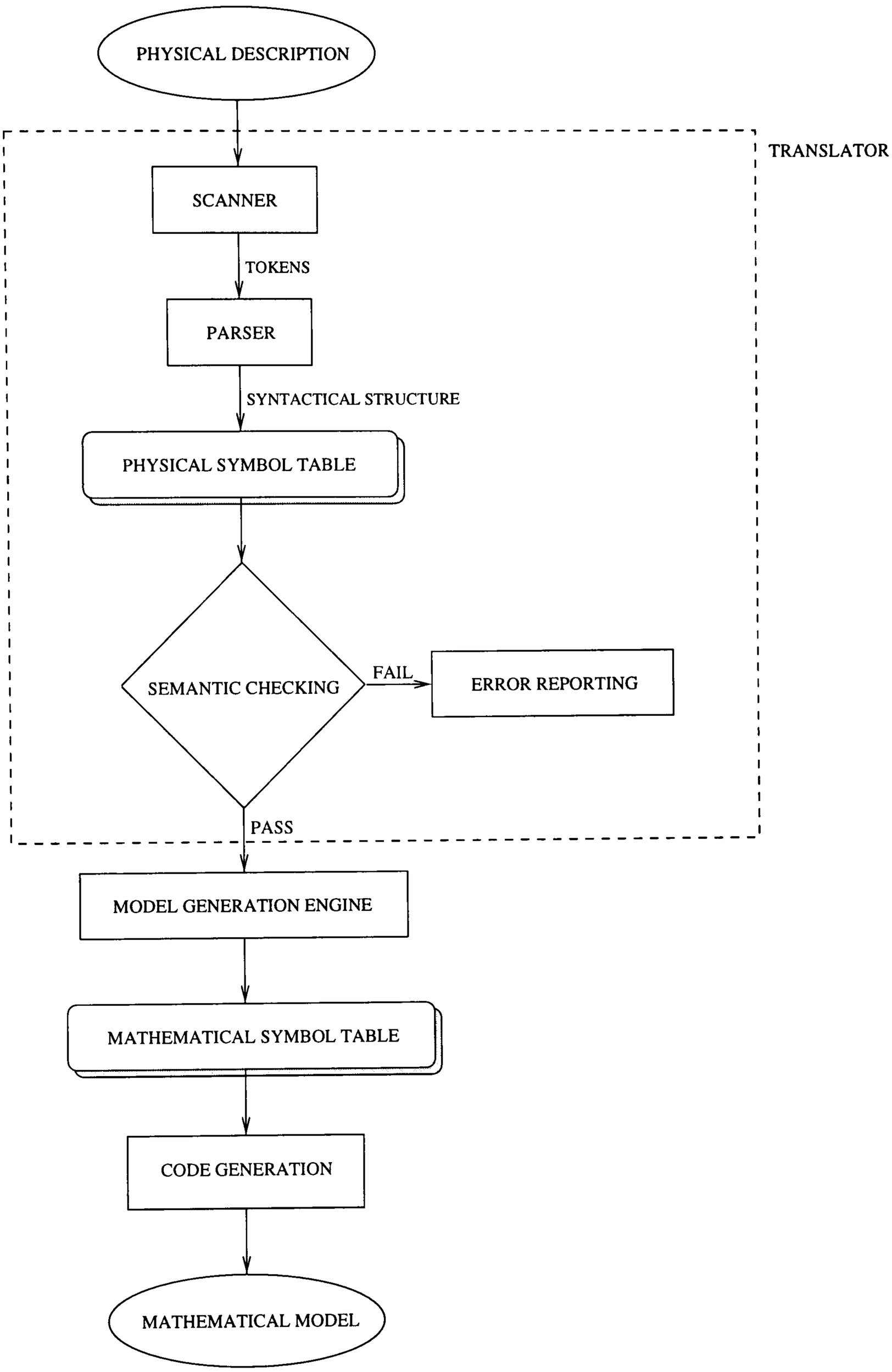


Figure 5.1: Software Architecture

symbol table is employed by the code generator to convert the internal representation into a specific target language.

5.2 The Translator

The main task of the translator is the validation of the syntactic and semantic structure of the input, declared in terms of the process modelling language. In addition, the translator is responsible for the construction of an internal representation of the physical description in memory for subsequent use by the model generation engine. The implementation of the translator is divided into three parts: the scanner or lexical analyser, the parser and the semantic checking subroutines. These are discussed below.

5.2.1 The Scanner

The scanner takes the character stream from the input device (in this case a text file) and divides it up into lexical tokens. These include keywords, string literals, constants, operators and punctuation symbols. The tokens are passed to the parser for further processing.

The implementation of the scanner in this work is carried out using the GNU project's flex (*F*ast *Lex*ical Analyser Generator) [Levine *et al.*, 1992]. Briefly, GNU (*G*nu's *N*ot *U*nix) is the project of the Free Software Foundation[1] and is an attempt to create a UNIX-like operating system with source code available publicly (GNU is not public domain, but is freely available and has a licence intended to keep it so). Flex is a software tool for producing a scanner as a C code subroutine from a set of descriptions of possible tokens based on regular expressions. The code generated by flex turns regular expressions into a form that enables the scanner to scan input extremely quickly, independent of the number of regular expressions that it is trying to match. The flex specification consists of three sections: the definition section, the rules section and the optional user subroutines section. The definition section sets up the execution environment in which flex operates. The rules section contains the declaration of the regular expressions and defines the actions to be taken when a particular pattern is matched. Finally the user subroutines section, which is optional and may contain any valid C code, is used typically to define additional support routines and in gMaGiC contains code for scanner/parser interaction and error reporting. The flex input specification used in gMaGiC is given in appendix C.

---

[1] see `http://www.gnu.ai.mit.edu/` and platform ports at `http://www.delorie.com/djgpp/` and `http://sourceware.cygnus.com/cygwin/`

5.2.2 The Parser

The main task of the parser is to establish the relationship between tokens identified by the scanner and check that they conform to the syntactical structure (grammar) of the process modelling language.

The implementation of the parser in this work is carried out using the GNU project's bison [Levine *et al.*, 1992]. In analogy to flex, bison is a software tool for producing a parser as a C code subroutine from a concise definition of a grammar. Bison works with the flex scanner, calling it whenever a new token is required. Tokens of interest are identified (not all tokens are of interest, whitespace and comments, for example, are usually ignored) and returned to the parser. The grammar definition in bison is declared in a series of production rules. Shift/reduce parsing is used to search for rules that match the tokens as they are identified and defines actions to be taken as each rule is reduced. In this work the tokens and rules serve to define the syntax and structure of the process modelling language and actions are used to construct the physical symbol table as each portion of the input is validated. The bison input specification is structured in the same way as flex and consists of a definition section, a rules section and an optional user subroutines section. The definition section includes the declarations of the tokens used both in the grammar and by the flex scanner and also defines the types of values on the parser stack. The rules section simply consists of the set of production rules that define the grammar and their associated actions. Finally, as with flex, the user subroutines section is used typically to define additional support code, although no extra routines are defined in the parser for gMaGiC. The bison input specification showing the grammar production rules (but not their actions) used in gMaGiC is given in appendix D.

In addition to validation of the syntactical structure of the input the parser is also responsible for construction of the physical symbol table. This is an abstract data structure used to store the physical description in memory for subsequent semantic checking and mathematical model generation. The physical symbol table in gMaGiC is implemented in C as a set of multiply linked lists (see, for example, Kruse *et al.* [1991]) structured around the process modelling language.

5.2.3 Semantic Checking

The scanner and parser together provide the means of validating the syntactical structure of the input physical description against the defined process modelling language. The task of the semantic checking routines is to verify that a syntactically valid input is also semantically correct, providing a meaningful and unambiguous physical description containing all information necessary for generation of the mathematical model. The details of the semantics of the process modelling language have been presented in chapter 3.

The semantic checking subroutines are implemented in C and use the data stored in the physical symbol table constructed by the parser to validate the (syntactically correct) input declaration. Four main categories of semantic errors are identified; ambiguities, errors in geometrical declarations, connection assignment errors and assumption validity errors. Ambiguities result from duplication of user-specified identifiers and cause conflicts that the model generation code is unable to resolve. Typical examples of ambiguities would include the specification of vessels or reservoirs with identical names. Errors in the geometrical declarations involve invalid or infeasible specifications, such as internal diameters larger than external diameters or ports declared at locations that do not lie on the vessel surface. Connection assignment errors result from declarations with redundant or over-specified ports or reservoirs. Semantic errors in any of these three categories are fatal and cause model generation to be aborted. Errors in the declaration of assumptions arise from their misapplication, such as the specification of perfect mixing in a distributed parameter model. Assumption validity errors may be fatal, in which case model generation is aborted, or non-fatal, in which case the offending declaration is ignored and a warning issued.

5.3 The Model Generation Engine

The result of successful translation of the input declaration is a validated internal representation of the physical description stored in memory in the physical symbol table. The second section of gMaGiC, the model generation engine, is responsible for both the formulation of an appropriate mathematical model from this representation and the construction of the mathematical symbol table for intermediate storage of the generated data and subsequent code generation.

The model generation engine forms the core of the gMaGiC implementation. It comprises a (large) set of C subroutines containing the coding of the model building procedure and model generation algorithms and methods for handling the abstract data structures used to store the mathematical information. The latter consist of routines for initialisation and assignment of new data types and functions for combining and structuring them to create the mathematical symbol table.

Full details of the model building procedure and model generation algorithms have been presented in chapter 4. The design and implementation of the mathematical symbol table is discussed below.

5.3.1 The Mathematical Symbol Table

The mathematical symbol table is an abstract data structure used to store an internal representation of the generated mathematical model in memory. In gMaGiC it is implemented in C and consists of three sections on two hierarchical levels, as shown in figure 5.2.

The bottom level comprises sections for storage of individual vessel and reservoir submodels whilst the top level holds the topological information describing their inter-connection which serves to complete the overall mathematical description.

In each section three main types of data structure are employed: linked lists, equation templates and expression trees. Linked lists are used to group data by type. In the vessel and reservoir submodel sections they are employed for storage of both the variable/parameter data, including the name (a text string), type (integer, real or physical type such as density or velocity) and dimension (lumped, 1D or 2D), and also sets of individual equations grouped by category. In the top-level connection section linked lists are used to store pointers to the submodels themselves and again to hold the set of individual (connection) equations. Equation templates and expression trees are used to represent individual mathematical equations. An equation template is an abstract data type defining the pattern of a particular category of equation. Conservation equations, for example, typically comprise accumulation, convective flow, dispersive flow and generation terms whilst constitutive and connection equations are structured simply into left and right hand sides. Each element in an equation template contains a linked list of pointers to expression trees. These are data structures based on triples (see, for example, Aho and Ullman [1977]) which provide a way of storing mathematical equations in a general form. Triples, so called because they contain three parts, consist simply of two arguments and an operator. The arguments may either be empty (null) or contain a model variable/parameter, a numerical constant or another triple structure. The operator is either a unary (if one of the arguments is null) or binary mathematical operator. Mathematical equations of arbitrary complexity are then represented as trees of nested triples, as shown by the example in figure 5.3. In gMaGiC triples are implemented in (ISO) C using the `(void *)` pointer definition.

5.4 The Code Generator

The final stage in the operation of gMaGiC is code generation, in which the internal representation of the mathematical data stored in the mathematical symbol table is formatted into a specific target language and the output written to a file.

The software architecture used in gMaGiC means that implementation of the code generator is relatively straightforward. It consists simply of a set of C subroutines for retrieval of data from the mathematical symbol table and formatting functions for fitting the output to the specific form of the target language. In the current version of gMaGiC the code generator formulates mathematical models in the format of the *gPROMS* modelling language [Barton and Pantelides, 1994; Oh and Pantelides, 1996] for direct use in the *gPROMS* process modelling environment. The nomenclature used for the output from gMaGiC is given in appendix A. Adoption of this architecture also

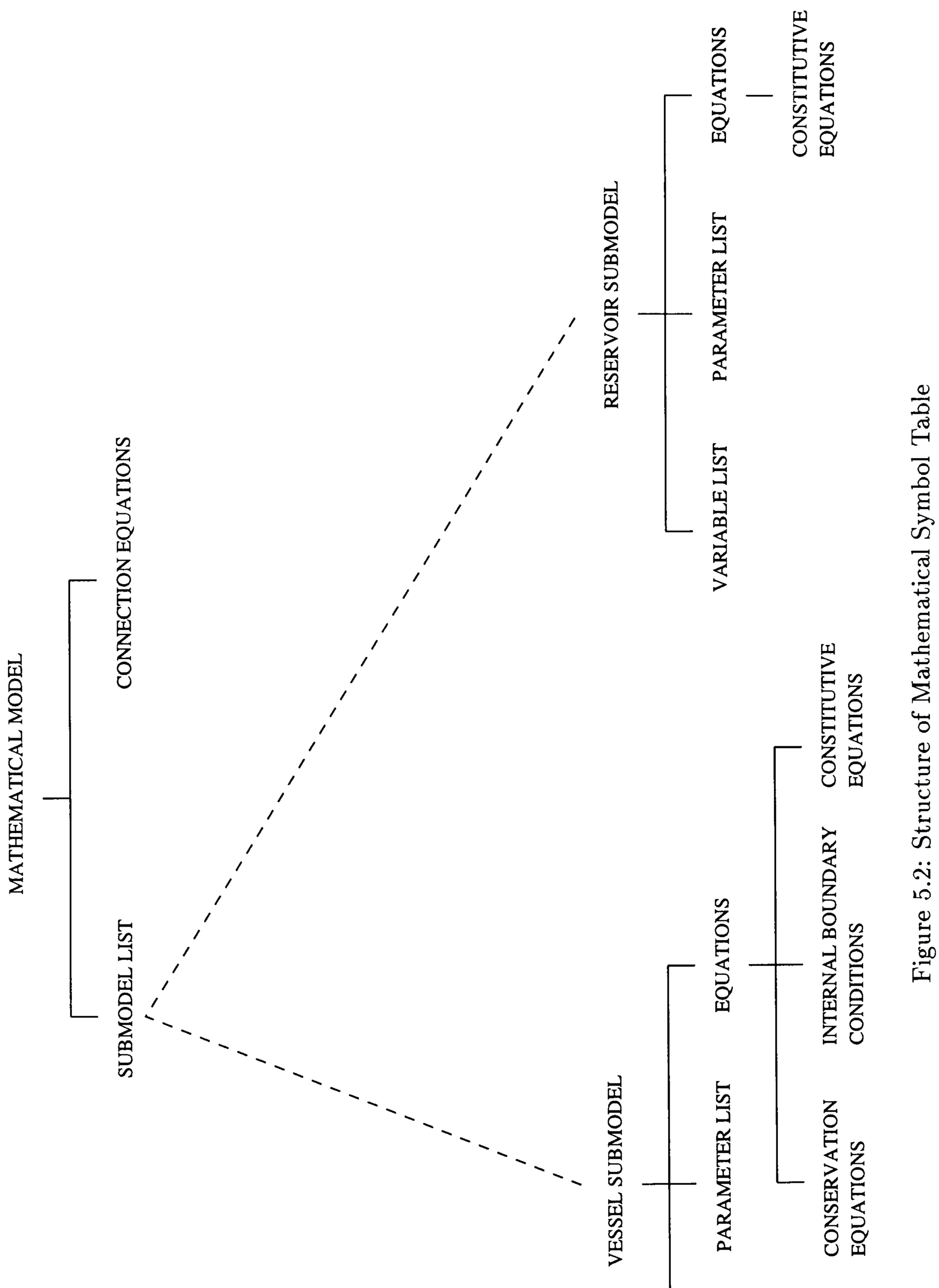


Figure 5.2: Structure of Mathematical Symbol Table

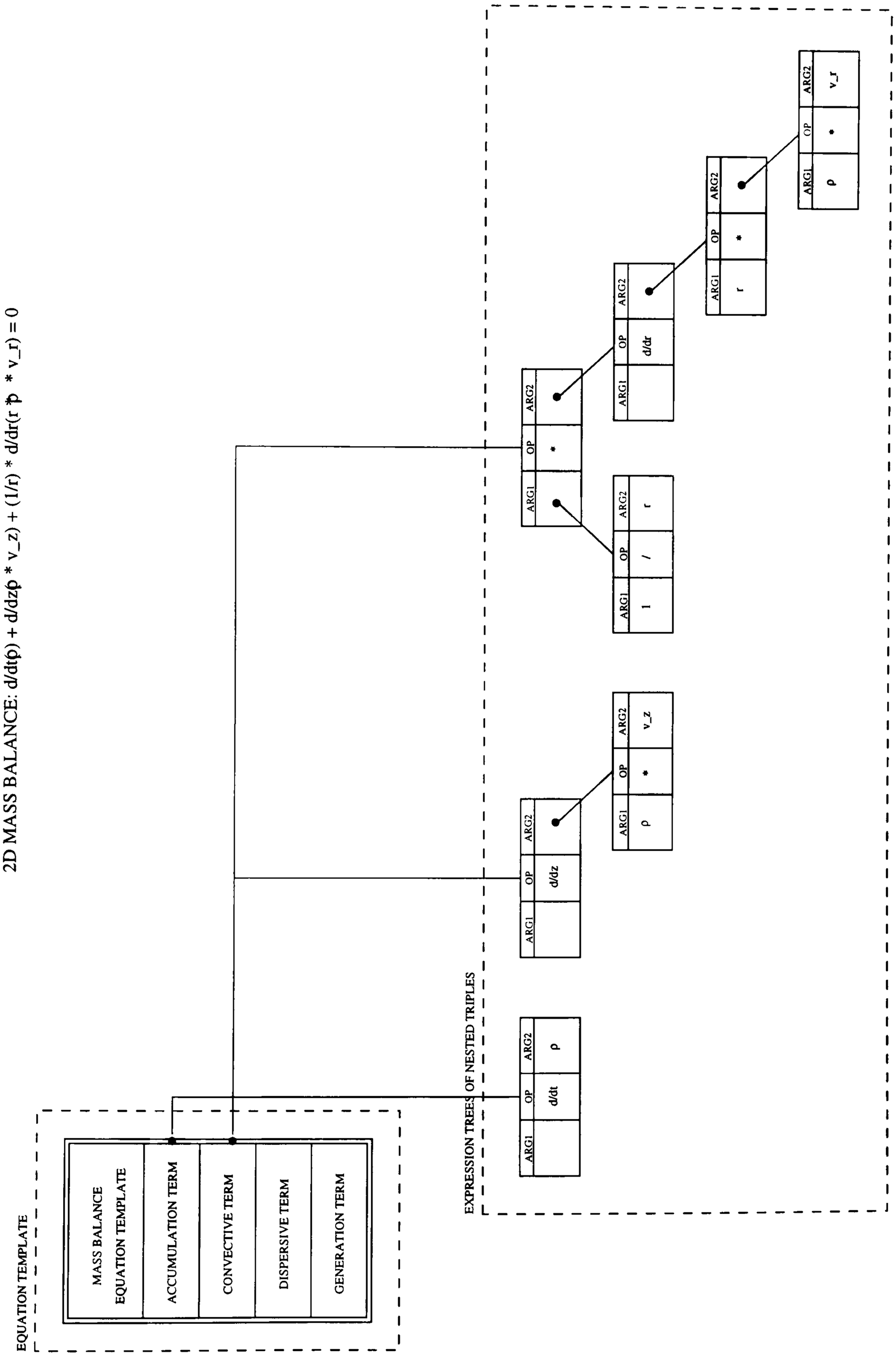


Figure 5.3: Example of Equation Templates and Expression Trees

has the consequence that the generation of models in different formats (target languages) requires only additional implementation in the code generator and therefore provides the potential for multi-functionality, permitting the generation of different forms of the same mathematical model as required.

5.5 Program Operation

Figure 5.4 illustrates the operation of gMaGiC. It contains three windows and two arrows indicating the flow of operation. The window on the left hand side shows the command line. The program is executed with the command

```
>generate <input file> <output file> <CR>
```

The first argument is the name of the executable. The second argument specifies the text file (shown in the top window on the right hand side) containing the process description in terms of the modelling language defined in chapter 3. This file is passed (internally) to the translator where it is checked for syntactical and semantic correctness. After successful translation the model generation engine is invoked and the mathematical model constructed and stored internally. The code generator then writes the output to the text file specified by the third argument in the command line (shown in the bottom window on the right hand side). Finally, internal data structures used during execution are removed from memory and the program terminates with a message indicating the name of the file where the model is stored. Examples of models generated by gMaGiC are given in appendix E.

5.6 Summary

In this chapter details of the implementation of gMaGiC (*g*eneral *M*odel *G*eneration *C*ode), a computer program for the automatic generation of process models, have been presented. The implementation is based on the formal process representation and model building procedure defined in chapters 3 and 4 respectively and consists of three main sections: the translator, the model generation engine and the code generator.

The translator takes the input declaration, defined in terms of the process modelling language defined in chapter 3, and checks its syntax and semantics. If translation is successful an internal representation of the physical description is stored internally in the physical symbol table. The model generation engine then constructs an appropriate mathematical model from this representation by applying the model building procedure developed in chapter 4 and stores the generated data in the mathematical symbol table. Finally, the code generator transforms the model stored in the mathematical symbol table into the format of a specific target language. In the current version of gMaGiC the output

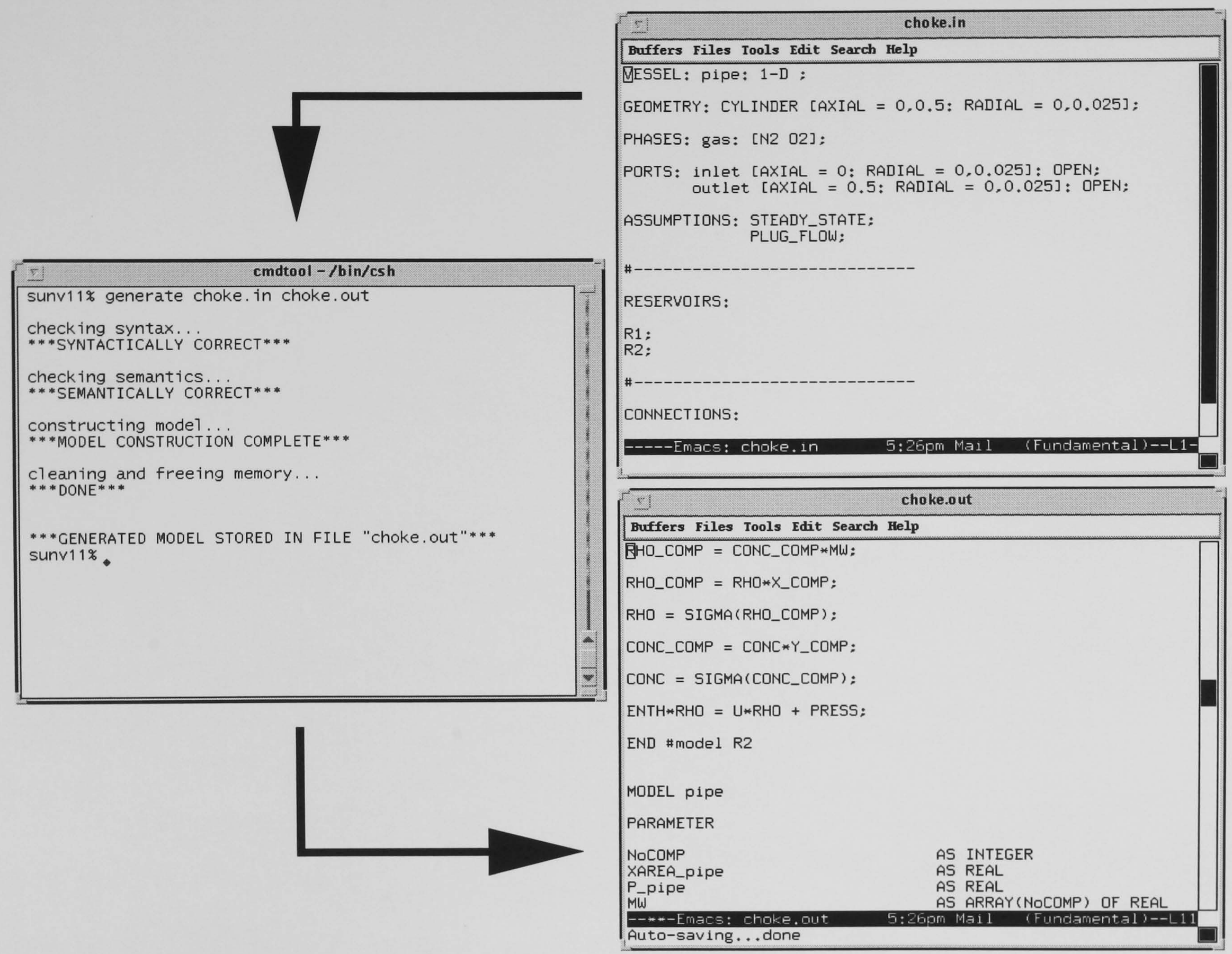


Figure 5.4: Example of gMaGiC Operation

is produced in the format of the *gPROMS* modelling language [Barton and Pantelides, 1994; Oh and Pantelides, 1996].

In the next chapter the success of the modelling approach developed in this thesis is demonstrated by the application of gMaGiC to several nontrivial case studies.

Chapter 6

Modelling Examples

In the previous chapters a process representation and systematic model building procedure were defined permitting the generation of mathematical models of process systems from purely physical descriptions. Details of the implementation of this approach in the computer program gMaGiC (*g*eneral *M*odel *G*eneration *C*ode) were also given.

In this chapter several case studies are presented to demonstrate and validate the modelling approach developed in this thesis and its software implementation in gMaGiC. They are chosen to illustrate the success of the approach in modelling systems exhibiting a variety of different physico-chemical behaviour including compressible and incompressible fluids, varying thermal characteristics (isothermal, adiabatic and finite heat transfer), laminar and turbulent flow regimes, single and multicomponent mixtures (both reacting and inert) and operation under steady state and dynamic conditions.

For all case studies a uniform presentation is adopted comprising three sections: an introduction, a discussion of the process representation and a description of the mathematical modelling and model validation. The introduction gives the background to the case study and the source of the data used together with any other relevant information such as analytical or empirical results or expected physico-chemical behaviour. The process representation section gives details of the process abstraction and physical description of the case study in terms of the conceptual framework and process modelling language defined in chapter 3, the latter forming the input specification to gMaGiC. Finally, the modelling and validation section describes the model(s) generated by gMaGiC and demonstrates that the results of their (numerical) solution are consistent with published data, thereby proving the validity of both the model generation approach and its software implementation.

Wherever possible numerical solution of the generated models is achieved using the *gPROMS* modelling environment [Barton and Pantelides, 1994; Oh and Pantelides, 1996] (hence construction of the generated models are constructed in the format of the *gPROMS* modelling language, see chapter 5). However, in some cases, limitations of the numerical methods and/or idiosyncrasies of the *gPROMS* environment mean that direct solution of the model equations as generated is not possible. The individual treatment of these cases is discussed as they arise in the appropriate sections (below).

6.1 Choking in Compressible Flow with Friction

For the first example the steady, adiabatic flow of a compressible fluid through a duct of constant cross-section is considered. In such flows the frictional effects bring about a change in the fluid velocity and an increase in the fluid entropy along the length of the duct. The maximum possible entropy occurs at the point where the fluid velocity is equal to the sonic velocity and hence where the Mach number (6.1) is equal to unity.

$$\mathrm{Ma} = \frac{v}{u} = \frac{v}{\sqrt{\gamma R T M_w}} \tag{6.1}$$

Therefore, for given inlet conditions, there is a maximum length of duct for which the Mach number of unity is reached at the outlet. The flow is then said to be choked. If the length of the duct is increased further the flow must be adjusted so that the Mach number at the outlet remains equal to unity. For subsonic flows this adjustment is achieved by an automatic reduction in the mass flow rate whilst for supersonic flows the adjustment is usually accompanied by the formation of shock waves [Daneshyar, 1976].

The particular case under consideration here is based on an example by Daneshyar [1976]. The problem is to determine both the maximum air flow rate through a duct of length 0.5 m and diameter 0.05 m, assuming a constant friction coefficient of 0.005, and the range of downstream pressures for which this flow rate will be achieved. The inlet conditions at the duct entrance are a stagnation temperature T_0 (6.2) of 340 K and stagnation pressure p_0 (6.3) of 650 kPa.

$$T_0 = T\left(1 + \frac{\gamma - 1}{2}\mathrm{Ma}^2\right) \tag{6.2}$$

$$\left(\frac{p_0}{p}\right) = \left(\frac{T_0}{T}\right)^{\frac{\gamma}{\gamma-1}} \tag{6.3}$$

For choked flows of perfect gases in systems with constant friction coefficients analytical solutions for the maximum duct length can be obtained from (6.4) (see Daneshyar [1976] for derivation).

$$4f\frac{L_{max}}{D} = \frac{1}{\gamma}\left(\frac{1}{\mathrm{Ma}_{in}^2} - 1\right) + \frac{\gamma+1}{2\gamma}\ln\left(\frac{\frac{\gamma+1}{2}\mathrm{Ma}_{in}^2}{1 + \frac{\gamma-1}{2}\mathrm{Ma}_{in}^2}\right) \tag{6.4}$$

However, for the purpose of illustrating and validating the modelling approach developed in this thesis it is more convenient to formulate the problem in a different manner. Using the mass flow rate from the analytical solution together with the stagnation temperature and pressure at the entrance to the duct, it is possible to calculate the inlet flow velocity and the inlet thermodynamic temperature and pressure. These specify fully the operating conditions at the duct entrance. A correct model will then predict an increase in velocity

from its inlet value to the sonic velocity (and hence a Mach number of unity) at the exit of the pipe. For higher flow rates the physical constraints mean that no solution is possible.

6.1.1 Process Representation

The process representation for this example is straightforward and is shown below (figure 6.1).

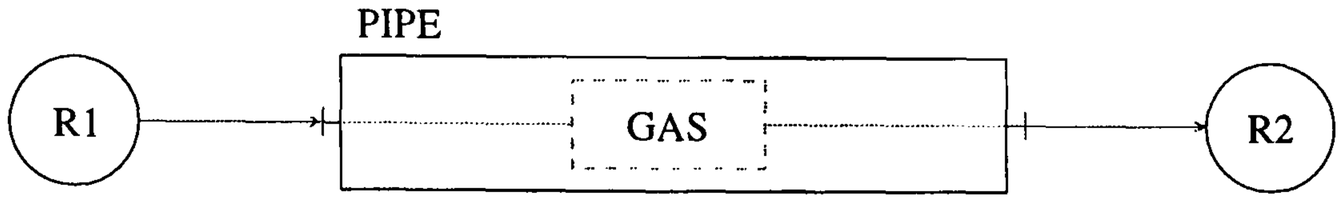


Figure 6.1: Process Representation for Choking Compressible Flow in a Pipe

The system consists of a single vessel (`pipe`) connected to two reservoirs (`R1` and `R2`) representing the upstream and downstream system boundaries. The vessel contains a single phase, multicomponent (but nonreacting) gas (air). Since the flow is adiabatic there is no heat transfer to or from the vessel and hence no other connections are needed.

The formal description of the system is made in terms of the process modelling language defined in chapter 3. The vessel is modelled simply as a horizontally oriented cylinder containing a single phase binary mixture (`gas`). The vessel is connected to the two reservoirs representing the upstream and downstream limits via two open ports (`inlet` and `outlet`). Following Daneshyar, a one dimensional, steady state model is generated and plug flow is assumed. The full language declaration is given below (figure 6.2).

6.1.2 Modelling and Validation

The mathematical model generated for this example from the process representation described above is given in appendix E.1[1]. It comprises four (sub-)models on two hierarchical levels. The bottom level contains two reservoir models (`R1` and `R2`) which define the states of the upstream and downstream system boundaries and a one dimensional vessel model (`pipe`) describing the control volume enclosed by the duct. The latter consists of steady state multicomponent mass, total energy and linear momentum balances together with constitutive equations in an appropriate form and an indication of the functional form of the equations of state. Since plug flow is assumed, the second order dispersive terms in the total energy and linear momentum equations are neglected. The top level contains a single model (`FLOWSHEET`) which describes the connectivity between the vessel and reservoirs. In this case, with plug flow in the vessel, it consists of simple equality relations

[1] All models given appendix E are raw output from gMaGiC in the format of the *gPROMS* modelling language [Barton and Pantelides, 1994; Oh and Pantelides, 1996] (see chapter 5). The complete *gPROMS* files are given in appendix F.

```
VESSEL: pipe: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0,0.5: RADIAL = 0,0.025];

PHASES: gas: [N2 O2];

PORTS: inlet [AXIAL = 0: RADIAL = 0,0.025]: OPEN;
       outlet [AXIAL = 0.5:RADIAL = 0,0.025]: OPEN;

ASSUMPTIONS: STEADY_STATE;
             PLUG_FLOW;
#-----------------------------
RESERVOIRS:

R1;
R2;
#-----------------------------
CONNECTIONS:

R1->pipe.inlet;
pipe.outlet->R2;
```

Figure 6.2: Language Declaration for Choking Flow

matching the reservoir variables with those at either end of the duct. The numerical specifications for this example are summarised in table 6.1.

Model	Quantity	Values	Units
R1	p	466×10^3	Pa
	T	309	K
	w	2.552	kg/s
	y_{N_2}	0.79	-
	y_{O_2}	0.21	-
pipe	D	0.05	m
	f	0.005	-
	L	0.5	m

Table 6.1: Numerical Specifications for Choking Flow

Validation of the mathematical model is provided in two ways. Firstly, for the flow to be choked, the flow velocity at the exit of the duct must be equal to the sonic velocity and, consequently, the Mach number equal to unity. Secondly, the existence of an analytical solution provides a convenient means of model verification because it can be compared to

the numerical solution of a model derived under the same assumptions.

For this example the analytical solution gives a maximum air flow rate of approximately 2.560 kg/s for downstream pressures of 315 kPa or less. Solving in *gPROMS* using the perfect gas law and a discretisation scheme of simple first order backward finite differences (BFDs) the numerical solution closest to the analytical solution is an air flow rate of 2.552 kg/s and downstream pressure of approximately 320 kPa (figure 6.3), giving an exit Mach number of 0.975 (figure 6.4).

These results (summarised in table 6.2) are in good agreement with the analytical solution in light of the fact that a simple numerical method was employed for discretisation of the model equations. In fact, to capture sonic and supersonic flows exactly usually requires sophisticated and often case-specific numerical codes (see, for example, Fletcher [1988] or Versteeg and Malalasekera [1995]).

	Analytical Solution	Numerical Solution (BFDs)
Mass Flow rate - kg/s	2.560	2.552
Mach Number -	1.000	0.975
Downstream Pressure - kPa	315	322

Table 6.2: Results from Choking Flow Simulation

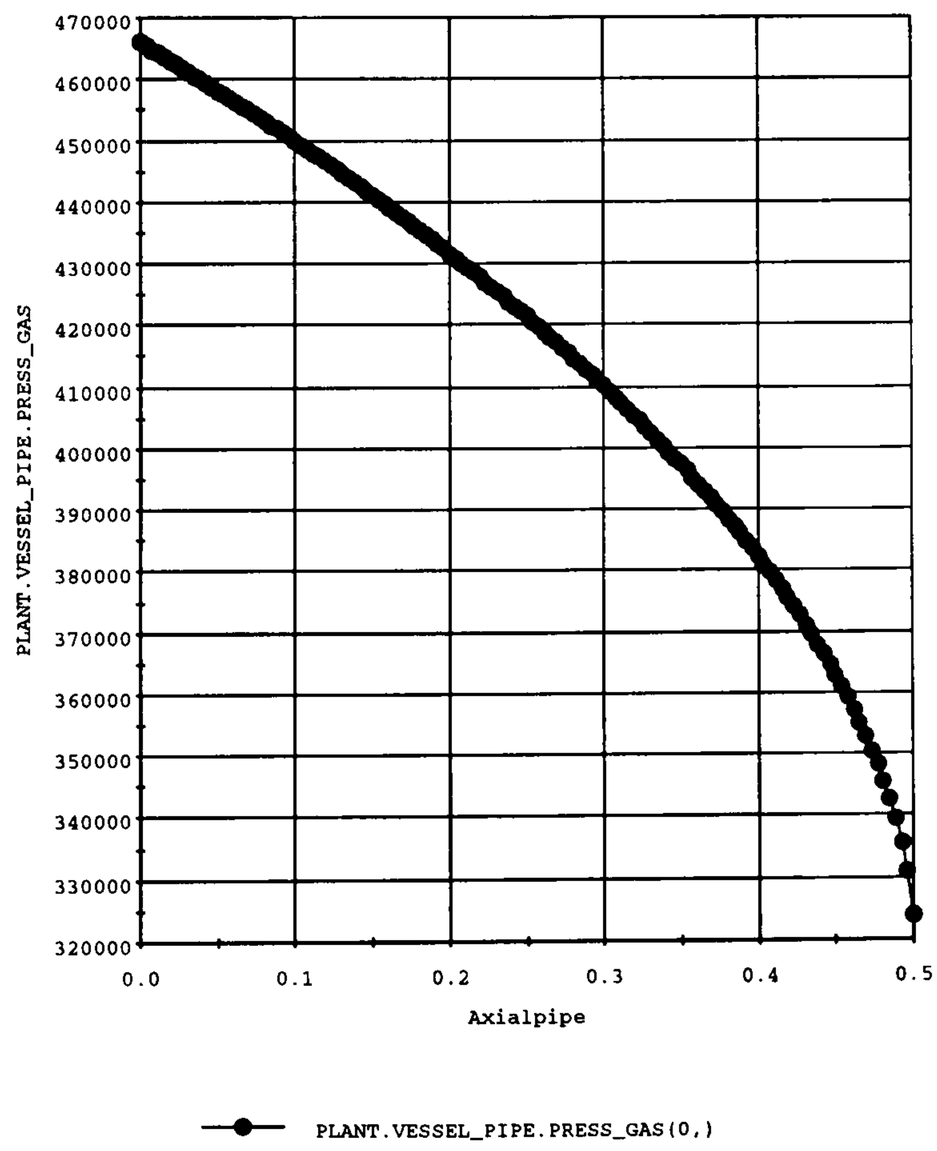


Figure 6.3: Pressure [Pa] versus Axial Position [m]

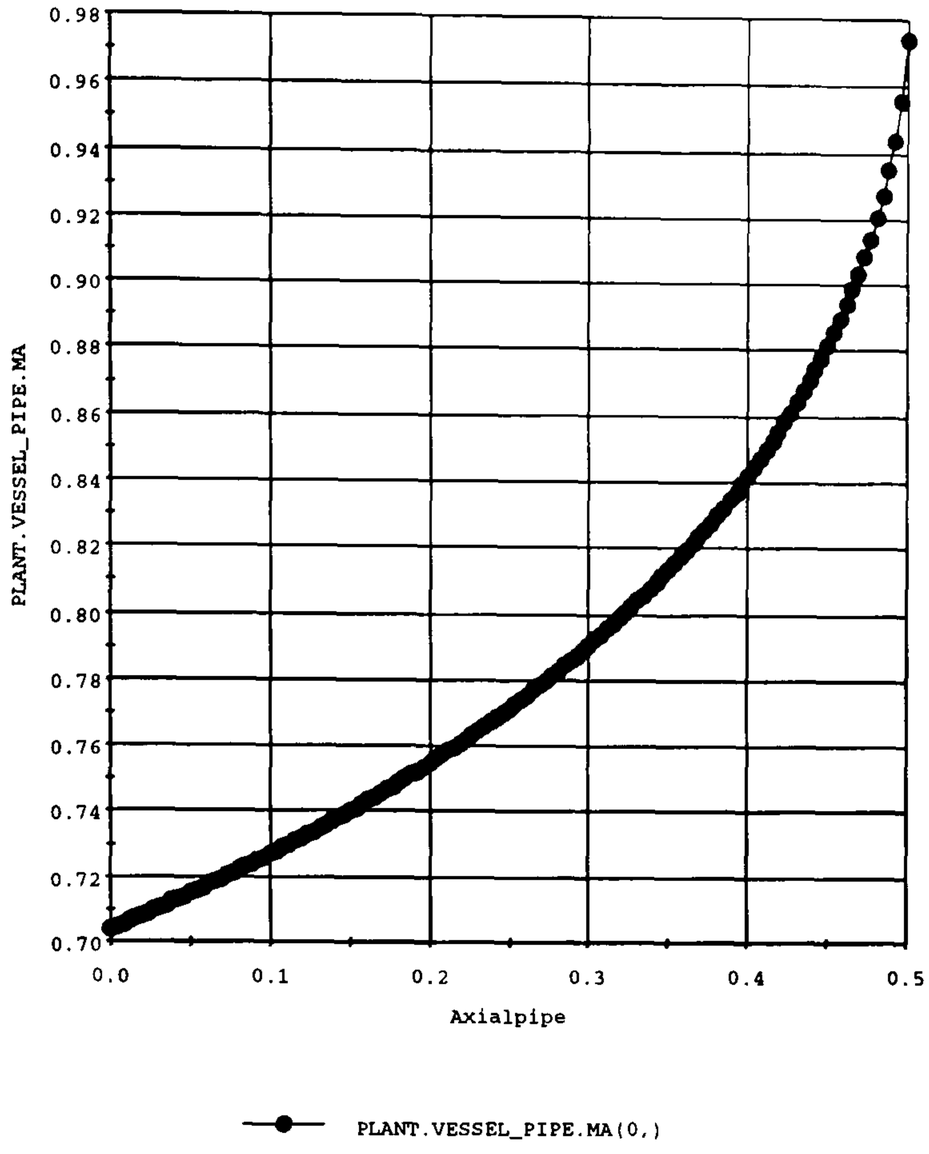


Figure 6.4: Mach Number [-] versus Axial Position [m]

6.2 Vinyl Acetate Reactor

In this example the modelling of a tubular reactor for the production of vinyl acetate monomer is considered, based on a study by Luyben and Tyreus [1998]. Three raw materials, ethylene (C_2H_4), oxygen (O_2) and acetic acid (HAc), are converted into the vinyl acetate (VAc) product, with water (H_2O) and carbon dioxide (CO_2) produced as by-products. The reaction scheme is as follows

$$C_2H_4 + HAc + \tfrac{1}{2}O_2 \rightarrow VAc + H_2O \qquad (A)$$

$$C_2H_4 + 3O_2 \rightarrow 2CO_2 + 2H_2O \qquad (B)$$

An inert component, ethane (C_2H_6), also enters with the fresh ethylene feed stream. The reactions are both irreversible with reaction rates given by the following (empirical) expressions

$$r_A = k_A \frac{p_{O_2} p_{C_2H_4} p_{HAc} (1 + 1.7 p_{H_2O})}{(1 + 0.583 p_{O_2}(1 + 1.7 p_{H_2O}))(1 + 6.8 p_{HAc})} \qquad (6.5)$$

$$r_B = k_B \frac{p_{O_2}(1 + 0.68 p_{H_2O})}{1 + 0.76 p_{O_2}(1 + 0.68 p_{H_2O})} \qquad (6.6)$$

where the rate constants have an Arrhenius-type dependence on temperature (6.7).

$$k = A \exp\left(-\frac{E_A}{RT}\right) \tag{6.7}$$

The exothermic reactions take place in the tubes of a fixed bed tubular reactor packed with a precious metal catalyst on a silica support. The reactor consists of 622 tubes each of diameter 3.7 cm and length 10 m. Cooling is provided by raising saturated steam on the shell-side of the tubes. For this example, following Luyben and Tyreus, the modelling of only a single reactor tube is considered and the shell-side cooling medium is taken to be at a constant temperature. Component flow rates for the entire process stream may then be obtained by multiplying the flow rates for a single tube by the total number of tubes in the reactor.

6.2.1 Process Representation

The process representation for the system is shown in figure 6.5. It consists of a single vessel (`reactorTube`) connected to two reservoirs representing the upstream and downstream system boundaries (`R1` and `R2` respectively). A third reservoir (`R3`) is used to model the shell-side cooling medium (saturated steam). If the dynamics of the shell-side were important it could also be defined as a vessel (see section 6.3 for such an example).

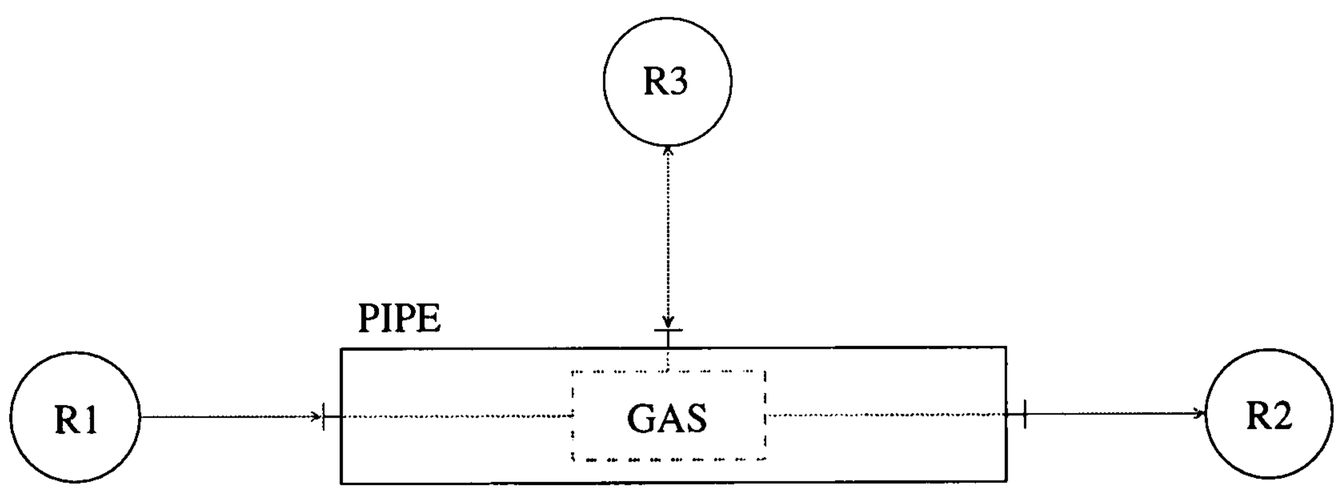


Figure 6.5: Process Representation for Vinyl Acetate Reactor

The language declaration for this example is given in figure 6.6. The reactor tube is modelled simply as a horizontally oriented cylinder containing a single phase, multicomponent fluid (**gas**). Connections to the upstream and downstream reservoirs (`R1` and `R2`) are made via two open ports (`inlet` and `outlet`) and that to the shell-side reservoir via a single closed port (`wall`). Following Luyben and Tyreus, a one dimensional model is generated with the assumptions of plug flow and quasi-steady state operation (time-independent momentum equation).

6.2.2 Modelling and Validation

The mathematical model generated for this example is given in appendix E.2. It comprises five (sub-)models on two hierarchical levels. The bottom level contains three reservoir models (`R1`, `R2` and `R3`) which define the states of the system boundaries (for `R3` this

```
VESSEL: reactorTube: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0,10: RADIAL = 0,0.0185];

PHASES: gas: [O2 CO2 C2H4 C2H6 VAc H20 HAc];

PORTS: inlet [AXIAL = 0: RADIAL = 0,0.0185]: OPEN;
       outlet [AXIAL = 10: RADIAL = 0,0.0185]: OPEN;
       wall [AXIAL = 0,10: RADIAL = 0.0185]: CLOSED;

ASSUMPTIONS: QUASI_STEADY_STATE;
             PLUG_FLOW;
#-----------------------------
RESERVOIRS:

R1;
R2;
R3;
#-----------------------------
CONNECTIONS:

R1->reactorTube.inlet;
reactorTube.outlet->R2;
reactorTube.wall->R3;
```

Figure 6.6: Language Declaration for Vinyl Acetate Reactor

consists simply of the temperature) and a one dimensional vessel model (**reactorTube**) describing the reactor control volume. The latter consists of dynamic multicomponent mass and total energy balances and a time-independent (quasi-steady state) linear momentum equation, together with a set of constitutive relations and an indication of the functional form of the equations of state. Since no reaction library is included in the current version of gMaGiC (see chapter 3) the reaction kinetics are entered manually prior to solution. The top level contains a single model (**FLOWSHEET**) comprising equations describing the three reservoir-vessel connections. These take the form of a set of equality relations matching variables across the open connections and also a film heat transfer model describing the energy exchange between the reactor and the shell-side cooling medium. The numerical specifications for this example are summarised in table 6.3.

The results from numerical solution of the model in *gPROMS* for this example are given in figures 6.7 and 6.8. These show steady state axial temperature and pressure profiles (respectively) obtained using the Soave-Redlich-Kwong equation of state (see, for example, Reid *et al.* [1987]) and a discretisation scheme of first order backward finite differences. They are in good agreement both with results published by Luyben and

Model	Quantity	Values	Units
R1	p	128	psia
	T	148.5	°C
	$\mathcal{W}$	19250	mol/min
	y_{O_2}	0.075	-
	y_{CO_2}	0.007	-
	$y_{C_2H_4}$	0.583	-
	$y_{C_2H_6}$	0.216	-
	y_{VAc}	0.000	-
	y_{HAc}	0.110	-
	y_{H_2O}	0.009	-
reactorTube	A_A	0.1036	†
	A_B	1.9365×10^5	†
	D	3.7	cm
	E_{A_A}/R	3674	K
	E_{A_B}/R	10116	K
	h	150	kcal/h °C m^2
	L	10	m
	ρ_{cat}	0.385	kg/l
R3	T	133	°C

† gives reaction rates (r_A and r_B) in mol/min g-cat

Table 6.3: Numerical Specifications for Vinyl Acetate Reactor

Tyreus [1998] and those obtained later by Cobden [2000] in an analysis of the same system. Discrepancies between the results obtained here and those of Luyben and Tyreus are most likely due to the different numerical schemes employed. Indeed, Cobden's axial temperature profile is in almost exact agreement with figure 6.7 since the same type and order of finite difference approximation is used for discretisation of the governing partial differential equations.

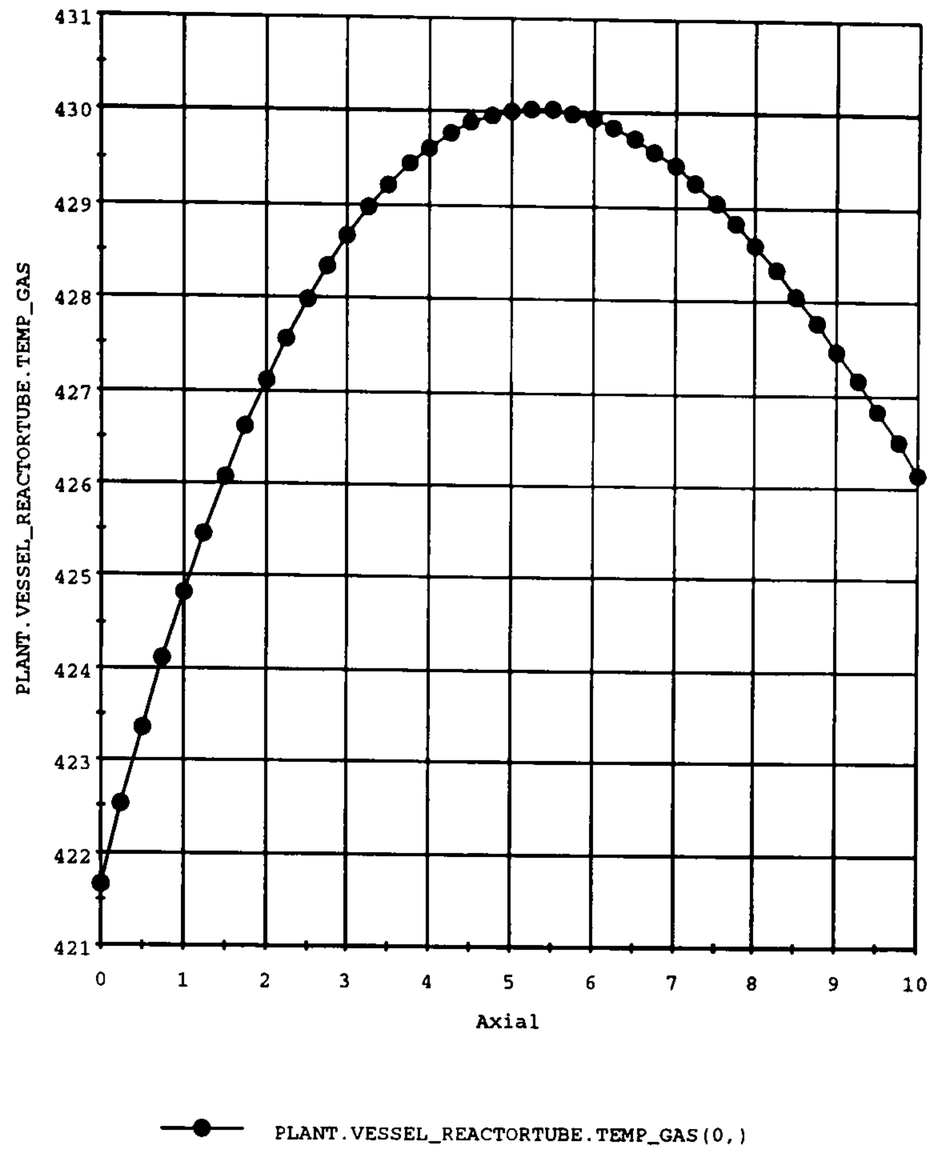


Figure 6.7: Temperature [K] versus Axial Position [m]

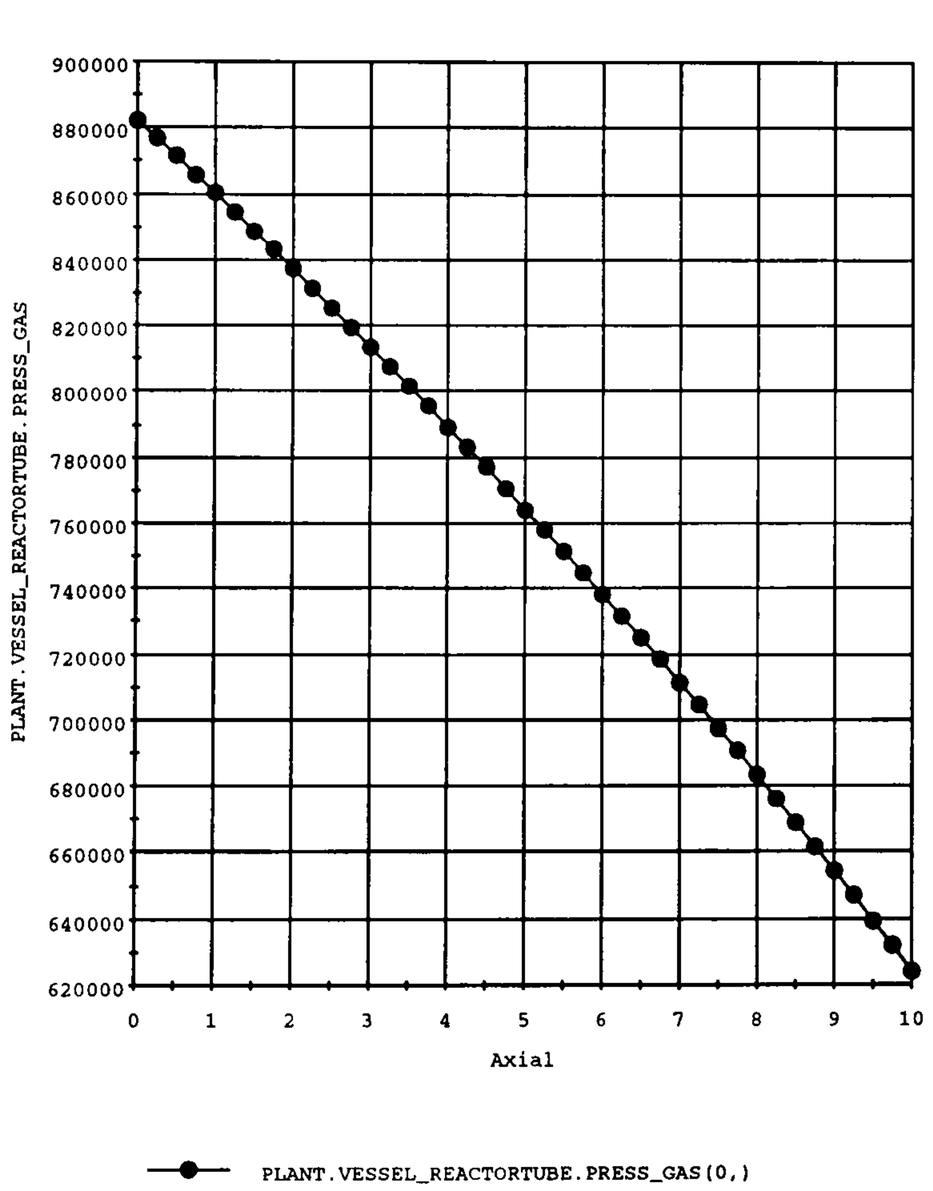


Figure 6.8: Pressure [Pa] versus Axial Position [m]

6.3 Double Pipe Heat Exchanger

In this example several mathematical models of a double (or concentric) pipe heat exchanger are considered based on a sample heat exchanger design calculation taken from Kern [1950]. The main aim of these examples is to demonstrate connectivity between mathematical models of different dimensionality and also the generation of different forms of model by the application of modelling assumptions.

The examples are based on a double pipe heat exchanger designed to heat 9820 lb/hr of cold benzene from 80 to 120 °F (300 to 322 K) using hot toluene which is cooled from 160 to 100 °F (344 to 311 K). The dimensions of the heat exchanger are summarised below.

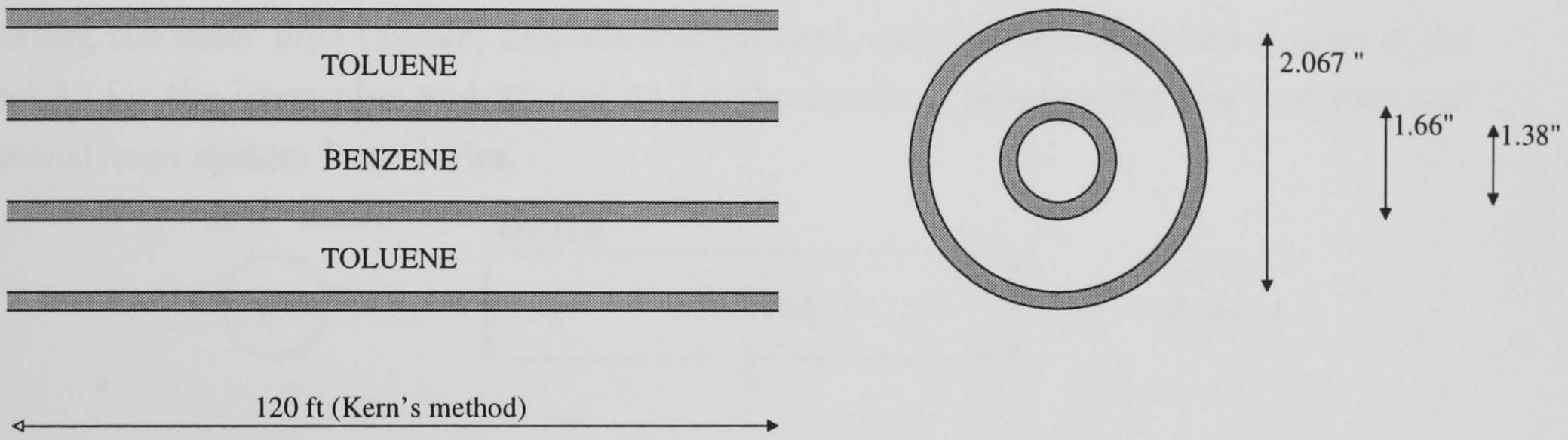


Figure 6.9: Heat Exchanger Dimensions

The heat transfer coefficients and Fanning friction factors in the inner pipe and annulus are given by standard dimensionless relations (6.8 and 6.9 respectively).

$$\mathrm{Nu} = 0.023\mathrm{Re}^{0.8}\mathrm{Pr}^{0.33} \tag{6.8}$$

$$f = 0.0035 + \frac{0.264}{\mathrm{Re}^{0.42}} \tag{6.9}$$

For a countercurrent flow arrangement with benzene in the inner pipe and toluene in the annulus Kern's method gives a required exchanger length of 120 ft (approximately 36 m).

For the purposes considered here Kern's example is modified and treated as a base case from which four alternative models are generated. In each alternative the inlet temperatures, flow rates and exchanger length are taken from Kern's solution but the geometrical description is simplified such that the wall thickness and its corresponding thermal resistance are ignored. All physical properties are assumed to be constant and are taken from Kern [1950]. Table 6.4 summarises the models considered.

	Inner Pipe	Annulus	Flow Arrangement
Case 1	macroscopic	macroscopic	
Case 2	1D	macroscopic	
Case 3	1D	1D	cocurrent
Case 4	1D	1D	countercurrent

Table 6.4: Model Alternatives for Double Pipe Heat Exchanger Example

6.3.1 Process Representation

The basic process representation is shown in figure 6.5. It consists of two vessels representing the inner pipe (`inner`) and annulus (`outer`), each connected to two reservoirs (`R1` and `R2` for the inner pipe and `R3` and `R4` for the annulus) representing the upstream and downstream system boundaries.

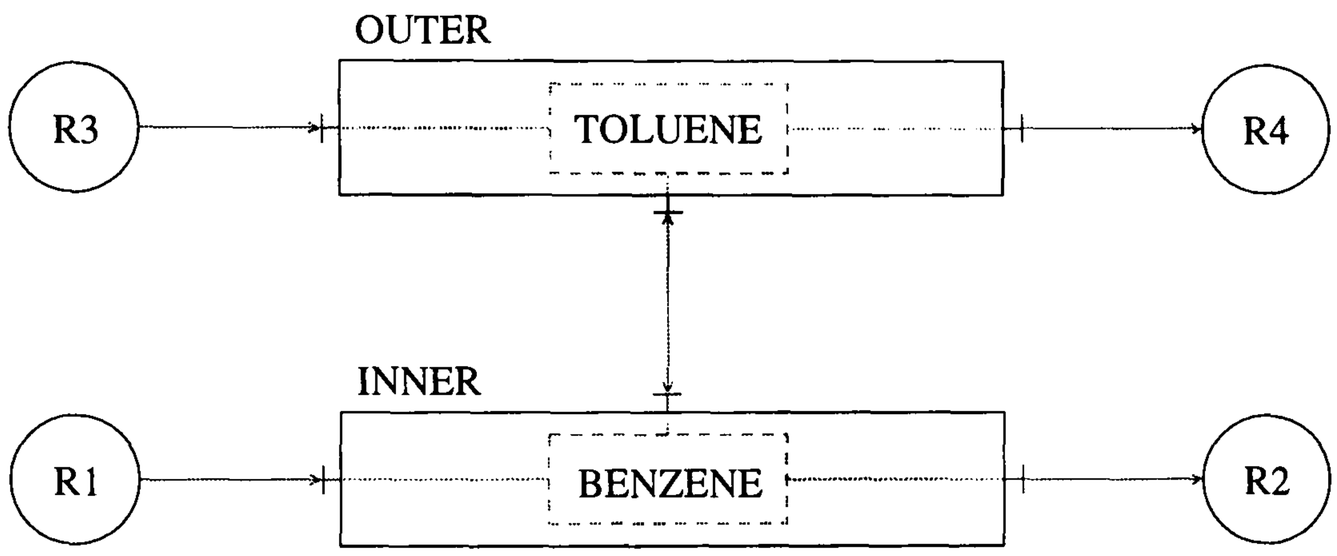


Figure 6.10: Process Representation for Double Pipe Heat Exchanger (cocurrent flow)

In all four cases both vessels `inner` and `outer` are modelled as horizontally oriented cylinders containing a single, incompressible liquid phase (`benzene` and `toluene` respectively). The connections to the source (`R1` and `R3`) and product (`R2` and `R4`) reservoirs are made via open ports (`inlet` and `outlet`) and the inter-vessel connection via closed ports (both named `wall`).

Modifications to the basic process representation and specific modelling details for each of the four cases are given below.

6.3.1.1 Case 1

For the first case the process representation is unchanged from the base case. In both the inner pipe and annulus macroscopic models are generated and perfect mixing is assumed. The full language declaration is given in figure 6.11.

6.3.1.2 Case 2

In case 2 the process representation is also unchanged from the base case. The vessel representing the annulus is modelled as in case 1 but, in this case, a one dimensional distributed parameter model is generated for the inner pipe assuming plug flow and quasi-steady state operation (time-independent momentum equation). The language declaration for this case is given in figure 6.12.

6.3.1.3 Case 3

For case 3 the process representation is again the same as the base case. However, in this case one dimensional models are generated for both vessels assuming plug flow and quasi-steady state operation. Both vessels are oriented horizontally and in the same positive direction (`LEFT`), indicating a cocurrent flow arrangement. The language declaration for this case is given in figure 6.13.

6.3.1.4 Case 4

For the final case the basic representation is modified to describe countercurrent flow by reversing the position of reservoirs `R3` and `R4`. The effect is to reverse the bulk flow direction in the annulus relative to that in the inner pipe. The modification is described in the modelling language in terms of the declaration of the vessel orientations (`LEFT` for the inner pipe and `RIGHT` for the annulus). In all other respects the modelling assumptions are the same as in case 3. The language declaration for this case is given in figure 6.14.

```
VESSEL: inner: MACROSCOPIC;

PHASES: cold: [Benzene];

PORTS: inlet: OPEN;
       outlet: OPEN;
       wall: CLOSED;

ASSUMPTIONS: INCOMPRESSIBLE;
             WELL_MIXED;
             NO_SHAFT_WORK;
#-----------------------------------------
VESSEL: outer: MACROSCOPIC;

PHASES: hot: [Toluene];

PORTS: inlet: OPEN;
       outlet: OPEN;
       wall: CLOSED;

ASSUMPTIONS: INCOMPRESSIBLE;
             WELL_MIXED;
             NO_SHAFT_WORK;
#-----------------------------------------
RESERVOIRS:
R1;
R2;
R3;
R4;
#-----------------------------------------
CONNECTIONS:

R1->inner.inlet;
inner.outlet->R2;
R3->outer.inlet;
outer.outlet->R4;
inner.wall->outer.wall;
```

Figure 6.11: Language Declaration for Double Pipe Heat Exchanger - Case 1

```
VESSEL: inner: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0, 36: RADIAL = 0, 0.02105];

PHASES: cold: [Benzene];

PORTS: inlet  [AXIAL = 0: RADIAL = 0, 0.02105]: OPEN;
       outlet [AXIAL = 36: RADIAL = 0, 0.02105]: OPEN;
       wall   [AXIAL = 0, 36: RADIAL = 0.02105]: CLOSED;

ASSUMPTIONS: QUASI_STEADY_STATE;
             INCOMPRESSIBLE;
             PLUG_FLOW;
#-----------------------------------------
VESSEL: outer: MACROSCOPIC;

PHASES: hot: [Toluene];

PORTS: inlet: OPEN;
       outlet: OPEN;
       wall: CLOSED;

ASSUMPTIONS: INCOMPRESSIBLE;
             WELL_MIXED;
             NO_SHAFT_WORK;
#-----------------------------------------
RESERVOIRS:

R1;
R2;
R3;
R4;
#-----------------------------------------
CONNECTIONS:

R1->inner.inlet;
inner.outlet->R2;
R3->outer.inlet;
outer.outlet->R4;
inner.wall->outer.wall;
```

Figure 6.12: Language Declaration for Double Pipe Heat Exchanger - Case 2

```
VESSEL: inner: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0, 36: RADIAL = 0, 0.02105]: LEFT;

PHASES: cold: [Benzene];

PORTS: inlet  [AXIAL = 0: RADIAL = 0, 0.02105]: OPEN;
       outlet [AXIAL = 36: RADIAL = 0, 0.02105]: OPEN;
       wall   [AXIAL = 0, 36: RADIAL = 0.02105]: CLOSED;

ASSUMPTIONS: QUASI_STEADY_STATE;
             INCOMPRESSIBLE;
             PLUG_FLOW;
#-----------------------------------------
VESSEL: outer: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0, 36: RADIAL = 0.02105, 0.02629]: LEFT;

PHASES: hot: [Toluene];

PORTS: inlet  [AXIAL = 0: RADIAL = 0.02105, 0.02629]: OPEN;
       outlet [AXIAL = 36: RADIAL = 0.02105, 0.02629]: OPEN;
       wall   [AXIAL = 0, 36: RADIAL = 0.02105]: CLOSED;

ASSUMPTIONS: QUASI_STEADY_STATE;
             INCOMPRESSIBLE;
             PLUG_FLOW;
#-----------------------------------------
RESERVOIRS:

R1;
R2;
R3;
R4;
#-----------------------------------------
CONNECTIONS:

R1->inner.inlet;
inner.outlet->R2;
R3->outer.inlet;
outer.outlet->R4;
inner.wall->outer.wall;
```

Figure 6.13: Language Declaration for Double Pipe Heat Exchanger - Case 3

```
VESSEL: inner: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0, 36: RADIAL = 0, 0.02105]: LEFT;

PHASES: cold: [Benzene];

PORTS: inlet  [AXIAL = 0: RADIAL = 0, 0.02105]: OPEN;
       outlet [AXIAL = 36: RADIAL = 0, 0.02105]: OPEN;
       wall   [AXIAL = 0, 36: RADIAL = 0.02105]: CLOSED;

ASSUMPTIONS: QUASI_STEADY_STATE;
             INCOMPRESSIBLE;
             PLUG_FLOW;
#---------------------------------------
VESSEL: outer: 1-D;

GEOMETRY: CYLINDER [AXIAL = 0, 36: RADIAL = 0.02105, 0.02629]: RIGHT;

PHASES: hot: [Toluene];

PORTS: inlet  [AXIAL = 0: RADIAL = 0.02105, 0.02629]: OPEN;
       outlet [AXIAL = 36: RADIAL = 0.02105, 0.02629]: OPEN;
       wall   [AXIAL = 0, 36: RADIAL = 0.02105]: CLOSED;

ASSUMPTIONS: QUASI_STEADY_STATE;
             INCOMPRESSIBLE;
             PLUG_FLOW;
#---------------------------------------
RESERVOIRS:

R1;
R2;
R3;
R4;
#---------------------------------------
CONNECTIONS:

R1->inner.inlet;
inner.outlet->R2;
R3->outer.inlet;
outer.outlet->R4;
inner.wall->outer.wall;
```

Figure 6.14: Language Declaration for Double Pipe Heat Exchanger - Case 4

6.3.2 Modelling and Validation

The generated mathematical models for the four cases are given in appendices E.3 - E.6. The models for each case consist of seven (sub-)models on two hierarchical levels. The bottom level contains four reservoir models (`R1`, `R2`, `R3` and `R4`) which define the states of the system boundaries and two vessel models (`inner` and `outer`) describing the control volumes enclosed by the inner pipe and annulus respectively. Details of the vessel models for each case are given below. The top level contains a single model (`FLOWSHEET`) which comprises equations describing the vessel-vessel and reservoir-vessel connections. Specific details of the former for each case are given below. The latter are the same in all cases and take the form of simple equality relations matching variables across the open connections. The (steady state) numerical specifications for each case are summarised in table 6.5.

Model	Quantity	Values	Units
`R1`	p	2	atm
	T	300	K
	w	9820	lb/hr
`inner`	$\hat{C}_p$	988.6	J/kg K
	D	4.21	cm
	k	0.1575	W/m K
	L	36	m
	γ	0.88	-
	μ	0.5×10^{-3}	Pa s
`R3`	p	2	atm
	T	344	K
	w	6330	lb/hr
`outer`	$\hat{C}_p$	1023.4	J/kg K
	D	5.23	cm
	k	0.1471	W/m K
	L	36	m
	γ	0.87	-
	μ	0.41×10^{-3}	Pa s

Table 6.5: Numerical Specifications for Concentric Pipe Heat Exchanger

Note that in the absence of phase changes and with constant physical properties simple equations of state for enthalpy (6.10) are used in all cases relative to an (arbitrary) datum of 298 K.

$$\hat{h} = \hat{C}_p \Delta T \tag{6.10}$$

All models are solved (numerically) in the *gPROMS* modelling environment. For dynamic systems comprising differential and algebraic equations (DAEs) (cases 1 and 2) time integration is carried out using the DAE solver DASOLV [Jarvis, 1993]. Distributed parameter models comprising integral, partial differential and algebraic equations (IP-DAEs) (cases 2, 3 and 4) are solved using a spatial discretisation scheme of simple first order backward finite differences. If dynamic, time integration of the resulting (discretised) DAE system is then carried out (within *gPROMS*) as before using DASOLV [Oh, 1995].

6.3.2.1 Case 1

The mathematical model for the first case is given in appendix E.3. Both vessel models for this case consist of dynamic, constant density mass, energy and linear momentum balances together with a set of constitutive equations and an indication of the functional form of the equation of state for enthalpy. At the top level the inter-vessel connection is described using a film heat transfer model with an overall heat transfer coefficient composed of the two film coefficients on either side.

The dynamic behaviour of the system is illustrated using a hypothetical operating policy. Initially, the fluids in both the inner pipe and annulus are at the benzene inlet temperature of (approximately) 300 K. After a time, the hot toluene entering the annulus is raised to its operating temperature linearly over a two minute time period. As the temperature of the toluene increases, a temperature difference between the inner pipe and annulus is established and heat is transferred to the benzene. At steady state (after approximately 300 seconds) the exit temperatures of the benzene and toluene are maintained at 314.5 K and 421.5 K respectively, giving an overall temperature difference driving the heat transfer of 7 K. A plot of the exit temperatures of the two fluids is shown in figure 6.15.

The steady state behaviour demonstrates the limitation of using perfectly mixed macroscopic models. The temperature driving force is under-predicted as compared with the logarithmic mean temperature difference of 16 K calculated using Kern's method. Consequently, the heating duty is not satisfied and the target benzene temperature of 322 K is not achieved.

6.3.2.2 Case 2

The mathematical model for case 2 is given in appendix E.4. In this case dynamic one dimensional conservation equations are used in the model for the inner pipe to account for the axial variation of temperature. As a consequence, the inter-vessel heat transfer in

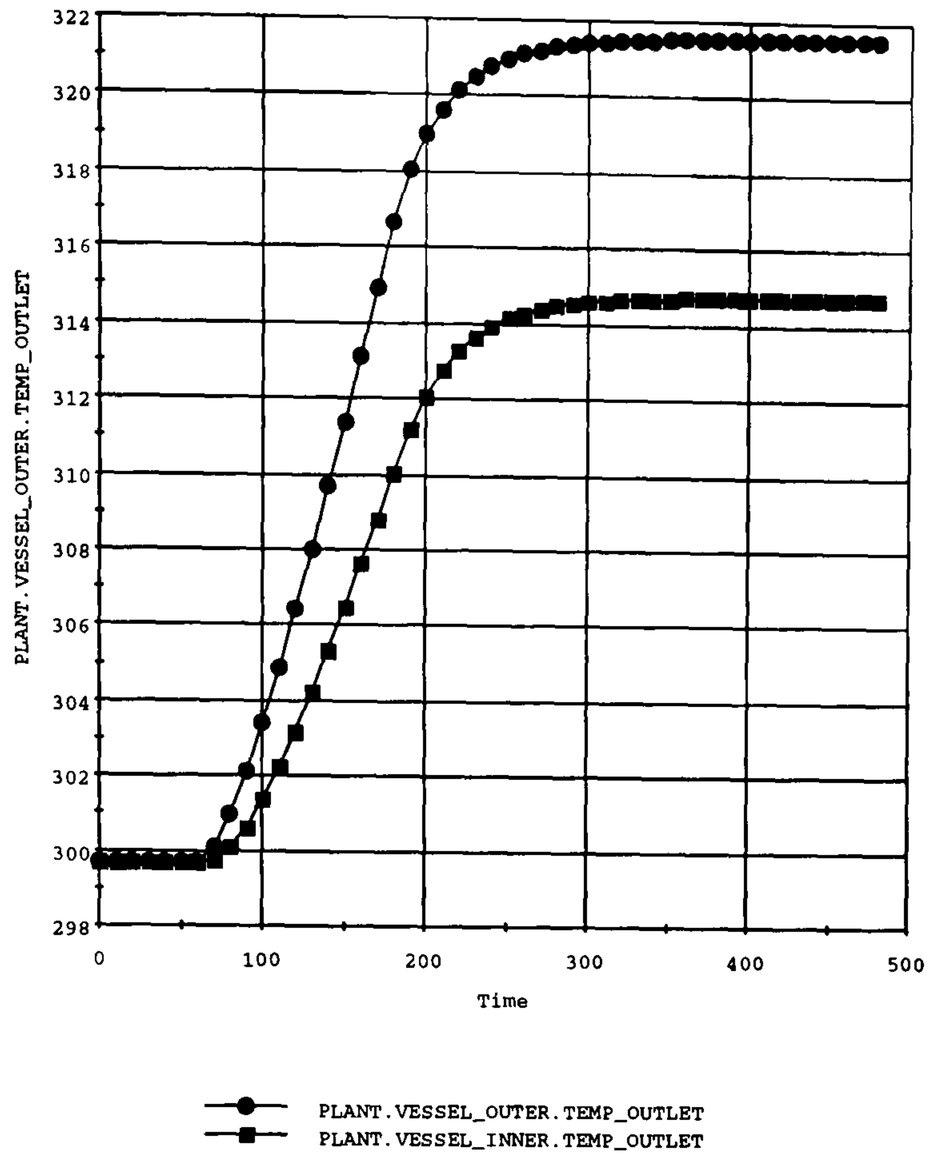


Figure 6.15: Temperature [K] versus Time [s]

the top-level model is described in terms of a spatially averaged equation for dimensional consistency. The model of the annulus and the (dynamic) operating policy are the same as for case 1.

Plots of the axial temperature profile of the inner pipe at five discrete points in time and the toluene exit temperature versus time are shown in figures 6.16 and 6.17 respectively. These graphs show the temperature profiles developing to their steady state values. In this case, the axially varying temperature in the inner pipe model gives a more realistic description of the temperature driving force and consequently at steady state more heat transfer is achieved than for case 1. However, the macroscopic model of the annulus means that the mean driving force is still less than for the base case and the heating requirement is still not satisfied.

6.3.2.3 Case 3

The model for case 3 is given in appendix E.5. In this case one dimensional models are generated for both the inner pipe and annulus and, at the top-level, the equations describing the inter-vessel heat transfer are set up to model a cocurrent flow arrangement.

The steady state benzene and toluene axial temperature profiles are given in figure 6.18. It shows (as expected) that the temperature profiles rapidly approach each other and little heat exchange is achieved in the last third of the exchanger. The result of

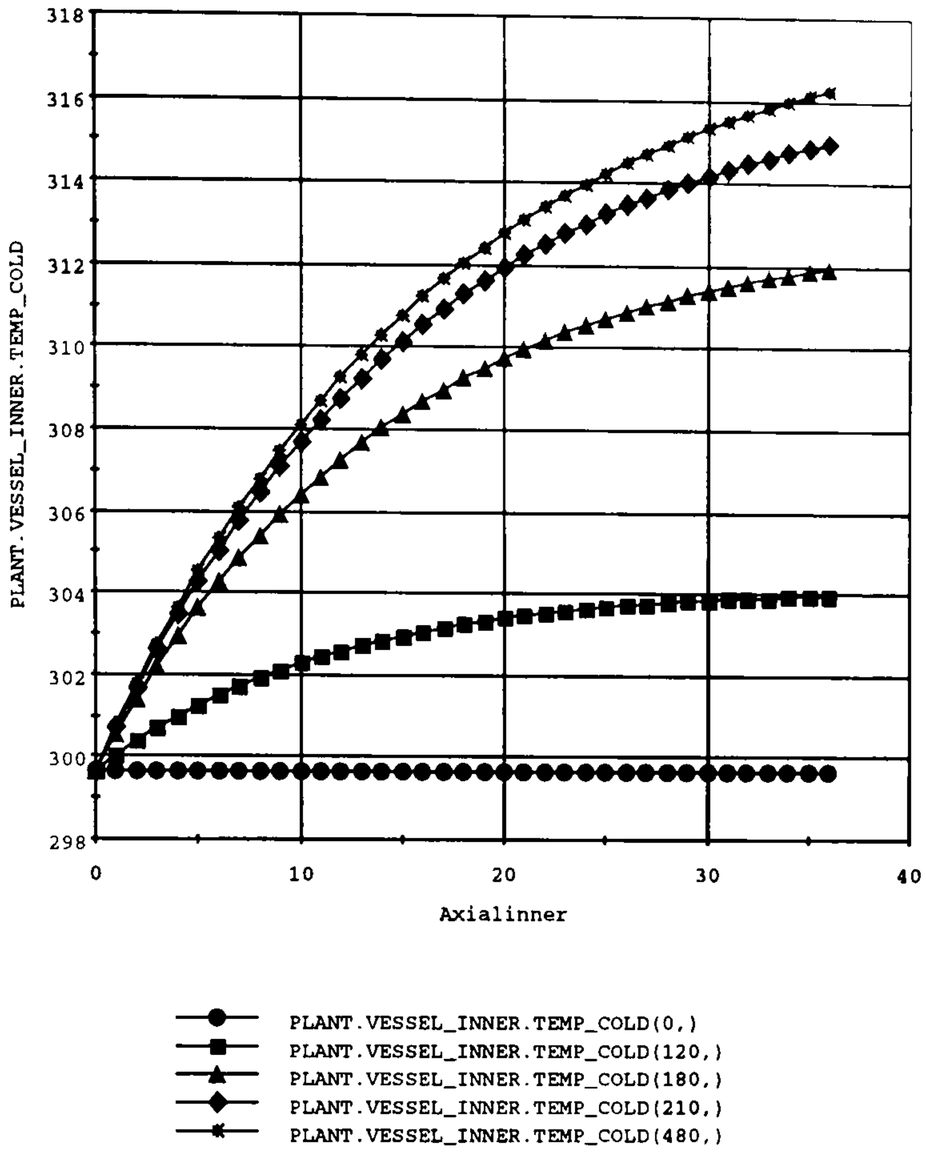


Figure 6.16: Temperature [K] (at different times) versus Axial Position [m]

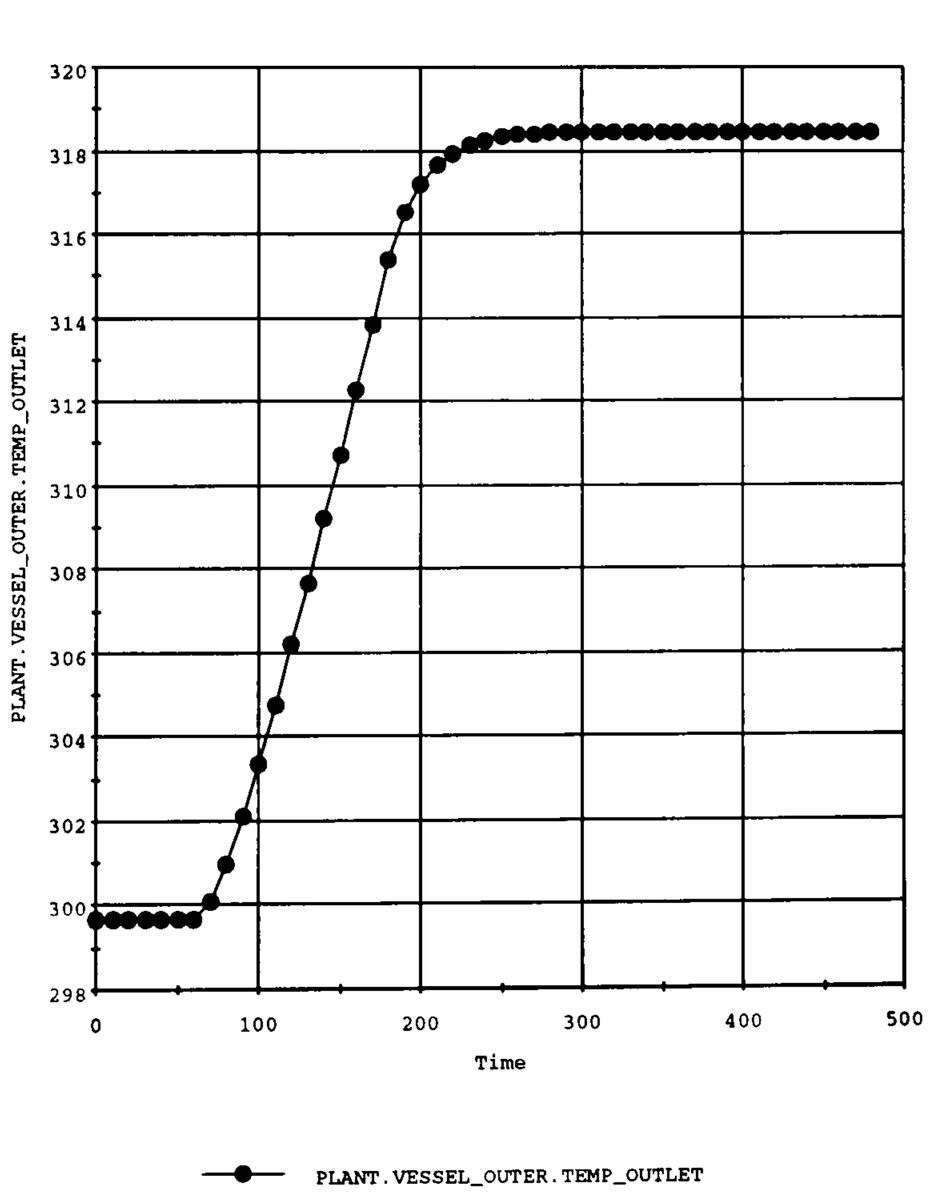


Figure 6.17: Temperature [K] versus Time [s]

the cocurrent flow arrangement is that the heating requirement is not satisfied since the available temperature driving force for heat transfer is used in a non-optimal manner.

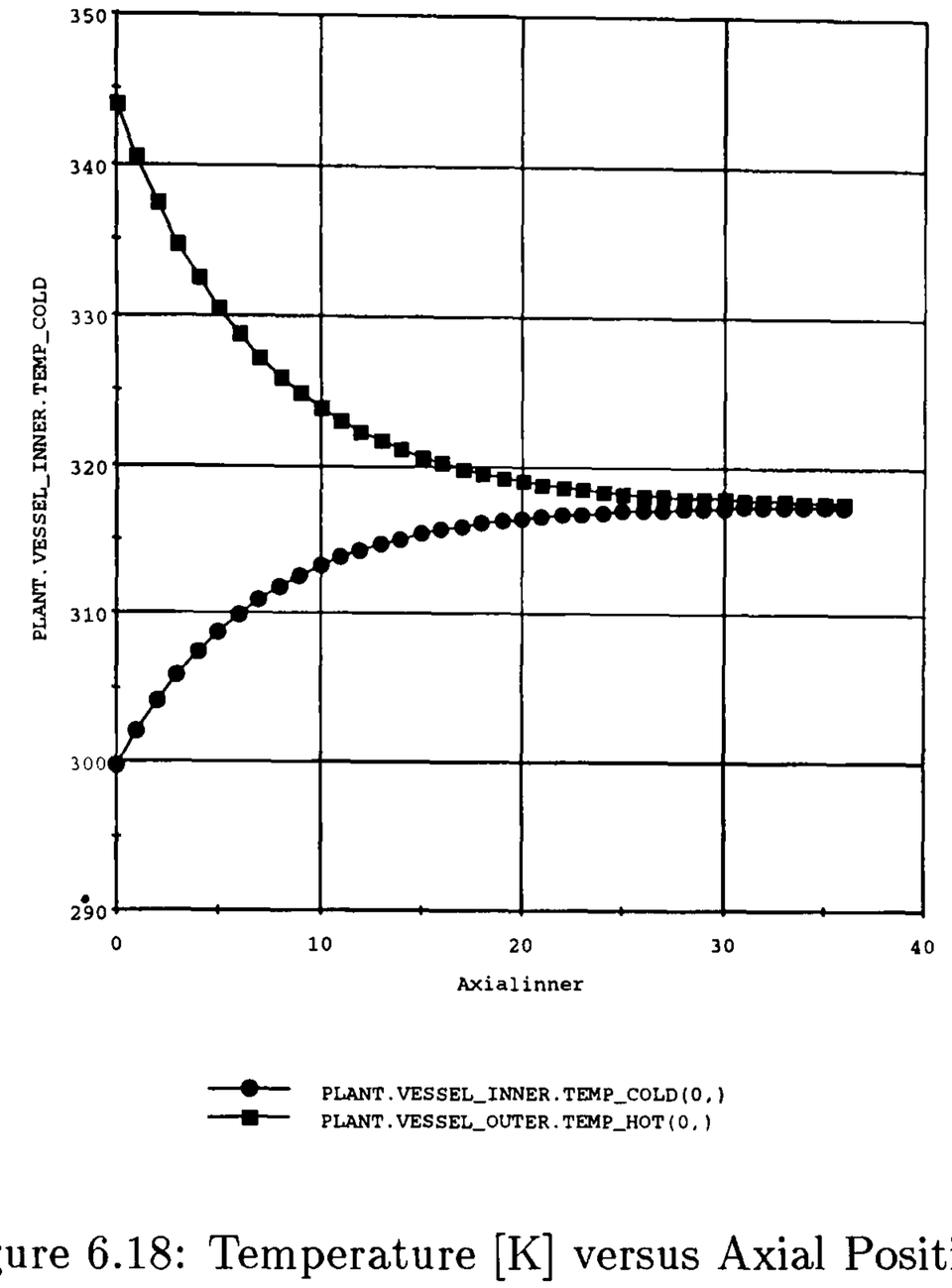


Figure 6.18: Temperature [K] versus Axial Position [m]

6.3.2.4 Case 4

The model for the final case is given in appendix E.6. Here the system is modelled in such a way as to simulate the results obtained from the base case solution more closely. One dimensional models of the inner pipe and annulus are used to account for the axial variation of temperature in both and, at the top-level, the equations describing the inter-vessel heat transfer are set up to model a countercurrent flow arrangement.

The steady state results are shown in figure 6.19, with the positive axial direction based on the inner pipe. This gives temperature profiles which are typical of countercurrent heat exchangers and illustrates the need to use logarithmic mean temperature differences in Kern's method. The graph also shows that, since the thermal resistance of the inter-vessel wall is ignored, the benzene target temperature is surpassed and toluene is cooled to a lower temperature than in the base case. These results therefore indicate that the required heating duty could be achieved with a lower toluene flow rate.

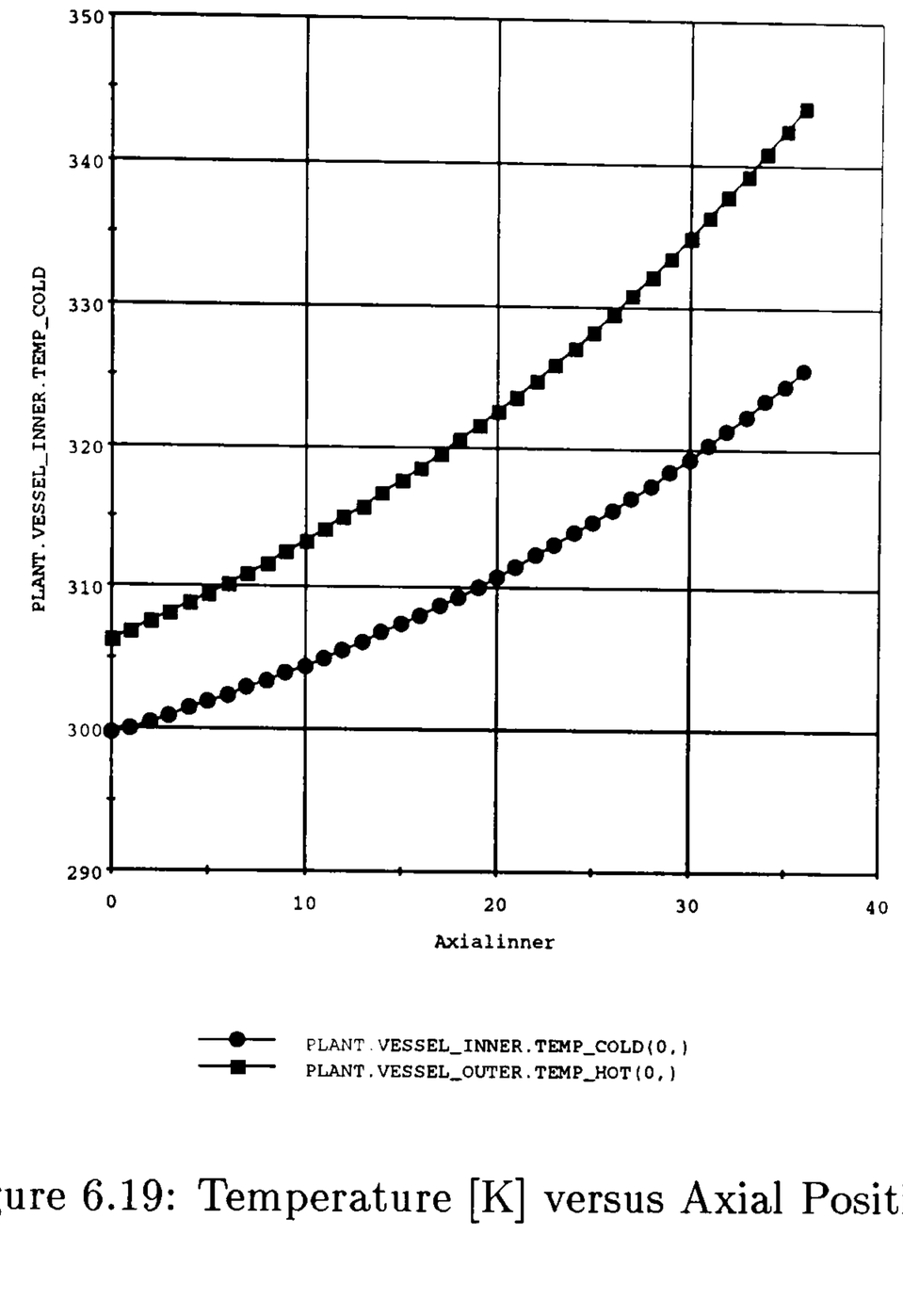


Figure 6.19: Temperature [K] versus Axial Position [m]

6.3.2.5 Summary

Table 6.6 summarises the results from the base case and each of the four cases considered above. The mean temperature differences given in the table are an approximation to the logarithmic mean calculated from (6.11) [Underwood, 1933]. This form of equation allows calculation for the cocurrent flow case where the exit temperatures from both vessels are (almost) the same since it avoids having to take the logarithm of zero.

$$\theta_m^{\frac{1}{3}} = \frac{1}{2}(\theta_1^{\frac{1}{3}} + \theta_2^{\frac{1}{3}}) \tag{6.11}$$

Case	T_{Benzene} In	T_{Benzene} Out	T_{Toluene} In	T_{Toluene} Out	θ_m	Target Reached ?	Results
Base	300	322	344	311	16	Yes	(Kern's method)
1	300	314.5	344	321.5	7	No	Fig. 6.15
2	300	316.2	344	318.5	7.8	No	Figs. 6.16 and 6.17
3	300	317.5	344	317.5	5.5	No	Fig. 6.18
4	300	325	344	307	12	Yes (surpassed)	Fig. 6.19.

Table 6.6: Summary of Results from Four Cases of Double Pipe Heat Exchanger

6.4 Laminar Pipe Flow

In this example the steady, isothermal flow of an incompressible Newtonian fluid through a pipe at low Reynolds number is considered. The fluid enters the pipe with a uniform velocity across its cross-section but is then immediately retarded by the viscous effects at the wall. Boundary layers form and gradually thicken as the distance from the entry point increases. Since the fluid in the boundary layers is retarded and the total flow rate remains constant, the fluid in the central stream is accelerated. At some distance from the entrance, the entrance length, the boundary layers at the walls join at the pipe axis to occupy the whole of the cross-section and from that point on remain at constant thickness. The flow is then said to be fully developed. The radial velocity profile at this point is then sustained downstream with a maximum value at the centreline.

At low Reynolds numbers the boundary layers are still streamline when fully developed flow commences and consequently the flow from that point on remains streamline. In such cases, for the fully developed region, an analytical solution (the famous Hagen-Poiseuille law) gives the maximum velocity as twice the inlet velocity and also leads to equation (6.12) for the radial velocity distribution.

$$\frac{v_z}{v_{z,max}} = \left[1 - \left(\frac{r}{R}\right)^2\right] \tag{6.12}$$

In the region of developing flow the nonlinear nature of the governing equations precludes the existence of an analytical solution. However, approximate solutions have led to a number of dimensionless correlations for determining the entrance length in terms of the pipe diameter and Reynolds number (see, for example, Schlichting [1968]).

The particular case of developing laminar flow given here is based on data from Lecomte [1998] and considers the flow of a viscous Newtonian liquid (glycerol) through a pipe of diameter 0.04 m and length 0.06 m at a (very low) Reynolds number of approximately 0.5. Physical properties are assumed to be constant and calculated from data by Reid *et al.* [1987].

6.4.1 Process Representation

The process representation for this example is straightforward as shown below (figure 6.20). It consists simply of a single vessel (`pipe`) connected to two reservoirs representing the upstream and downstream boundaries (`R1` and `R2` respectively) via open ports. The language declaration is given in figure 6.21. The pipe is modelled as a horizontally oriented cylinder containing a single, incompressible liquid phase (`glycerol`).

For this example a two dimensional model is generated and a laminar flow regime is indicated. The flow is also assumed to be isothermal and at steady state. Note that

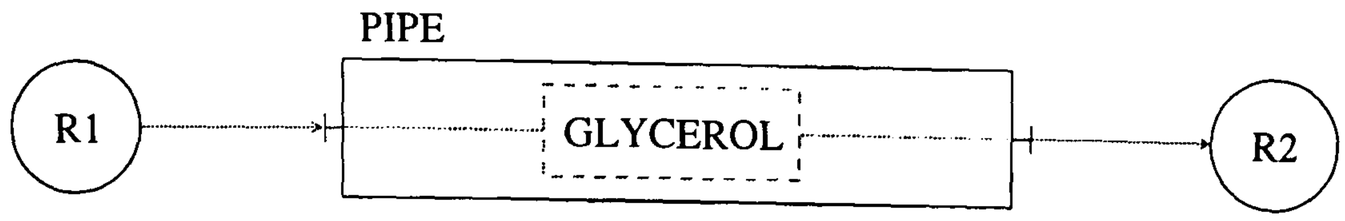


Figure 6.20: Process Representation for Laminar Pipe Flow

the process representation for this case is close to that for choking flow (see section 6.1) but the modelling assumptions give rise to a different form of mathematical model.

```
VESSEL: pipe: 2-D;

GEOMETRY: CYLINDER [AXIAL = 0, 0.06: RADIAL = 0, 0.02];

PHASES: glycerol;

PORTS: inlet  [AXIAL = 0: RADIAL = 0, 0.02]: OPEN;
       outlet [AXIAL = 0.06: RADIAL = 0, 0.02]: OPEN;

ASSUMPTIONS: ISOTHERMAL;
             INCOMPRESSIBLE;
             LAMINAR;
             STEADY_STATE;
#-----------------------------------------
RESERVOIRS:
R1;
R2;
#-----------------------------------------
CONNECTIONS:

R1->pipe.inlet;
pipe.outlet->R2;
```

Figure 6.21: Language Declaration for Laminar Pipe Flow

6.4.2 Modelling and Validation

The mathematical model generated for this example is given in appendix E.7. It consists of four (sub-)models on two hierarchical levels. The bottom level contains two reservoir models (`R1` and `R2`) which define the states of the upstream and downstream boundaries and a two dimensional vessel model (`pipe`) describing the control volume enclosed by the pipe. The latter consists of a steady state, constant density continuity equation and two (orthogonal) linear momentum balances but, because the system is isothermal, does not include a balance equation for energy. In addition the model contains boundary conditions describing no-slip at the wall and axial symmetry and also constitutive equations for the

two dimensional Newtonian shear stresses. Since laminar flow is assumed the conservation equations are exact (unaveraged) and hence the Reynolds stress terms are omitted. The top level contains a single model (FLOWSHEET) which describes the connections between the reservoirs and the vessels. At the vessel inlet these consist of relations equating the fixed variables (density and temperature) to those of the upstream reservoir and also equations defining flat spatial profiles of distributed velocity and pressure variables. At the outlet zero tangential velocity gradients define the exit boundary conditions whilst the fixed and distributed variables are related either directly or by spatial averaging to their (downstream) reservoir counterparts. The numerical specifications for this example are summarised in table 6.7.

Model	Quantity	Values	Units
R1	p	100009	Pa (g)
	T	298	K
	v	0.01	m/s
pipe	D	0.04	m
	L	0.06	m
	ρ	1261	kg/m^3
	μ	0.9336	Pa s

Table 6.7: Numerical Specifications for Laminar Pipe Flow

The mathematical model generated by gMaGiC for this example cannot be solved directly in the *gPROMS* modelling environment.

In *gPROMS*, solution domains for systems of PDAEs are spatially discretised using simple rectangular grids. The continuous PDAE system is then approximated by a square set of discrete equations and variables applied at these grid points. To give a square (discretised) equation set, boundary conditions are applied only at the grid points that lie on the boundaries of the solution domain and the other differential and algebraic equations usually at both the interior points and also (necessarily) at selective boundary points. However, such a discrete approximation takes no account of the continuity and conservation properties which constrain the continuous solution over the entire extent of the solution domain.

In this example, to provide a square discretised system, it is necessary to enforce the continuity and radial momentum equations only at interior points but, because pressure boundary conditions are only required at one end of the axial and radial domains, the axial momentum equation must be enforced at both the interior points and also at some of the boundary points. Thus, at these boundary points, the discrete approximation ignores the fact that the axial momentum equation must be consistent with the continuity

equation (since the latter is not applied there). Consequently, the discrete approximation fails to converge to a solution.

Fortunately, in this example, the viscosity and density are constant and it is possible to reformulate the (continuous) model by replacing the radial and axial linear momentum equations generated by gMaGiC (appendix E.7) with (equivalent) simplified forms (6.13 and 6.14 respectively), derived by substituting for the shear stresses and simplifying using the continuity equation.

$$\frac{\partial}{\partial z}\rho v_z v_r + \frac{1}{r}\frac{\partial}{\partial r} r\rho v_r v_r + \frac{\partial p}{\partial r} - \mu\left(\frac{\partial}{\partial r}\frac{1}{r}\frac{\partial}{\partial r} r v_r + \frac{\partial^2 v_r}{\partial z^2}\right) = 0 \qquad (6.13)$$

$$\frac{\partial}{\partial z}\rho v_z v_z + \frac{1}{r}\frac{\partial}{\partial r} r\rho v_r v_z + \frac{\partial p}{\partial r} - \mu\left(\frac{1}{r}\frac{\partial}{\partial r} r\frac{\partial v_z}{\partial r} + \frac{\partial^2 v_z}{\partial z^2}\right) = 0 \qquad (6.14)$$

These equations have the advantage of being consistent with the continuity equation throughout the solution domain and, consequently, the discrete approximation to the reformulated (continuous) model is also consistent not only at the interior points but also at the boundaries. The reformulated model is solved using a discretisation scheme based on one-sided and centred finite differences and yields the results given below.

Figure 6.22 shows the radial variation of the axial velocity component at different axial positions.

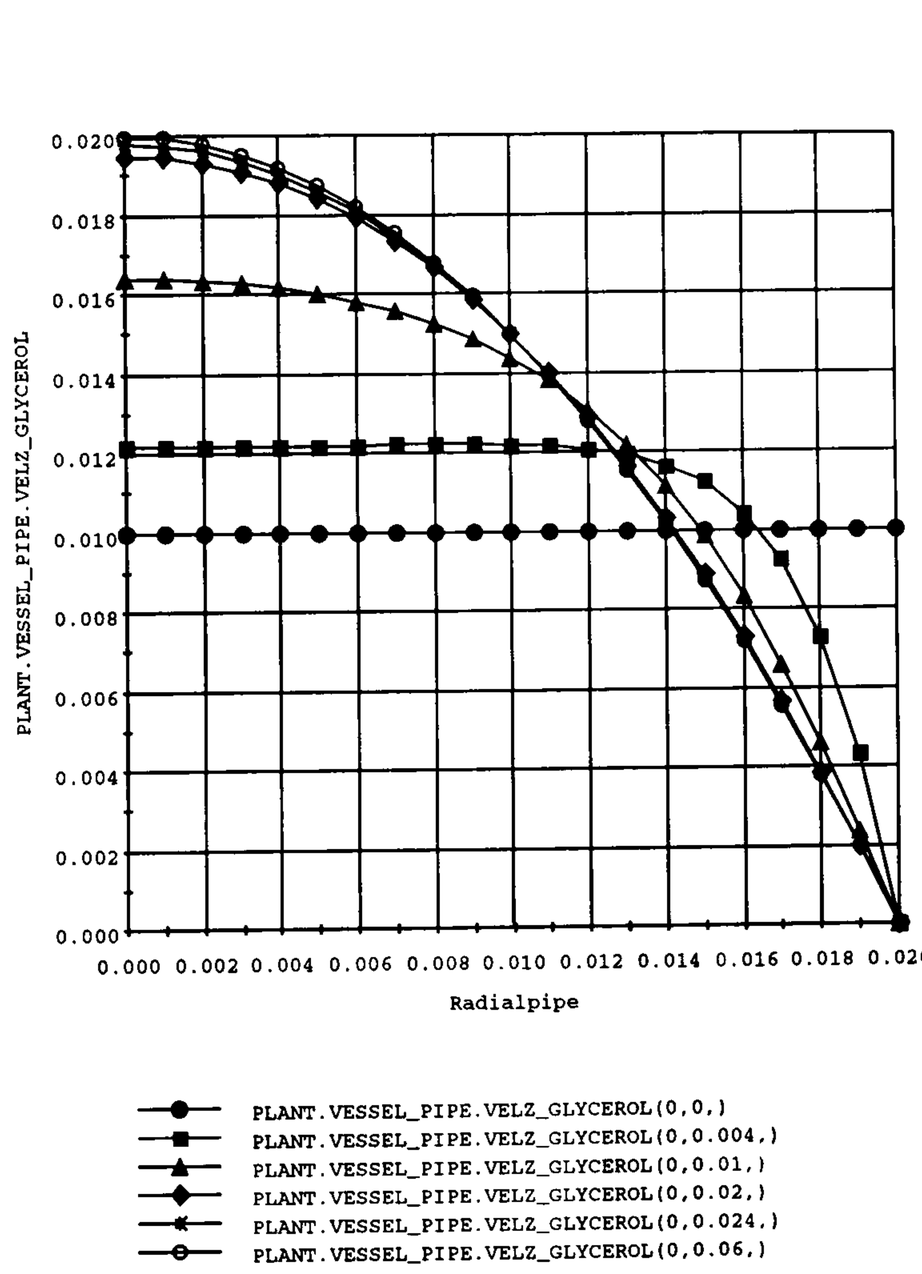


Figure 6.22: Axial Velocity [m/s] (at different axial positions) versus Radial Position [m]

These curves are qualitatively typical of those given in standard texts (see, for example, Bird *et al.* [1960] or Coulson and Richardson [1990]) and are quantitatively both in close agreement with those obtained by Lecomte [1998] and, for the fully developed region. consistent with the analytical solution (6.12) mentioned above.

Figure 6.23 shows the axial variation of the centreline axial velocity component. This illustrates graphically the entrance length for the pipe (defined as the distance downstream of the entrance at which the centreline velocity reaches 99 % of the maximum) which is calculated as approximately $1.2D$. This result is in good agreement with the findings of Dombrowski *et al.* [1993] for low Reynolds numbers and is again close to that obtained by Lecomte [1998].

Figure 6.24 gives the axial variation of the radial velocity component at the centreline. This shows a negative velocity near the entrance (indicating the region where the developing boundary layers cause the central portion of the fluid to be accelerated) which tends to zero downstream as the flow becomes fully developed. Again this result is in good agreement with that of Lecomte [1998].

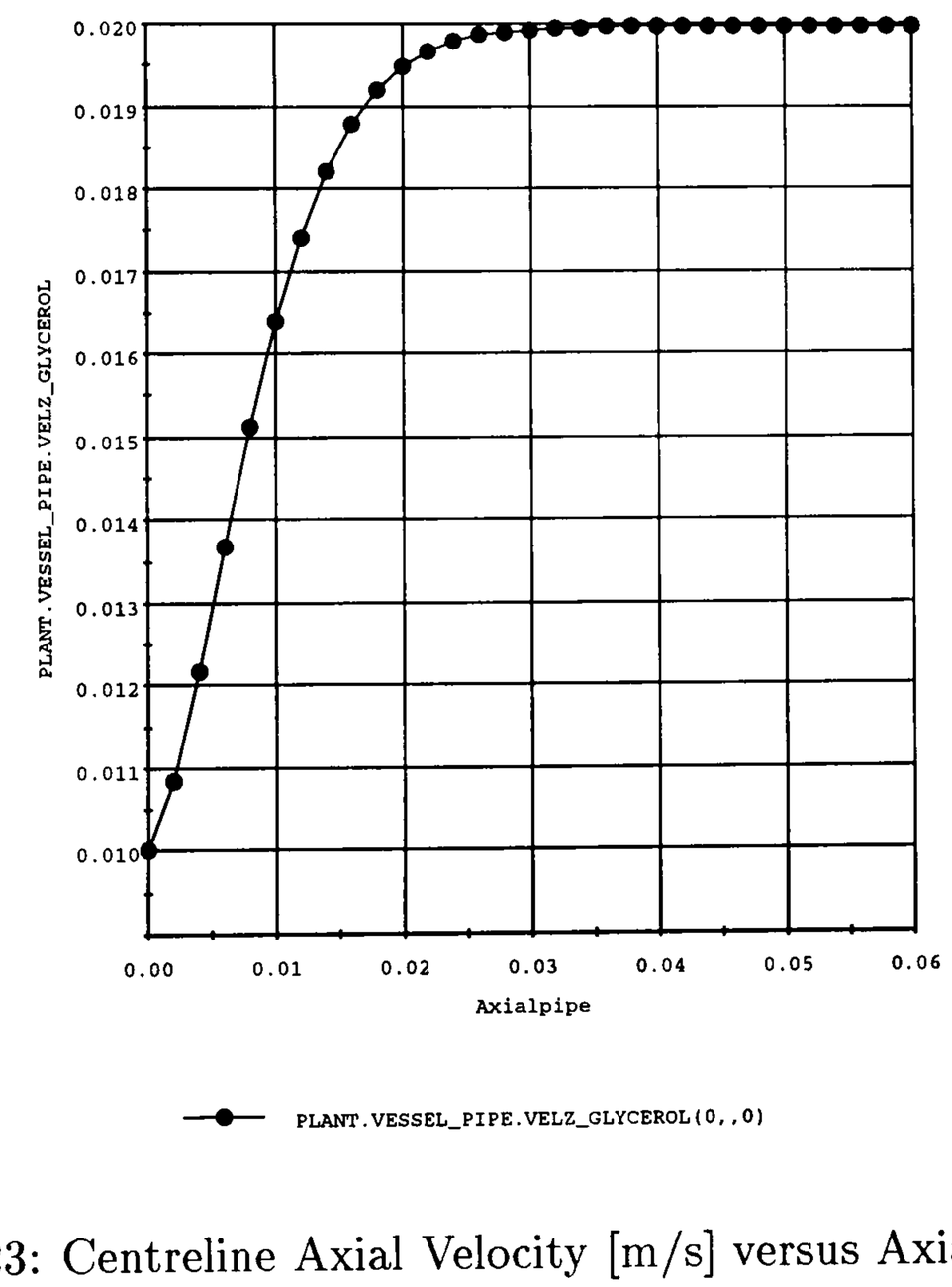


Figure 6.23: Centreline Axial Velocity [m/s] versus Axial Position [m]

Finally, figure 6.25 gives the axial pressure profile at the pipe centreline. This shows a characteristic maximum downstream of the entrance identified by Dombrowski *et al.* [1993] and an overall pressure drop close to that obtained by Lecomte [1998].

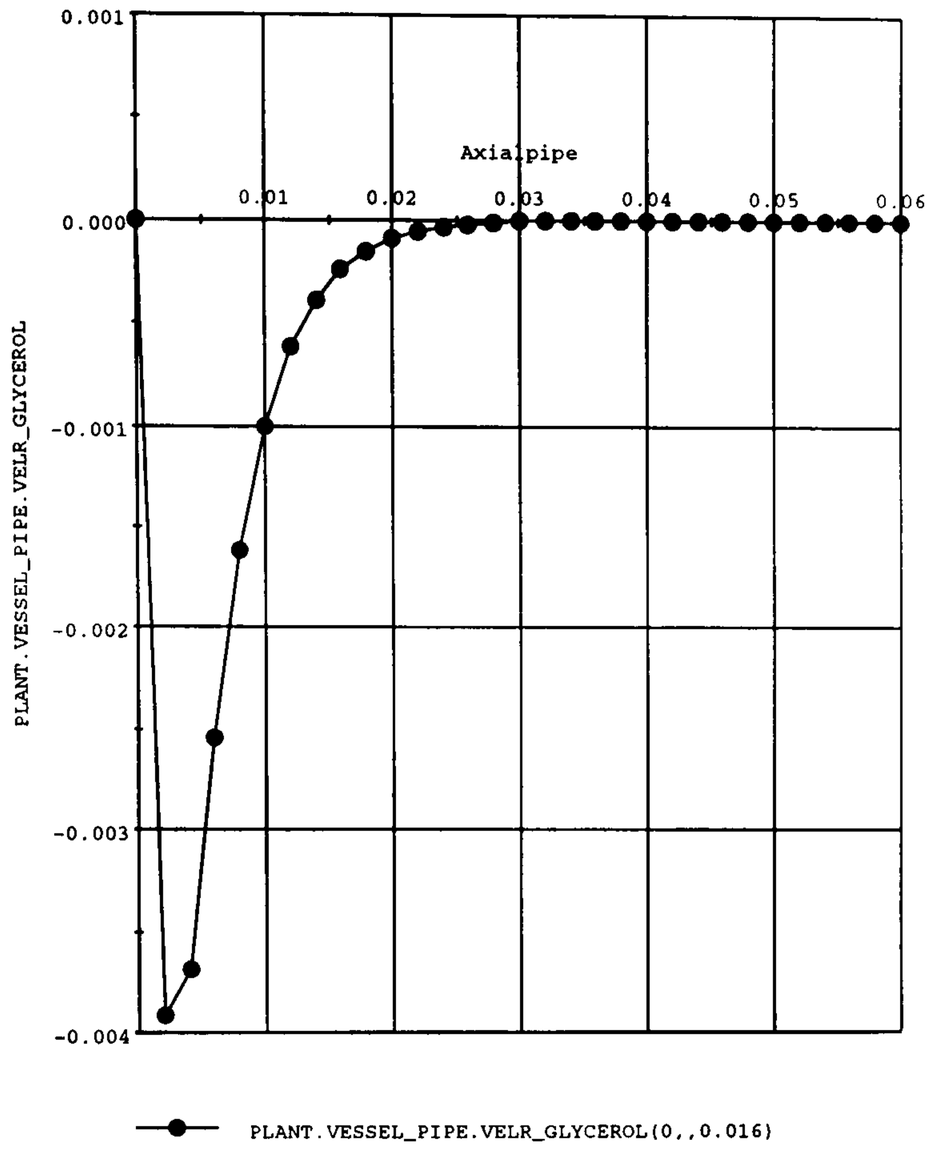


Figure 6.24: Radial Velocity [m/s] (at r = 0.8R) versus Axial Position [m]

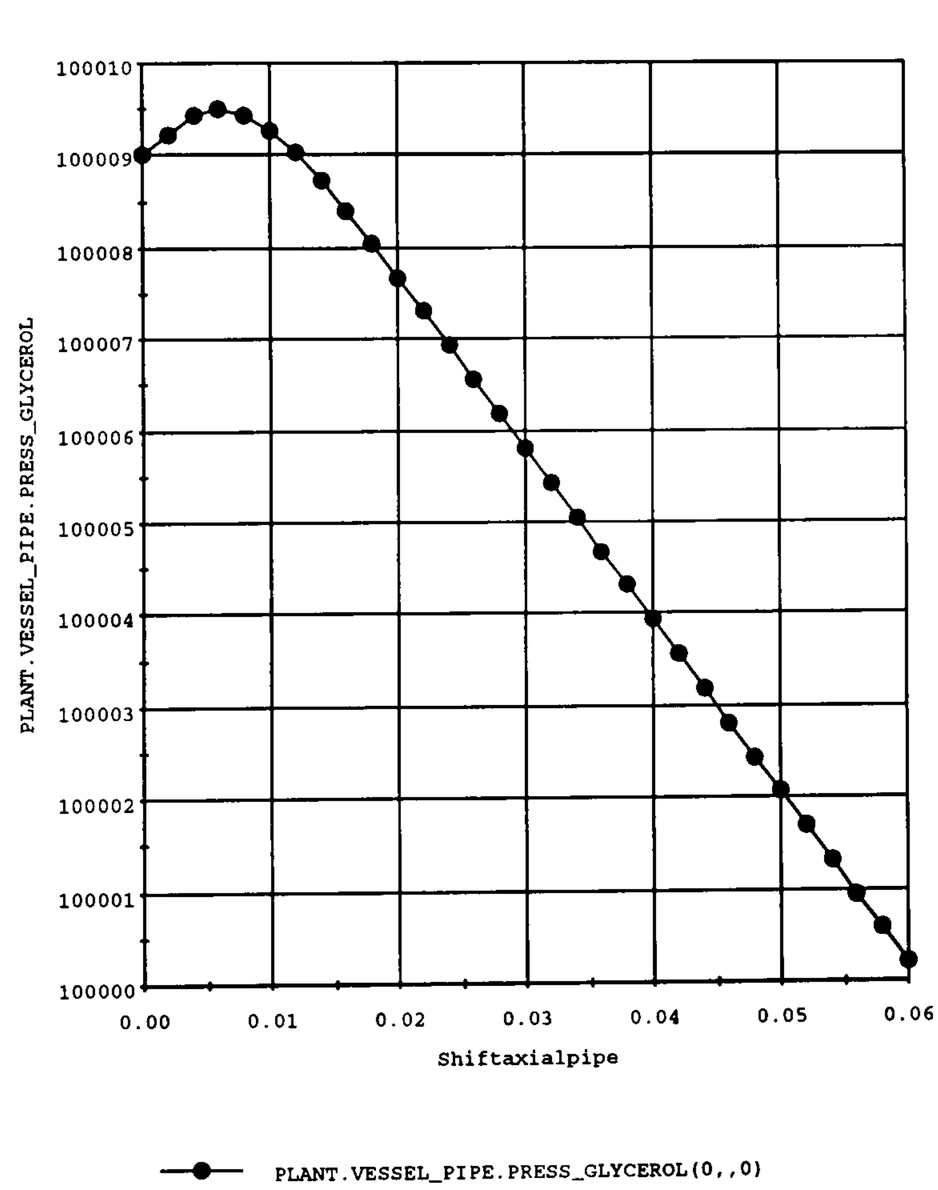


Figure 6.25: Centreline Pressure [Pa] versus Axial Position [m]

6.5 Turbulent Pipe Flow

For the final example the steady, turbulent flow of an incompressible fluid through a pipe of constant cross-section is considered. This example illustrates the use of an eddy viscosity and simple turbulence model to describe the mean properties in a turbulent flow field. The physical situation is similar to the laminar pipe flow example considered above (see section 6.4) but the higher flow velocity at the inlet is such that turbulent boundary layers are formed initially and, consequently, a turbulent flow regime is established at the point where the flow becomes fully developed.

No exact mathematical analysis of turbulent flows is yet available but it has been shown experimentally that the maximum centreline velocity is roughly 5/4 of the average velocity and that the radial velocity distribution in high Reynolds number flows is well-approximated by the Prandtl one-seventh power law (see, for example, Bird *et al.* [1960]).

$$\frac{v_z}{v_{z,max}} \approx \left[1 - \left(\frac{r}{R}\right)^{\frac{1}{7}}\right] \tag{6.15}$$

The particular example of turbulent pipe flow given here is based on data from Coulson and Richardson [1990] and considers the isothermal flow of water through a pipe of length 100 m and diameter 50 mm subject to an overall pressure drop of 50 kPa. In their analysis, Coulson and Richardson use a graphical plot of Reynolds number versus friction group to calculate an overall cross-sectional average velocity of approximately 1.6 m/s for walls with a relative roughness of 2.6×10^{-4}. For the purposes of the (numerical) analysis considered here, an algebraic mixing length model is used for the prediction of the turbulent viscosity with the assumption of smooth walls and, for simplicity, the density and molecular viscosity of the water are taken as constant using data from Coulson and Richardson [1990].

6.5.1 Process Representation

The process representation for this example (figure 6.26) is almost identical to that for the laminar case.

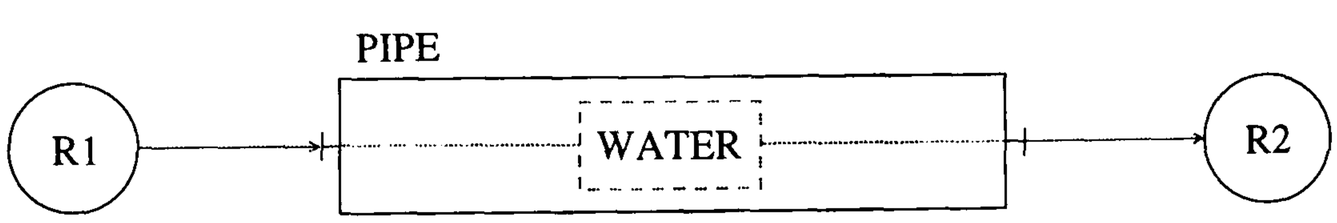


Figure 6.26: Process Representation for Turbulent Pipe Flow

It again consists simply of a single vessel (`pipe`) containing a pure fluid phase (`water`) connected to two reservoirs representing the upstream and downstream boundaries (`R1`

and R2 respectively). The main difference (apart from the fluid type and pipe dimensions) is that the simplification of laminar flow is omitted, indicating that a full averaged model including terms arising from Reynolds stresses should be generated. The language declaration for this case is given in figure 6.27.

```
VESSEL: pipe: 2-D;

GEOMETRY: CYLINDER [AXIAL = 0, 100: RADIAL = 0, 0.025];

PHASES: water;

PORTS: inlet  [AXIAL = 0: RADIAL = 0, 0.025]: OPEN;
       outlet [AXIAL = 100: RADIAL = 0, 0.025]: OPEN;

ASSUMPTIONS: ISOTHERMAL;
             INCOMPRESSIBLE;
             STEADY_STATE;
#-----------------------------------------
RESERVOIRS:

R1;
R2;
#-----------------------------------------
CONNECTIONS:

R1->pipe.inlet;
pipe.outlet->R2;
```

Figure 6.27: Language Declaration for Turbulent Pipe Flow

6.5.2 Modelling and Validation

The mathematical model generated for this example is given in appendix E.8. It is close to that for laminar pipe flow given above and differs only in the model of the vessel pipe. In this case, where no assumptions are made regarding the flow regime, the linear momentum equations are constructed in terms of effective shear stresses which consist of both molecular and Reynolds stress components. Consequently, the model includes additional constitutive equations describing this relation and also relating the Reynolds stresses to the averaged variables using an eddy-viscosity model. The numerical specifications for this example are summarised in table 6.8.

As with the case of laminar pipe flow, the mathematical model generated by gMaGiC for this example cannot be solved directly in the *gPROMS* modelling environment due to inconsistencies introduced by the discrete approximations (see section 6.4). However, in this case the variable (turbulent) viscosity precludes a simple model reformu-

Model	Quantity	Values	Units
pipe	D	0.05	m
	L	100	m
	$-\Delta P$	50	kPa
	T	298	K
	ρ	1000	kg/m^3
	μ	1×10^{-3}	Pa s

Table 6.8: Numerical Specifications for Turbulent Pipe Flow

lation and therefore an alternative method of model validation must be employed. The method used here is based on a semi-analytical approach whereby the model equations generated by gMaGiC are simplified for the region of fully developed flow and the reduced equations solved in *gPROMS*. The results for fully developed flow are then compared and shown to be consistent with both the Prandtl one-seventh power law (6.15) and the average velocity obtained by Coulson and Richardson [1990]. Both of these results are based on fully developed flows inasmuch as they do not consider entrance or exit effects.

In the region of fully developed flow

$$\frac{\partial v_z}{\partial z} = 0 \tag{6.16}$$

so that $v_z = v_z(r)$. The mass conservation equation (from appendix E.8) dictates

$$\frac{\partial v_z}{\partial z} + \frac{1}{r}\frac{\partial}{\partial r}(r v_r) = 0 \tag{6.17}$$

so

$$\frac{1}{r}\frac{\partial}{\partial r} r v_r = 0 \tag{6.18}$$

This can be integrated to give

$$r v_r = 0 \tag{6.19}$$

and from the boundary condition at the wall (appendix E.8)

$$v_r = 0 \qquad @\ r = R \tag{6.20}$$

so

$$v_r = 0 \tag{6.21}$$

The linear momentum equations (appendix E.8) are

$$\frac{\partial}{\partial z}\rho v_z v_r + \frac{1}{r}\frac{\partial}{\partial r}r\rho v_r v_r + \frac{\partial p}{\partial r} - \frac{\partial}{\partial z}\tau^e_{rz} - \frac{1}{r}\frac{\partial}{\partial r}r\tau^e_{rr} + \frac{\tau^e_{\theta\theta}}{r} = 0 \tag{6.22}$$

$$\frac{\partial}{\partial z}\rho v_z v_z + \frac{1}{r}\frac{\partial}{\partial r}r\rho v_r v_z + \frac{\partial p}{\partial z} - \frac{\partial}{\partial z}\tau^e_{zz} - \frac{1}{r}\frac{\partial}{\partial r}r\tau^e_{rz} \tag{6.23}$$

Using an eddy-viscosity model (appendix E.8) and substituting from (6.16) and (6.21)

$$\tau^e_{zz} = 2(\mu + \mu_t)\frac{\partial v_z}{\partial z} = 0 \tag{6.24}$$

$$\tau^e_{rr} = 2(\mu + \mu_t)\frac{\partial v_r}{\partial r} = 0 \tag{6.25}$$

$$\tau^e_{rz} = (\mu + \mu_t)\left(\frac{\partial v_z}{\partial r} + \frac{\partial v_r}{\partial z}\right) = (\mu + \mu_t)\frac{\partial v_z}{\partial r} \tag{6.26}$$

$$\tau^e_{\theta\theta} = 2(\mu + \mu_t)\frac{v_r}{r} = 0 \tag{6.27}$$

The momentum equations (6.22 and 6.23) therefore reduce to

$$\frac{\partial p}{\partial r} - \frac{\partial}{\partial z}(\mu + \mu_t)\frac{\partial v_z}{\partial r} = 0 \tag{6.28}$$

$$\frac{\partial p}{\partial z} - \frac{1}{r}\frac{\partial}{\partial r}r(\mu + \mu_t)\frac{\partial v_z}{\partial r} = 0 \tag{6.29}$$

Using a simple mixing length formula for the turbulent viscosity

$$\mu_t = \rho l_m^2 \left|\frac{\partial v_z}{\partial r}\right| \tag{6.30}$$

where, for fully developed, smooth-walled pipe flows the mixing length l_m, is given by an amended form of Nikuradse's formula (6.31), which includes the van Driest modification accounting for the semi-laminar near-wall region [Patankar and Spalding, 1970; Launder and Spalding, 1972].

$$\frac{l_m}{R} = \left[0.14 - 0.08\left(\frac{r}{R}\right)^2 - 0.06\left(\frac{r}{R}\right)^4\right]\left[1 - \exp\left(-\frac{(R-r)\tau_w^{\frac{1}{2}}\rho^{\frac{1}{2}}}{26\mu}\right)\right] \tag{6.31}$$

Noting from (6.31) that $\mu_t = \mu_t(r)$, gives, from (6.28)

$$\frac{\partial p}{\partial r} = 0 \tag{6.32}$$

and hence $p = p(z)$ so

$$\frac{\partial p}{\partial z} = \frac{dp}{dz} = \frac{p_L - p_0}{L} = \frac{1}{r}\frac{\partial}{\partial r}r(\mu + \mu_t)\frac{\partial v_z}{\partial r} \tag{6.33}$$

Solving (6.30), (6.31) and (6.33) simultaneously in *gPROMS* using a numerical scheme of second order centred finite differences gives the well-known turbulent velocity profile shown in figure 6.28. This result (for smooth walls) gives a mean velocity of 1.64 m/s which is in excellent agreement with that obtained by Coulson and Richardson [1990]. The velocity profile is also in close agreement with (6.15) and gives a calculated average velocity consistent with experimental measurements of approximately 0.83 times the maximum (see, for example, Bird *et al.* [1960]).

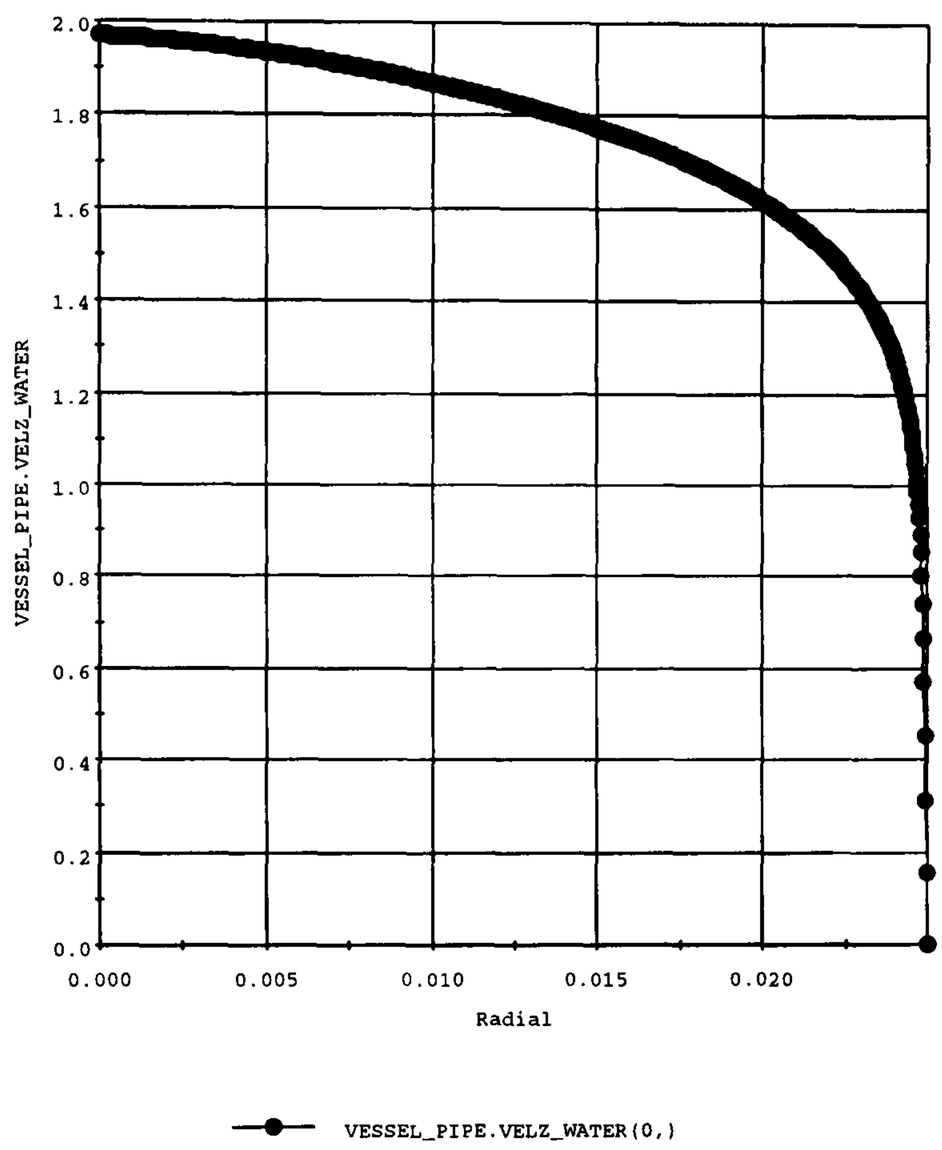


Figure 6.28: Fully Developed Axial Velocity [m/s] versus Radial Position [m]

6.6 Summary

In this chapter several examples have been presented to demonstrate and validate the modelling approach developed in this thesis. The examples show its success in modelling systems exhibiting a variety of physico-chemical behaviour. They include more traditional process models, such as those of the vinyl acetate reactor and double pipe heat exchanger, and also more detailed models where a description of the fluid mechanics is essential to accurately capture the physical behaviour, as with the examples of choking in compressible flow with friction and laminar and turbulent pipe flows.

In the final chapter conclusions to the research are drawn and some suggestions for future work are given.

Chapter 7

Conclusions

In this thesis a systematic methodology for the automatic generation of mathematical models of process systems has been developed and implemented in the computer program gMaGiC (*g*eneral *M*odel *G*eneration *C*ode).

To satisfy the project objectives the methodology has been divided into two main parts: the process representation and the model building procedure. The process representation is designed to permit the description of a process system in purely physical terms and in a manner which is consistent with the continuum hypothesis. The significance of the representation is that it completely decouples the physical description of a process system from the formulation of the mathematical equations describing its behaviour. The model building procedure is based on a rigorous theoretical approach to process modelling based on continuum mechanics and supports the generation of combined lumped and distributed parameter process models of varying complexity which are consistent with the underlying fluid mechanics.

The validity of the approach has been proven by its application to several non-trivial case studies. These have been chosen to demonstrate its success in modelling systems exhibiting a variety of physico-chemical behaviour including compressible and incompressible fluids, varying thermal characteristics (isothermal, adiabatic and finite heat transfer), laminar and turbulent flow regimes, single and multicomponent mixtures (reacting and inert) and operation under steady state and dynamic conditions.

7.1 Thesis Summary

The background to the work was given in chapter 1, which began with an introduction to the use of computer-aided modelling tools in chemical and process engineering. It was noted that the principal limitation of existing approaches is that they require the user to formulate an appropriate mathematical model for the particular process system of interest. Those few prototype software tools that have been reported in the literature which try to address this problem were then reviewed critically. An introduction to the field of computational fluid dynamics (CFD) was also given. It was noted that the development of computer-based CFD tools and of traditional process modelling tools has

proceeded largely independently of one another, despite the fact that both are dealing with systems based on the same physico-chemical behaviour. From this discussion, the major objective of this work was identified as the development of a software tool for the automatic generation of mathematical models of process systems from purely physical descriptions. A particular focus was established as the generation of models describing both spatial and temporal variation of the system properties which are consistent with the underlying fluid mechanics.

In chapter 2 the mathematical modelling of single and multiphase systems was considered, using a rigorous theoretical approach based on continuum mechanics. It focused on the formulation of models appropriate for the description of turbulent and multiphase systems and discussed the importance of averaging operations in the derivation of mathematical descriptions which are consistent with the continuum hypothesis. It was noted that, for multiphase systems in particular, there is considerable uncertainty in the formulation of not only the governing differential equations but, more importantly, in the closure relationships needed to resupply information on flow structure and distribution lost during averaging. Consideration was also given to the specification of appropriate boundary conditions and the derivation of simplified models. The material presented in this chapter forms the basis of the model generation approach developed in this thesis and is also used to define its scope. For the latter it was argued that the problems in the formulation of predictive multiphase models, particularly the closure issue, mean that the automatic generation of mathematical models of multiphase systems is not feasible at present and would not be considered in this thesis.

In chapter 3 a process representation was presented which permits the description of a process system in purely physical terms. The representation consists of two parts: a general framework supporting the decomposition of a process system into basic conceptual elements and a process modelling language encapsulating the framework and providing a formal means of describing a process system in terms of it. The conceptual framework is based on the premise that all material undergoing processing is instantaneously in a stable thermodynamic state, implying that it exists in well-defined, continuous thermodynamic phases. A process is therefore ultimately represented as a collection of interacting phases contained within vessels, which are linked together via connections. The process modelling language is based upon the declaration of keywords and is structured closely around the conceptual framework. It allows the formal declaration of both the basic conceptual elements that comprise the physical representation and also any additional information required to complete the process description, such as geometrical attributes and modelling assumptions.

In chapter 4 a model building procedure was given which permits the automatic generation of mathematical models of process systems from purely physical descriptions. The procedure is based on the rigorous theoretical approach to process modelling pre-

sented in chapter 2 and involves the systematic application of model generation algorithms to the physical description of a process system declared in terms of the process representation and modelling language defined in chapter 3.

Details of the implementation of the process representation and model building procedure in the computer program gMaGiC (*g*eneral *M*odel *G*eneration *C*ode) were presented in chapter 5. The implementation is based around the architecture of a software compiler and consists of three main sections: the translator, the model generation engine and the code generator. The translator takes the input specification, declared in terms of the process modelling language defined in chapter 3, and checks for syntactical and semantic correctness. If the translation is successful the model generation engine constructs an appropriate mathematical model by applying the model building algorithms presented in chapter 4. Finally, the code generator transforms the model stored internally into the format of a specific target language. In the current version of gMaGiC the output is produced in the format of the *gPROMS* modelling language [Barton and Pantelides, 1994; Oh and Pantelides, 1996].

Finally, in chapter 6, several case studies were presented which demonstrate and validate the modelling approach. The examples show the success of gMaGiC in modelling systems exhibiting a variety of physico-chemical behaviour. They include systems described by more traditional process models and also those where it is necessary to use more detailed models which consider the underlying fluid mechanics to accurately capture the physical behaviour.

7.2 Future Work

The principal limitation of the model generation approach developed in this thesis is that it is restricted to single phase systems. The theoretical difficulties associated with multiphase modelling mean that the formulation of general predictive multiphase models (even manually) is not possible at present. A possible way forward in terms of the model generation approach developed in this thesis could be to adopt one of the more standard two-fluid models (perhaps restricted to spatial variation in a single dimension) and invoke the necessary closure relationships for each particular system under consideration from predefined libraries. However, even then such an approach would be fairly restrictive in terms of the class of systems it could describe and the development of appropriate libraries of closure laws would not be an easy task. Indeed, it could be argued that if a user had enough modelling experience and physical knowledge to be able to construct libraries of closure laws, then it is unlikely that they would require a software tool for generating the multiphase conservation equations themselves. At present multiphase modelling is still an area of active research and hence the automatic generation of general predictive multiphase models remains as a challenge.

The other main limitation of this work is that it is employs only simple geometrical descriptions (single regular cylinders and annuli). The inclusion of more complex geometrical information would require facilities for detailed surface modelling of each vessel to permit accurate imposition of boundary conditions and the spatial subdivision of the vessel volume (mesh generation) required for numerical solution of the generated model and also for verification of the interconnections between vessels (for example, checking that connected ports are of the same size, shape and orientation). This is clearly impractical using the present text based approach and instead would require the development of a computer-based graphics package or computer-aided design tool, or interfacing to an existing system. The question also arises as to how to represent the model equations and boundary conditions in irregular geometries. For regular geometries the selection of an appropriate coordinate system with which to span the control volume is usually straightforward but in more complex systems it may be difficult to represent the equations in a form which is convenient both for inspection and numerical solution. This is highlighted by the approaches used in CFD tools where the user does not have access to the basic underlying equations which are actually embedded within the numerical methods that are used for their solution. At present, *gPROMS* [Barton and Pantelides, 1994; Oh and Pantelides, 1996] is the only commercial process modelling environment with capabilities for the representation and solution of distributed parameter process models and this is limited to rectangular domains.

The flow examples given in chapter 6 indicate that standard numerical methods and modelling environments cannot always deal with the generated models, which sometimes require further manipulation, reformulation and/or partial analytical simplification. The gap between the formulation of rigorous mathematical models and models suitable for numerical solution could be a possible direction for future research.

There are other areas of the approach, particularly the implementation, which could be improved with further work. From a practical point of view the inclusion of facilities for (symbolic) numerical analysis of the generated models, such as the identification of sets of free variables and differential index checking, would be useful extensions. Likewise, the development of external libraries for defining chemical reactions would be a useful addition and a similar approach might be possible for including different turbulence models. Finally, though not a particular focus of the work, a more user-friendly implementation would perhaps improve the accessibility of the software.

References

Aho, A. V. and J. D. Ullman [1977]. *Principles of Compiler Design.* Addison-Wesley Publishing Company.

Anderson, D. A., J. C. Tannehill, and R. H. Pletcher [1984]. *Computational Fluid Mechanics and Heat Transfer.* Hemisphere Publishing Corporation.

Andersson, M. [1993]. *OMSIM and OMOLA Tutorial and User's Manual.* Technical Report TFRT-7504: Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

AspenTech [1988]. *ASPEN PLUS User Guide.* Cambridge, MA.

Banerjee, S. [1997]. Multifluid models for multiphase flows. Lecture given at Imperial College London.

Barber, J. A., J. D. Perkins, and R. W. H. Sargent [1998]. Boundary Conditions for Flow with Dispersion. *Chemical Engineering Science* **53**(7), pp. 1463 – 1464.

Barton, P. I. [1992]. *The Modelling and Simulation of Combined Discrete/Continuous Processes.* PhD thesis, University of London.

Barton, P. I. and C. C. Pantelides [1994]. Modelling of combined discrete/continuous processes. *AIChE J.* **40**, pp. 966 – 979.

Basta, N. [1997]. Process Simulation - Having It Your Way. *Chemical Engineering* **104**(4), pp. 155 – 160.

Bieszczad, J., A. Koulouris, and G. Stephanopoulos [1999]. Model.la: A Phenomena-based Modeling Environment for Computer-aided Process Design. In *Fifth International Conference on Foundations of Compter-Aided Process Design FOCAPD '99*, Colorado, USA. Paper C03.

Bird, R. B. [1957]. The equations of change and the macroscopic mass, momentum and energy balances. *Chemical Engineering Science* **6**, pp. 123 – 131.

Bird, R. B., W. E. Stewart, and E. N. Lightfoot [1960]. *Transport Phenomena.* John Wiley & Sons.

BOGUSCH, R., B. LOHMANN, AND W. MARQUARDT [1996]. Computer-Aided Process Modeling With MODKIT. In *Proc. Chemputers Europe III*, Frankfurt, Germany.

BOGUSCH, R. AND W. MARQUARDT [1995]. A formal representation of model equations. *Computers chem. Engng.* **19**, pp. S211 – S216. Suppl.

BOURÉ, J. A. [1987]. Two-phase flow models: The closure issue. In G. F. Hewitt, J. M. Delhaye, and N. Zuber (Eds.), *Multiphase Science and Technology*, Vol. 3, Chapter 1, pp. 3 – 30. USA. Hemisphere Publishing Corporation.

BOURÉ, J. A. AND J. M. DELHAYE [1982]. General Equations and Two-Phase Flow Modelling. In G. Hestroni (Ed.), *Handbook of Multiphase Systems*, pp. [1–36] – [1–95]. Hemisphere Publishing Corporation.

CHO, J. H. [1997]. *Computer Aids for Mathematical Model Building.* PhD thesis, University of London.

COBDEN, A. L. [2000]. *Integrated Process Design and Control.* PhD thesis, University of London. (in preparation).

COULSON, J. M. AND J. F. RICHARDSON [1990]. *Chemical Engineering (Fluid Flow, Heat Transfer & Mass Transfer)*, Vol. 1. Pergamon Press.

DANCKWERTS, P. V. [1953]. Continuous flow systems. *Chemical Engineering Science* **2**(1), pp. 1 – 13.

DANESHYAR, H. [1976]. *One-Dimensional Compressible Flow.* Pergamon Press.

DEKKERS, R., H. A. PREISIG, AND M. WEISS [1999]. SyMoM - a symbolic manipulation package for dynamical models of physical systems. *Computers chem. Engng.* **23**, pp. S657 – S660. Suppl.

DIETERICH, E. E. AND G. EIGENBERGER [1995]. Bimap - a tool for computer aided modeling in chemical reaction engineering. *Computers chem. Engng.* **19**, pp. S773 – S778. Suppl.

DIETERICH, E. E. AND G. EIGENBERGER [1997]. The ModuSim concept for modular modeling and simulation in Chemical Engineering. *Computers chem. Engng.* **21**, pp. S805 – S809. Suppl.

DIETERICH, E. E., A. SALDEN, J. SCHÄFER, J. SCHMIDT, AND G. EIGENBERGER [1997]. Computer aided modeling with BIMAP. *Computers chem. Engng.* **21**(10), pp. 1191 – 1201.

DOMBROWSKI, N., E. A. FOUMENY, S. OOKAWARA, AND A. RIZA [1993]. The Influence of Reynolds Number on the Entry Length and Pressure Drop for Laminar Pipe Flow. *The Canadian Journal of Chemical Engineering* **71**, pp. 472 – 476.

DRENGSTIG, T., S. O. WASBØ, AND B. A. FOSS [1997]. A formal graphical based process modeling methodology. *Computers chem. Engng* **21**, pp. S835 – S840. Suppl.

DREW, D. A. AND G. B. WALLIS [1994]. Fundamentals of Two-Phase Flow Modelling. In G. F. Hewitt, J. M. Delhaye, N. Zuber, J. H. Kim, and R. T. Lahey (Eds.), *Multiphase Science and Technology*, Vol. 8, Chapter 1, pp. 1 – 67. USA. Hemisphere Publishing Corporation.

FELL, N. [1997]. Simulation almost as good as the real thing. *The Chemical Engineer* (637), pp. 15 – 18.

FLETCHER, C. A. J. [1988]. *Computational Techniques for Fluid Dynamics - Specific Techniques for Different Flow Categories*, Vol. 2. USA. Springer-Verlag.

FLETCHER, C. A. J. [1991]. *Computational Techniques for Fluid Dynamics - Fundamentals and General Techniques* (2nd ed.), Vol. 1. Springer-Verlag.

FROMENT, G. AND K. BISCHOFF [1990]. *Chemical Reactor Analysis and Design* (2nd ed.). John Wiley & Sons, Inc.

GERSTLAUER, A., M. HIERLEMANN, AND W. MARQUARDT [1993]. *On the representation of balance equations in a knowledge-based process modelling tool.* Technical report, RWTH Aachen, Aachen, Germany. TR1993-03.

GOSMAN, A. D. AND R. I. ISSA [1996]. Computational Fluid Dynamics. Lecture course given at Imperial College London.

HANGOS, K. M. AND I. T. CAMERON [1997]. The Formal Representation of Process System Modelling Assumptions and their Implications. *Computers chem. Engng.* **21**, pp. S823 – S828. Suppl.

HEWITT, G. F. [1997]. [untitled]. Private Communication.

HEYDWEILLER, J. C., R. F. SINCOVEC, AND L. T. FAN [1977]. Dynamic simulation of chemical processes described by distributed and lumped parameter models. *Computers chem. Engng.* **1**, pp. 125 – 131.

HIMMELBLAU, D. M. AND K. B. BISCHOFF [1968]. *Process Analysis and Simulation: Deterministic Systems.* USA. John Wiley & Sons Inc.

HINZE, J. O. [1975]. *Turbulence* (2nd ed.). New York. McGraw-Hill.

ISHII, M. [1975]. *Thermo-fluid Dynamic Theory of Two-Phase Flow.* Eyrolles.

ISHII, M. [1990]. Two fluid model for two-phase flow. In G. F. Hewitt, J. M. Delhaye, and N. Zuber (Eds.), *Multiphase Science and Technology*, Vol. 5, Chapter 1, pp. 1 – 63. USA. Hemisphere Publishing Corporation.

JARVIS, R. B. [1993]. *Robust Dynamic Simulation of Chemical Engineering Processes.* PhD thesis, University of London.

JENSEN, A. K. AND R. GANI [1996]. A computer aided system for generation of problem specific process models. *Computers chem. Engng.* **20**, pp. S145 – S150. Suppl.

JENSEN, A. K. AND R. GANI [1999]. A Computer Aided Modeling System. *Computers chem. Engng.* **23**, pp. S673 – S678. Suppl.

JONES, W. P. [1996]. Turbulence modelling for variable density and combusting flows. In *Combustion and Turbulence in Two-Phase Flows*, Lecture Series 1996-02. von Karman Institute for Fluid Mechanics.

KERN, D. Q. [1950]. *Process Heat Transfer.* McGraw-Hill.

KREYSZIG, E. [1993]. *Advanced Engineering Mathematics.* John Wiley & Sons.

KRÖNER, A., P. HOLL, W. MARQUARDT, AND E. D. GILLES [1990]. DIVA - an open architecture for dynamic simulation. *Computers chem. Engng.* **14**(11), pp. 1289 – 1295.

KRUSE, R. L., B. P. LEUNG, AND C. L. TONDO [1991]. *Data Structures and Program Design in C.* Prentice-Hall International Editions.

LAHEY, R. T. [1998]. A CFD Analysis of Multidimensional Two-Phase Flow and Heat Transfer using a Four Field Model. Seminar given at Imperial College London.

LAKNER, R., I. T. CAMERON, AND K. M. HANGOS [1999]. An Assumption-Driven Case-Specific Model Editor. *Computers chem. Engng.* **23**, pp. S695 – S698. Suppl.

LAUNDER, B. E. AND D. B. SPALDING [1972]. *Mathematical Models of Turbulence.* London Academic Press.

LECOMTE, M. [1998]. *Modeling of a Safety Relief System.* Technical report, Centre for Process Systems Engineering, Imperial College, London, UK.

LEVINE, J. R., T. MASON, AND D. BROWN [1992]. *lex & yacc.* O'Reilly and Associates, Inc.

LUYBEN, M. L. AND B. D. TYREUS [1998]. An industrial design/control study for the vinyl acetate monomer process. *Computers chem. Engng.* **22**(7/8), pp. 867 – 877.

LUYBEN, W. L. [1990]. *Process modeling, simulation and control for chemical engineers.* McGraw-Hill.

MARQUARDT, W. [1991]. Dynamic process simulation - recent progress and future challenges. In R. Arkun and W. H. Ray (Eds.), *Chemical Process Control - CPCIV.* CACHE Publications.

MARQUARDT, W. [1994]. Computer-aided generation of chemical engineering process models. *International Chemical Engineering* **34**(1), pp. 28 – 46.

MARQUARDT, W. [1996]. Trends in computer-aided process modeling. *Computers chem. Engng.* **20**(6/7), pp. 591 – 609.

MATTSSON, S. AND M. ANDERSSON [1992]. Omola - an object-oriented modelling language. In M. Jamshidi and C. J. Herget (Eds.), *Recent Advances in Computer-Aided Control Systems Engineering*, pp. 291 – 310. Elsevier Science Publishers B.V.

MURALIDHAR, K. AND T. SUNDARARAJAN [1995]. *Computational Fluid Flow and Heat Transfer.* New Delhi. Narosa Publishing House. IIT Kanpur Series of Advanced Texts.

OH, M. [1995]. *Modelling and Simulation of Combined Lumped and Distributed Parameter Processes.* PhD thesis, University of London.

OH, M. AND C. C. PANTELIDES [1996]. A modelling and simulation language for combined lumped and distributed parameter systems. *Computers chem. Engng.* **20**(6/7), pp. 611 – 633.

PANTELIDES, C. C. [1995]. Modelling, simulation and optimization of hybrid processes. In *Paper presented at Workshop on 'Analysis and Design of Event-Driven Operations in Process Systems'.* London.

PANTELIDES, C. C. [1998]. *Dynamic Behaviour of Process Systems.* Centre for Process Systems Engineering, Imperial College London. Lecture notes.

PANTELIDES, C. C. AND P. I. BARTON [1993]. Equation-oriented dynamic simulation: current status and future perspectives. *Computers chem. Engng.* **17S**, pp. 263 – 285.

PANTELIDES, C. C. AND H. I. BRITT [1995]. Multipurpose process modelling environments. In L. T. Biegler and M. F. Doherty (Eds.), *Proc. Conf. on Foundations of Computer-Aided Process Design '94.* CACHE Publications.

PANTELIDES, C. C., D. GRITSIS, K. R. MORRISON, AND R. W. H. SARGENT [1988]. The mathematical modelling of transient systems using differential-algebraic equations. *Computers chem. Engng.* **12**(5), pp. 449 – 454.

PATANKAR, S. V. AND D. B. SPALDING [1970]. *Heat and Mass Transfer in Boundary Layers. A general calculation procedure* (2nd ed.). Intertext Books, London.

PERKINS, J. D. AND G. W. BARTON [1987]. Modelling and simulation in process operation. In G. V. Reklaitis and H. D. Spriggs (Eds.), *Computer Aided Process Operations*, pp. 287 – 316.

PERKINS, J. D. AND R. W. H. SARGENT [1982]. SPEEDUP: A computer program for steady-state and dynamic simulation and design of chemical processes. *AIChE Symp. Ser.* **78**(214), pp. 1 – 11.

PERKINS, J. D., R. W. H. SARGENT, R. VÁZQUEZ-ROMÁN, AND J. H. CHO [1996]. Computer Generation of Process Models. *Computers chem. Engng.* **20**(6/7), pp. 635 – 639.

PIÈLA, P. C., T. G. EPPERLY, K. M. WESTERBERG, AND A. W. WESTERBERG [1991]. ASCEND: An object-oriented computer environment for modeling and analysis: The modeling language. *Computers chem. Engng.* **15**, pp. 53 – 72.

PREISIG, H. A. [1995]. MODELLER - an object-oriented computer-aided modelling tool. *AIChE Symp. Ser.* **91**(304), pp. 328 – 331.

PREISIG, H. A. [1996]. Computer-aided modelling - two paradigms on control. *Computers chem. Engng.* **20**, pp. S981 – S986. Suppl.

PREISIG, H. A., T. Y. LEE, AND F. LITTLE [1990]. A prototype computer-aided modelling tool for life-support system models. In *20th Intersociety Conference on Environmental Systems*, Williamsburg, Virginia. SAE Technical Paper Series No. 901269.

PREISIG, H. A., T. Y. LEE, F. LITTLE, AND B. WRIGHT [1989]. On the representation of life-support system models. SAE Technical Paper Series No.891479.

REID, R. C., J. M. PRAUSNITZ, AND B. E. POLING [1987]. *The Properties of Gases & Liquids* (4th ed.). McGraw-Hill.

RICHARDSON, S. M. [1989]. *Fluid Mechanics*. Hemisphere Publishing Corporation.

ROBERTS, I. F. [1997]. Conservation equations, two-phase. In G. F. Hewitt, G. L. Shires, and Y. V. Polezhaev (Eds.), *International Encyclopedia of Heat & Mass Transfer*, pp. 223 – 230. CRC Press.

ROGERS, G. F. C. AND Y. R. MAYHEW [1992]. *Engineering Thermodynamics: work and heat transfer* (4th ed.). Longman Scientific & Technical.

RUBESIN, M. W. AND W. C. ROSE [1973]. *The Turbulent Mean-Flow, Reynolds-Stress and Heat-Flux Equations in Mass-Averaged Dependent Variables.* Technical report. NASA, Ames Research Center, Calif., USA. NASA TM X-62248.

SARGENT, R. W. H. [1983]. Advances in modelling and analysis of chemical process systems. *Computers chem. Engng.* **7**(4), pp. 219 – 237.

SCHIESSER, W. E. [1991]. *The Numerical Method of Lines: Integration of Partial Differential Equations.* Academic Press, Inc.

SCHLICHTING, H. [1968]. *Boundary-layer Theory* (6th ed.). McGraw-Hill.

SEXTON, C. [1996]. *Newnes C Pocket Book* (2nd ed.). B. H. Newnes.

SINCLAIR, J. L. [1999]. CFD for Multiphase Flow: Research Codes and Commercial Software. In *Fifth International Conference on Foundations of Computer-Aided Process Design FOCAPD '99*, Colorado, USA. Paper I13.

SOWA, C. J. [1997]. Make the Move to Dynamic Simulation. *Chemical Engineering Progress* **93**(8), pp. 44 – 47.

SPALDING, D. B. AND G. F. HEWITT [1992]. Computer code descriptions. In G. F. Hewitt, J. M. Delhaye, and N. Zuber (Eds.), *Multiphase Science and Technology*, Vol. 6, Chapter 4, pp. 721 – 808. USA. Hemisphere Publishing Corporation.

SPEZIALE, C. G. [1991]. Analytical Methods for the Development of Reynolds-stress Closures in Turbulence. *Annu. Rev. Fluid Mech.*. **23**, pp. 107 – 157.

STANDART, G. [1964]. The mass, momentum and energy equations for heterogeneous flow systems. *Chemical Engineering Science* **19**, pp. 227 – 236.

STEPHANOPOULOS, G. AND C. HAN [1996]. Intelligent systems in process engineering: a review. *Computers chem. Engng.* **20**(6/7), pp. 743 – 791.

STEPHANOPOULOS, G., G. HENNING, AND H. LEONE [1990]. Model.la. A Modeling Language for Process Engineering. *Computers chem. Engng.* **14**(8), pp. 813 – 869.

STEPHANOPOULOS, G., J. JOHNSTON, T. KRITICOS, R. LAKSHMANAN, M. MAVROVOUNIOTIS, AND C. SILETTI [1987]. DESIGN-KIT: An Object- oriented Environment for Process Engineering. *Computers chem. Engng.* **11**(6), pp. 655 – 674.

TELNES, K. [1992]. *Computer Aided Modeling of Dynamic Processes based on Elementary Physics.* PhD thesis, University of Trondheim, The Norwegian Institute of Technology.

TRÄNKLE, F., A. GERSTLAUER, M. ZEITZ, AND E. D. GILLES [1997]. Application of the Modeling and Simulation Environment PROMOT/DIVA to the Modeling of Distillation Processes. *Computers chem. Engng.* **21**, pp. S841 – S846. Suppl.

UNDERWOOD, A. J. V. [1933]. Graphical computation of logarithmic mean temperature difference. *Industrial Chemist* **9**, pp. 167.

VÁZQUEZ-ROMÁN, R. [1992]. *Computer Aids For Process Model Building.* PhD thesis, University of London.

VÁZQUEZ-ROMÁN, R., J. KING, AND R. BAÑARES-ALCÁNTARA [1996]. KBMoSS: A process engineering modelling support system. *Computers chem. Engng.* **20**, pp. S309 – S314. Suppl.

VERSTEEG, H. K. AND W. MALALASEKERA [1995]. *An introduction to computational fluid dynamics. The finite volume method.* Longman Scientific & Technical.

WESTERWEELE, M. R., H. A. PREISIG, AND M. WEISS [1999]. Concept and Design of Modeller, a Computer-Aided Modelling Tool. *Computers chem. Engng.* **23**, pp. S751 – S754. Suppl.

WHITAKER, S. [1968]. *Introduction to Fuid Mechanics.* Prentice-Hall, Inc.

WILCOX, D. C. [1993]. *Turbulence Modeling for CFD.* DCW Industries, Inc.

WULFF, W. [1990]. Computational methods for multiphase flow. In G. F. Hewitt, J. M. Delhaye, and N. Zuber (Eds.), *Multiphase Science and Technology*, Vol. 5, Chapter 3, pp. 85 – 238. USA. Hemisphere Publishing corporation.

YADIGAROGLU, G. [1988]. Basic Equations. In *Short Course on Modelling of Mulitphase Systems for Industrial Applications.* Zurich. Swiss Federal Institute of Technology.

Appendix A

Nomenclature

A.1 Mathematical Notation

A.1.1 Symbols

a	arbitrary constant or coefficient	–
a_i	interfacial transfer area per unit volume	L^{-1}
A	area or	L^2
	Arrhenius constant	[varies]
b	arbitrary constant or coefficient	–
B	boundary	L^2
c	arbitrary constant or coefficient or	–
	concentration (molar density)	mole/L^3
c_i	concentration (molar density) of component i	mole/L^3
C	constant	–
$\hat{C}_p$	specific heat capacity at constant pressure	L^2/t^2T
$\hat{C}_v$	specific heat capacity at constant volume	L^2/t^2T
d	arbitrary constant or coefficient	–
D	diffusion (or dispersion) coefficient or	L^2/t
	model domain (control volume)	L^3
e	arbitrary constant or coefficient	–
$\hat{E}$	specific total energy	L^2/t^2
E_A	activation energy	ML/t^2mole
f	arbitrary constant or coefficient	–
$f(x)$	arbitrary function	–
f	friction factor	–
F	frictional force	ML/t^2
$\mathbf{F}$	specific external body force	L/t^2
$\mathbf{g}$	gravitational acceleration	L/t^2
$g(x)$	arbitrary function	–
h	film heat transfer coefficient or	M/t^3T
	liquid level	L

$\hat{h}$	specific enthalpy	L^2/t^2
$\hat{H}$	total enthalpy	L^2/t^2
$h(x)$	arbitrary function	–
$\mathbf{I}$	unit tensor	–
J	mass, energy or momentum flux (section (2.5.1)	[varies]
J_i and $\mathbf{J}_i$	mass flux of component i (scalar and vector)	M/L^2t
$\mathbf{J}_k$	as defined in text	[varies]
k	kinetic energy of turbulence or	L^2/t^2
	thermal conductivity or	ML/t^3T
	reaction rate constant	[varies]
l_m	mixing length	L
L	length	L
m	interphase mass flux per unit interfacial area	
	per unit volume	M/L^2t
M	mass	M
Ma	Mach number	–
$\mathbf{M}$	interphase momentum transfer per unit volume	M/L^2t^2
$\mathbf{M}_{ik}$	interfacial force	M/L^2t^2
M_w	molecular weight	M/mole
n	number of components	–
$\mathbf{n}$	unit normal vector	L
N_i	total molar flux of component i	mole/L^2t
Nu	Nusselt number	–
p	pressure	M/Lt^2
P	perimeter	L
$\mathbf{P}$	momentum	ML/t
Pr	Prandtl number	–
q_{ki}	interfacial energy flux	M/t^2
$\mathbf{q}$	heat flux	M/t^3
Q	net energy added to system across closed surfaces	ML^2/t^3
r	spatial coordinate or	L
	rate of production by individual chemical reaction(s)	[varies]
R	radius or	L
	universal gas constant	ML^2/t^2Tmole
R_i	net rate of production of component i by chemical reaction	[varies]
Re	fluid Reynolds number	–
S or $\mathbf{S}$	surface (scalar and vector)	L^2

S_ϕ	scalar source term	[varies]
t	time	t
T	temperature	T
$\mathbf{T}$	total stress tensor $= -p\mathbf{I} + \tau$	M/Lt^2
u	dependent variable or	–
	speed of sound	L/t
$\hat{U}$	specific internal energy	L^2/t^2
v	(scalar) component of velocity vector	L/t
$\mathbf{v}$	velocity vector	L/t
V	volume	L^3
$\hat{V}$	specific volume	L^3/M
w and $\mathbf{w}$	mass flow rate (scalar and vector)	M/t
W	net work done by system	ML^2/t^3
$\mathcal{W}$	molar flow rate	mole$/t$
x	any variable (scalar or vector)	[varies]
x_i	mass fraction of component i	–
$\mathbf{x}$	mass fraction vector	–
y_i	mole fraction of component i	–
z	spatial coordinate	L

A.1.2 Greek Symbols

γ	ratio of specific heat capacities $\hat{C}_p/\hat{C}_v$ or	–
	specific gravity	–
Γ	scalar flux proportionality constant	[varies]
Γ_k	interphase mass transfer per unit volume	M/L^3t
δ_{ij}	Kronecker delta	–
ϵ	local void fraction	–
θ	spatial coordinate or	–
	temperature difference	T
Λ	interphase energy transfer per unit volume	M/Lt^3
μ	viscosity	M/Lt
π	constant $= 3.14159...$	–
ρ	mass density	M/L^3
τ	deviatoric stress tensor	M/Lt^2
Υ	total interphase mass transfer	M/t
ϕ	any variable (scalar)	[varies]
$\hat{\Phi}$	specific potential energy	L^2/t^2
χ_k	as defined in text	[varies]

ψ_k	as defined in text	[varies]

A.1.3 Subscripts/Superscripts

0	at 0
A	phase A
	area
B	phase B or
	boundary
e	effective
i	interface or component i
in	inlet
k	k-th phase or plane
i, j, k	indices for index notation
l	lower limit
m	mean
max	maximum
n	normal
out	outlet
r	r coordinate
s or S	surface
t	turbulent
u	upper limit
V	volume
w	wall
x	cross section
z	z coordinate

A.1.4 Miscellaneous

$\overline{x}$	time or ensemble average of x
x'	fluctuating component of x
$\tilde{x}$	mass-average of x
x''	mass-averaged fluctuating component of x
$\langle x \rangle$	spatial average of x
$\hat{x}$	specific value of x
$\overline{\overline{x}}$	double time or ensemble average of x or
	phase mean of x
$\overline{x'}$	time or ensemble average of fluctuating component of x
$\overline{\tilde{x}}$	time or ensemble average of mass-average of x

$\overline{x''}$ time or ensemble average of mass-averaged fluctuating component of x

A.2 Notation used in gMaGiC

In the current version of gMaGiC the generated mathematical models are produced in the format of the *gPROMS* simulation language [Barton and Pantelides, 1994; Oh and Pantelides, 1996] for direct use in the *gPROMS* modelling environment (see chapter 5). Consequently, the model variables are labelled, typed and referenced as shown below in accordance with *gPROMS* rules and conventions.

A.2.1 Variable Labelling, Typing and Referencing

The name of each model variable or parameter is structured into two parts separated by an underscore: a variable type and a specific identifier taken from the process description

`VariableType_`*identifier*

The first part of the variable name simply identifies the physical type of the variable and uses the symbols given below (section A.2.2). The second part identifies a specific instance of that variable associated with a specific phase, port or vessel using identifiers specified in the process modelling language declaration.

Following the convention used in *gPROMS*, variables within models are then referenced globally using the pathname convention

`ModelName.VariableType_`*identifier*

where `ModelName` is the name given to the individual reservoir, vessel or (overall) connection submodels.

A.2.2 Variable Types

`k`	kinetic energy of turbulence	k
`g`	component of gravitational acceleration	g
`qR`	effective heat flux component	q_r^e
`qZ`	effective heat flux component	q_z^e
`ALPHA`	heat transfer coefficient	h
`AREA`	area	A
`CONC`	total molar density	c
`CONC_COMP`	molar density of component i	c_i
`COND`	thermal conductivity	k
`CONDeff`	effective thermal conductivity	k_e
`CONDturb`	turbulent thermal conductivity	k_t

DISP	dispersion coefficient	D
DISPeff	effective dispersion coefficient	D_e
DISPturb	turbulent dispersion coefficient	D_t
ENTH	specific enthalpy	$\hat{h}$
E	specific total energy	$\hat{E}$
F	frictional force at wall	F
J_COMPR	effective mass flux component	J^e_{ir}
J_COMPZ	effective mass flux component	J^e_{iz}
MW	molecular weight	M_w
NoCOMP	number of components	n
P	perimeter	P
PRESS	pressure	p
Q	heat flux at wall	Q
REACT_COMP	net rate of production of component i by chemical reaction	R_i
RHO	mass density	ρ
RHO_COMP	mass density of component i	ρ_i
TAURR	shear stress component	τ_{rr}
TAURReff	effective shear stress component	τ^e_{rr}
TAURRturb	turbulent shear stress component	τ^t_{rr}
TAURZ	shear stress component	τ_{rz}
TAURZff	effective shear stress component	τ^e_{rz}
TAURZturb	turbulent shear stress component	τ^t_{rz}
TAUZZ	shear stress component	τ_{zz}
TAUZZeff	effective shear stress component	τ^e_{zz}
TAUZZturb	turbulent shear stress component	τ^t_{zz}
TAUO	shear stress at wall	τ_w
TAUOO	shear stress component	$\tau_{\theta\theta}$
TAUOOeff	effective shear stress component	$\tau^e_{\theta\theta}$
TAUOOturb	turbulent shear stress component	$\tau^t_{\theta\theta}$
TEMP	temperature	T
TOTENTH	specific total enthalpy	$\hat{H}$
U	specific internal energy	$\hat{U}$
VEL	velocity	v
VELR	velocity component	v_r
VELZ	velocity component	v_z
VISC	molecular viscosity	μ
VISCturb	turbulent (eddy) viscosity	μ_t
VOL	volume	V

X_COMP	mass fraction of component i	x_i
XAREA	cross sectional area	A
Y_COMP	mole fraction of component i	y_i

Appendix B

Averaging Operators

B.1 Basic Definitions

The standard averaging operations are self-explanatory and are given below.

Time Average over interval Δt

$$\overline{x} = \frac{1}{\Delta t} \int_{\Delta t} x dt \tag{B.1}$$

Ensemble Average over N (experimental) realisations

$$\overline{x} = \lim_{N \to \infty} \left[\frac{1}{N} \sum_{i=1}^{N} x|_i \right] \tag{B.2}$$

Spatial Average over Ω

$$\langle x \rangle = \frac{1}{\Omega} \int_{\Omega} x d\Omega \tag{B.3}$$

B.2 Special Definitions for Multiphase Modelling

The derivation of a time averaged multiphase model involves the definition of an averaging time interval Δt, assumed to be large enough to 'smooth out' the local instantaneous fluctuations of the fluid properties yet small compared with the macroscopic time constant of the unsteadiness of the bulk flow [Ishii, 1975]. For any point in the flow domain, this time interval is composed of the time occupied by interfaces $[\Delta t]_S$ and the time occupied by phases $[\Delta t]_T = \sum_k [\Delta t]_k$, so that:

$$\Delta t = [\Delta t]_S + [\Delta t]_T \tag{B.4}$$

A phase density function X_k is then defined as:

$X_k = 1$ if phase k is present at the point
$X_k = 0$ if not

Based on these definitions, the following quantities and averages are defined:

Local Void Fraction ϵ_k

$$\epsilon_k = \frac{1}{\Delta t}\int_{[\Delta t]_T} X_k dt = \frac{\Delta t_k}{\Delta t} \tag{B.5}$$

k-th Phase Average

$$\overline{x}_k = \frac{1}{\Delta t}\int_{[\Delta t]_k} x_k dt \tag{B.6}$$

k-th Phase Mean

$$\overline{\overline{x}}_k = \frac{\overline{x_k}}{\epsilon_k} \tag{B.7}$$

k-th Phase Mass-Weighted Average

$$\tilde{x}_k = \frac{\overline{\rho_k x_k}}{\overline{\rho_k}} \tag{B.8}$$

Further details on the multiphase averaging operators defined above are given by Ishii [1975; 1990].

Appendix C

Flex Input Specification

```
/* definitions section */

%{
#include "parser.tab.h"
#include "global.h"

int yylineno = 1;
%}

comment         "#".*
endofline       [\n]
delimiter       [ \t]
whitespace      {delimiter}+
letter          [A-Za-z]
underscore      "_"
string          {letter}({underscore})*
digit           [0-9]
number          ({digit}+|({digit}*\.{digit}+)([eE][-+]?{digit}+)?)
id              {string}({string}|{digit})*
arrow           "->"

%%

/* rules section */

{comment}       { }
{whitespace}    { }
{endofline}     {yylineno++;}

1-D                     {return ONE_D;}
2-D                     {return TWO_D;}
3-D                     {return THREE_D;}
ANGULAR                 {return ANGULAR;}
ASSUMPTIONS             {return ASSUMPTIONS;}
AXIAL                   {return AXIAL;}
CLOSED                  {return CLOSED;}
CONNECTIONS             {return CONNECTIONS;}
CYLINDER                {return CYLINDER;}
DOWN                    {return DOWN;}
FRICTIONLESS            {return FRICTIONLESS;}
GEOMETRY                {return GEOMETRY;}
```

```
INCOMPRESSIBLE          {return INCOMPRESSIBLE;}
ISOTHERMAL              {return ISOTHERMAL;}
LAMINAR                 {return LAMINAR;}
LEFT                    {return LEFT;}
MACROSCOPIC             {return MACROSCOPIC;}
NO_SHAFT_WORK           {return NO_SHAFT_WORK;}
OPEN                    {return OPEN;}
PLUG_FLOW               {return PLUG_FLOW;}
PORTS                   {return PORTS;}
PHASES                  {return PHASES;}
QUASI_STEADY_STATE      {return QUASI_STEADY_STATE;}
RADIAL                  {return RADIAL;}
RESERVOIRS              {return RESERVOIRS;}
RIGHT                   {return RIGHT;}
STEADY_STATE            {return STEADY_STATE;}
UP                      {return UP;}
VESSEL                  {return VESSEL;}
WELL_MIXED              {return WELL_MIXED;}

{number}        {yylval.dval = atof(yytext);
                 return NUMBER;}

{id}            {yylval.sval = (char *) strdup(yytext);
                 return IDENTIFIER;}

{arrow}         {return ARROW;}
{underscore}    {printf("Error line %d: leading or lone underscore\n",
                        yylineno);
                 printf("***ABORTED***\n");
                 exit(0);}

\;              {return ';';}
\:              {return ':';}
\[              {return '[';}
\]              {return ']';}
\,              {return ',';}
\.              {return '.';}
\=              {return '=';}

.               {printf("Warning line %d: `%s' unrecognised\n",
                        yylineno, yytext);
                 printf("***ABORTED***\n");
                 exit(0);}
%%

/* user subroutines section */

yywrap()
{
  return(1);
}
```

```
yyerror(char *s)
{
  fprintf(stdout, "Error line %d: %s at or near '%s'\n",
          yylineno, s, yytext);
  printf("***ABORTED***\n");
  exit(0);
}
```

Appendix D

Bison Input Specification

```
/* definition section */

%{
#include "global.h"
%}

%union {
  int ival;
  double dval;
  char *sval;
  struct PhaseSect *phaseSect;
  struct VesselGeometry *geometrySect;
  struct PortSect *portSect;
  struct PortLocation *portLocationSect;
  struct DomainSpec *domainData;
  struct AssumptionSect *assumptionSect;
  struct ComponentSect *componentSect;
}

%token          VESSEL
%token          RESERVOIRS
%token          CONNECTIONS

%token <ival>   MACROSCOPIC
%token <ival>   ONE_D
%token <ival>   TWO_D
%token <ival>   THREE_D

%token          GEOMETRY
%token          AXIAL
%token          RADIAL
%token          ANGULAR
%token          CYLINDER
%token <ival> UP
%token <ival> DOWN
%token <ival> LEFT
%token <ival> RIGHT

%token          PHASES

%token          PORTS
```

```
%token <ival> OPEN
%token <ival> CLOSED

%token        ASSUMPTIONS
%token <ival> ISOTHERMAL
%token <ival> INCOMPRESSIBLE
%token <ival> WELL_MIXED
%token <ival> STEADY_STATE
%token <ival> PLUG_FLOW
%token <ival> QUASI_STEADY_STATE
%token <ival> NO_SHAFT_WORK
%token <ival> FRICTIONLESS
%token <ival> LAMINAR

%token <sval> IDENTIFIER
%token <dval> NUMBER
%token        ';'
%token        ':'
%token        '['
%token        ']'
%token        ','
%token        '.'
%token        '='
%token        ARROW

%type<ival>              yyVesselType
%type<phaseSect>         yyPhaseBlock
%type<phaseSect>         yyMultiplePhase
%type<phaseSect>         yySinglePhase

%type<componentSect>     yyComponentBlock
%type<componentSect>     yyMultipleComponents
%type<componentSect>     yySingleComponent

%type<geometrySect>      yyGeometryBlock
%type<geometrySect>      yyGeometrySection
%type<geometrySect>      yyVesselSize

%type<portSect>          yyPortBlock
%type<portSect>          yyMultiplePort
%type<portSect>          yySinglePort
%type<ival>              yyPortType
%type<portLocationSect>  yyPortLocation
%type<domainData>        yyAxialSpec
%type<domainData>        yyRadialSpec
%type<domainData>        yyAngularSpec

%type<assumptionSect>    yyAssumptionBlock
%type<assumptionSect>    yyMultipleAssumptions
%type<assumptionSect>    yySingleAssumption
%type<ival>              yyAssumptionName
%type<ival>              yyOrientation
```

```
%%

/* rules section */

/* GRAMMAR DEFINITION */

/* A Complete Input */

yyModelBlock : yyVesselBlock
               yyConnectionBlock
             | yyVesselBlock
               yyReservoirBlock
               yyConnectionBlock
             ;

/* Multiple Vessels */

yyVesselBlock : ySingleVessel
              | yyVesselBlock ySingleVessel
              ;

/* Single Vessel */

ySingleVessel : VESSEL ':' IDENTIFIER ':'
                yyVesselType
                yyGeometryBlock
                yyPhaseBlock
                yyPortBlock
                yyAssumptionBlock
              ;

/* Vessel Types */

yyVesselType : MACROSCOPIC  ';'
             | ONE_D ';'
             | TWO_D ';'
             | THREE_D ';'
             ;

/* Phase Section */

yyPhaseBlock : PHASES ':' yyMultiplePhase
             ;

/* Multiple Phases */

yyMultiplePhase : ySinglePhase
                | yyMultiplePhase ySinglePhase
                ;

/* Single Phase */
```

```
ySinglePhase : IDENTIFIER ':' '[' yyComponentBlock ']' ';'
             | IDENTIFIER ';'
             ;

/* Component Section */

yyComponentBlock :  yyMultipleComponents
                 ;

/* Multiple Components */

yyMultipleComponents : ySingleComponent
                     | yyMultipleComponents ySingleComponent
                     ;

/* Single Component */

ySingleComponent : IDENTIFIER
                 ;

/* Vessel Geometry */

yyGeometryBlock :
                | GEOMETRY ':' yyGeometrySection
                ;

yyGeometrySection : CYLINDER '[' yyVesselSize ']' ':' yyOrientation ';'
                  ;

yyVesselSize : AXIAL '=' NUMBER ',' NUMBER ':'
               RADIAL '=' NUMBER ',' NUMBER ':'
               ANGULAR '=' NUMBER ',' NUMBER
             | AXIAL '=' NUMBER ',' NUMBER ':'
               RADIAL '=' NUMBER ',' NUMBER
             ;

yyOrientation : UP
              | DOWN
              | LEFT
              | RIGHT
              ;

/* Port Section */

yyPortBlock : PORTS ':' yyMultiplePort
            ;

/* Multiple Ports */

yyMultiplePort : ySinglePort
               | yyMultiplePort ySinglePort
```

```
                ;

/* Single Port */

ySinglePort : IDENTIFIER ':' yyPortType ';'
            | IDENTIFIER '[' yyPortLocation ']' ':' yyPortType ';'
            ;

yyPortLocation : yyAxialSpec ':'
                 yyRadialSpec ':'
                 yyAngularSpec
               | yyAxialSpec ':'
                 yyRadialSpec
               ;

yyAxialSpec : AXIAL '=' NUMBER ',' NUMBER
            | AXIAL '=' NUMBER
            ;

yyRadialSpec : RADIAL '=' NUMBER ',' NUMBER
             | RADIAL '=' NUMBER
             ;

yyAngularSpec : ANGULAR '=' NUMBER ',' NUMBER
              | ANGULAR '=' NUMBER
              ;

yyPortType : OPEN
           | CLOSED
           ;

/* Assumption Section */

yyAssumptionBlock :
                  | ASSUMPTIONS ':' yyMultipleAssumptions

                  ;

/* Multiple Assumptions */

yyMultipleAssumptions : ySingleAssumption
                      | yyMultipleAssumptions ySingleAssumption
                      ;

/* Single Assumption */

ySingleAssumption : yyAssumptionName ';'
                  | yyAssumptionName ':' IDENTIFIER ';'
                  ;

yyAssumptionName : STEADY_STATE
                 | FRICTIONLESS
```

```
        | ISOTHERMAL
        | NO_SHAFT_WORK
        | WELL_MIXED
        | INCOMPRESSIBLE
        | PLUG_FLOW
        | QUASI_STEADY_STATE
        | LAMINAR
        ;

/* Reservoir Section */

yyReservoirBlock : RESERVOIRS ':' yyMultipleReservoirs
                 ;

/* Multiple Reservoirs */

yyMultipleReservoirs : yySingleReservoir
                     | yyMultipleReservoirs yySingleReservoir
                     ;

/* Single Reservoir */

yySingleReservoir : IDENTIFIER ';'
                  ;

/* Connection Section */

yyConnectionBlock : CONNECTIONS ':' yyMultipleConnections
                  ;

/* Multiple Connections */

yyMultipleConnections : yySingleConnection
                      | yyMultipleConnections yySingleConnection
                      ;

/* Single Connection */

yySingleConnection : IDENTIFIER '.' IDENTIFIER ARROW
                     IDENTIFIER '.' IDENTIFIER ';'
                   | IDENTIFIER ARROW
                     IDENTIFIER '.' IDENTIFIER ';'
                   | IDENTIFIER '.' IDENTIFIER ARROW
                     IDENTIFIER ';'
                   ;
%%
```

Appendix E

Generated Models

Models begin overleaf.

E.1 Choking in Compressible Flow with Friction

```
DECLARE
  TYPE

        Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
        Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
        Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
        SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
        Temperature      = 300 : 0 : 1e4              UNIT = "K"
        Positive         = 10 : 0 :10e10              UNIT = "-"
        Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
        Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
        Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END


MODEL R1

PARAMETER

NoCOMP                          AS INTEGER
MW                              AS ARRAY(NoCOMP) OF REAL


VARIABLE

CONC                            AS Concentration
CONC_COMP                       AS ARRAY(NoCOMP) OF Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
RHO_COMP                        AS ARRAY(NoCOMP) OF Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity
X_COMP                          AS ARRAY(NoCOMP) OF Normalised
Y_COMP                          AS ARRAY(NoCOMP) OF Normalised


SET

NoCOMP := 2;


EQUATION

#Equations of State
#==================

#ENTH = ENTH(PRESS, TEMP, X_COMP)

#RHO = RHO(PRESS, TEMP, X_COMP)


#Constitutive Equations
#======================

RHO_COMP = CONC_COMP*MW;

RHO_COMP = RHO*X_COMP;

RHO = SIGMA(RHO_COMP);

CONC_COMP = CONC*Y_COMP;

CONC = SIGMA(CONC_COMP);

ENTH*RHO = U*RHO + PRESS;

END #model R1


MODEL R2

PARAMETER

NoCOMP                          AS INTEGER
MW                              AS ARRAY(NoCOMP) OF REAL


VARIABLE

CONC                            AS Concentration
CONC_COMP                       AS ARRAY(NoCOMP) OF Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
RHO_COMP                        AS ARRAY(NoCOMP) OF Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity
X_COMP                          AS ARRAY(NoCOMP) OF Normalised
Y_COMP                          AS ARRAY(NoCOMP) OF Normalised


SET

NoCOMP := 2;


EQUATION

#Equations of State
#==================

#ENTH = ENTH(PRESS, TEMP, X_COMP)

#RHO = RHO(PRESS, TEMP, X_COMP)


#Constitutive Equations
#======================

RHO_COMP = CONC_COMP*MW;

RHO_COMP = RHO*X_COMP;

RHO = SIGMA(RHO_COMP);

CONC_COMP = CONC*Y_COMP;

CONC = SIGMA(CONC_COMP);

ENTH*RHO = U*RHO + PRESS;

END #model R2


MODEL pipe

PARAMETER

NoCOMP                          AS INTEGER
XAREA_pipe                      AS REAL
P_pipe                          AS REAL
```

```
MW                              AS ARRAY(NoCOMP) OF REAL


DISTRIBUTION_DOMAIN

AXIALpipe                       AS (0 : 0.5)


VARIABLE

CONC_COMP_gas                   AS DISTRIBUTION(NoCOMP, AXIALpipe) OF Concentration
CONC_gas                        AS DISTRIBUTION(AXIALpipe) OF Concentration
ENTH_gas                        AS DISTRIBUTION(AXIALpipe) OF SpecificEnergy
E_gas                           AS DISTRIBUTION(AXIALpipe) OF SpecificEnergy
PRESS_gas                       AS DISTRIBUTION(AXIALpipe) OF Pressure
REACT_COMP_gas                  AS DISTRIBUTION(NoCOMP, AXIALpipe) OF Notype
RHO_COMP_gas                    AS DISTRIBUTION(NoCOMP, AXIALpipe) OF Density
RHO_gas                         AS DISTRIBUTION(AXIALpipe) OF Density
TAU0_gas                        AS DISTRIBUTION(AXIALpipe) OF Notype
TEMP_gas                        AS DISTRIBUTION(AXIALpipe) OF Temperature
TOTENTH_gas                     AS DISTRIBUTION(AXIALpipe) OF SpecificEnergy
U_gas                           AS DISTRIBUTION(AXIALpipe) OF SpecificEnergy
VELZ_gas                        AS DISTRIBUTION(AXIALpipe) OF Velocity
X_COMP_gas                      AS DISTRIBUTION(NoCOMP, AXIALpipe) OF Normalised
Y_COMP_gas                      AS DISTRIBUTION(NoCOMP, AXIALpipe) OF Normalised


 SET

 NoCOMP := 2;

 P_pipe := 2*3.141592654*(0.025);

 XAREA_pipe := 3.141592654*(0.025*0.025);


 EQUATION

 #1-D Mass Balances
 #==================

 FOR z := 0 TO 0.5 DO

         PARTIAL((RHO_COMP_gas(, z)*VELZ_gas(z)), AXIALpipe)
          - REACT_COMP_gas(, z) = 0;

 END

 #1-D Energy Balance
 #==================

 FOR z := 0 TO 0.5 DO

         PARTIAL((RHO_gas(z)*VELZ_gas(z)*TOTENTH_gas(z)), AXIALpipe) = 0;

 END

 #1-D Momentum Balance
 #====================

 FOR z := 0 TO 0.5 DO

          PARTIAL((RHO_gas(z)*VELZ_gas(z)*VELZ_gas(z)), AXIALpipe)
           + PARTIAL(PRESS_gas(z), AXIALpipe)
           + (TAU0_gas(z)*(P_pipe/XAREA_pipe)) = 0;

 END


#Equations of State
#==================

FOR z := 0 TO 0.5 DO

#ENTH_gas(z) = ENTH_gas(z)(PRESS_gas(z), TEMP_gas(z), X_COMP_gas(, z))

#RHO_gas(z) = RHO_gas(z)(PRESS_gas(z), TEMP_gas(z), X_COMP_gas(, z))

END


#Constitutive Equations
#======================

FOR z := 0 TO 0.5 DO

  RHO_COMP_gas(, z) = CONC_COMP_gas(, z)*MW;

  RHO_COMP_gas(, z) = RHO_gas(z)*X_COMP_gas(, z);

  RHO_gas(z) = SIGMA(RHO_COMP_gas(, z));

  CONC_COMP_gas(, z) = CONC_gas(z)*Y_COMP_gas(, z);

  CONC_gas(z) = SIGMA(CONC_COMP_gas(, z));

  E_gas(z) = U_gas(z)+0.5*VELZ_gas(z)*VELZ_gas(z);

  ENTH_gas(z)*RHO_gas(z) = U_gas(z)*RHO_gas(z) + PRESS_gas(z);

  TOTENTH_gas(z) = ENTH_gas(z)+0.5*VELZ_gas(z)*VELZ_gas(z);

END

END #model pipe


MODEL FLOWSHEET

UNIT    RESERVOIR_R1    AS R1
UNIT    RESERVOIR_R2    AS R2
UNIT    VESSEL_pipe     AS pipe

EQUATION

RESERVOIR_R1.TEMP = VESSEL_pipe.TEMP_gas(0);

RESERVOIR_R1.VEL = VESSEL_pipe.VELZ_gas(0);

RESERVOIR_R1.RHO_COMP = VESSEL_pipe.RHO_COMP_gas(, 0);

VESSEL_pipe.TEMP_gas(0.5) = RESERVOIR_R2.TEMP;

VESSEL_pipe.VELZ_gas(0.5) = RESERVOIR_R2.VEL;

VESSEL_pipe.RHO_COMP_gas(, 0.5) = RESERVOIR_R2.RHO_COMP;


END #model FLOWSHEET
```

E.2 Vinyl Acetate Reactor

```
DECLARE
  TYPE

        Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
        Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
        Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
        SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
        Temperature      = 300 : 0 : 1e4              UNIT = "K"
        Positive         = 10 : 0 :10e10              UNIT = "-"
        Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
        Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
        Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END


MODEL R1

PARAMETER

NoCOMP                          AS INTEGER
MW                              AS ARRAY(NoCOMP) OF REAL


VARIABLE

CONC                            AS Concentration
CONC_COMP                       AS ARRAY(NoCOMP) OF Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
RHO_COMP                        AS ARRAY(NoCOMP) OF Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity
X_COMP                          AS ARRAY(NoCOMP) OF Normalised
Y_COMP                          AS ARRAY(NoCOMP) OF Normalised


SET

NoCOMP := 7;


EQUATION

#Equations of State
#===================

#ENTH = ENTH(PRESS, TEMP, X_COMP)

#RHO = RHO(PRESS, TEMP, X_COMP)


#Constitutive Equations
#======================

RHO_COMP = CONC_COMP*MW;

RHO_COMP = RHO*X_COMP;

RHO = SIGMA(RHO_COMP);

CONC_COMP = CONC*Y_COMP;

CONC = SIGMA(CONC_COMP);

ENTH*RHO = U*RHO + PRESS;

END #model R1


MODEL R2

PARAMETER

NoCOMP                          AS INTEGER
MW                              AS ARRAY(NoCOMP) OF REAL


VARIABLE

CONC                            AS Concentration
CONC_COMP                       AS ARRAY(NoCOMP) OF Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
RHO_COMP                        AS ARRAY(NoCOMP) OF Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity
X_COMP                          AS ARRAY(NoCOMP) OF Normalised
Y_COMP                          AS ARRAY(NoCOMP) OF Normalised


SET

NoCOMP := 7;


EQUATION

#Equations of State
#===================

#ENTH = ENTH(PRESS, TEMP, X_COMP)

#RHO = RHO(PRESS, TEMP, X_COMP)


#Constitutive Equations
#======================

RHO_COMP = CONC_COMP*MW;

RHO_COMP = RHO*X_COMP;

RHO = SIGMA(RHO_COMP);

CONC_COMP = CONC*Y_COMP;

CONC = SIGMA(CONC_COMP);

ENTH*RHO = U*RHO + PRESS;

END #model R2


MODEL R3

VARIABLE

TEMP                            AS Temperature

END #model R3
```

```
MODEL reactorTube

PARAMETER

NoCOMP                  AS INTEGER
XAREA_reactorTube       AS REAL
P_wall                  AS REAL
P_reactorTube           AS REAL
MW                      AS ARRAY(NoCOMP) OF REAL


DISTRIBUTION_DOMAIN

AXIALreactorTube                AS (0 : 10)


VARIABLE

ALPHA_wall              AS DISTRIBUTION(AXIALreactorTube) OF Notype
CONC_COMP_gas           AS DISTRIBUTION(NoCOMP, AXIALreactorTube) OF Concentration
CONC_gas                AS DISTRIBUTION(AXIALreactorTube) OF Concentration
ENTH_gas                AS DISTRIBUTION(AXIALreactorTube) OF SpecificEnergy
E_gas                   AS DISTRIBUTION(AXIALreactorTube) OF SpecificEnergy
PRESS_gas               AS DISTRIBUTION(AXIALreactorTube) OF Pressure
Q_wall                  AS DISTRIBUTION(AXIALreactorTube) OF Notype
REACT_COMP_gas          AS DISTRIBUTION(NoCOMP, AXIALreactorTube) OF Notype
RHO_COMP_gas            AS DISTRIBUTION(NoCOMP, AXIALreactorTube) OF Density
RHO_gas                 AS DISTRIBUTION(AXIALreactorTube) OF Density
TAU0_gas                AS DISTRIBUTION(AXIALreactorTube) OF Notype
TEMP_gas                AS DISTRIBUTION(AXIALreactorTube) OF Temperature
TOTENTH_gas             AS DISTRIBUTION(AXIALreactorTube) OF SpecificEnergy
U_gas                   AS DISTRIBUTION(AXIALreactorTube) OF SpecificEnergy
VELZ_gas                AS DISTRIBUTION(AXIALreactorTube) OF Velocity
X_COMP_gas              AS DISTRIBUTION(NoCOMP, AXIALreactorTube) OF Normalised
Y_COMP_gas              AS DISTRIBUTION(NoCOMP, AXIALreactorTube) OF Normalised


SET

NoCOMP := 7;

P_wall := 2*3.141592654*0.0185;

XAREA_reactorTube := 3.141592654*(0.0185*0.0185);

P_reactorTube := 2*3.141592654*(0.0185);


EQUATION

#1-D Mass Balances
#===================

FOR z := 0 TO 10 DO

  $RHO_COMP_gas(, z)
        + PARTIAL((RHO_COMP_gas(, z)*VELZ_gas(z)), AXIALreactorTube)
        - REACT_COMP_gas(, z) = 0;

END

#1-D Energy Balance
#===================

FOR z := 0 TO 10 DO

  $(E_gas(z)*RHO_gas(z))
        + PARTIAL((RHO_gas(z)*VELZ_gas(z)*TOTENTH_gas(z)), AXIALreactorTube)
        - Q_wall(z)*(P_wall/XAREA_reactorTube) = 0;

END

#1-D Momentum Balance
#====================

FOR z := 0 TO 10 DO

        PARTIAL((RHO_gas(z)*VELZ_gas(z)*VELZ_gas(z)), AXIALreactorTube)
         + PARTIAL(PRESS_gas(z), AXIALreactorTube)
         + (TAU0_gas(z)*(P_reactorTube/XAREA_reactorTube)) = 0;

END


#Equations of State
#==================

FOR z := 0 TO 10 DO

#ENTH_gas(z) = ENTH_gas(z)(PRESS_gas(z), TEMP_gas(z), X_COMP_gas(, z))

#RHO_gas(z) = RHO_gas(z)(PRESS_gas(z), TEMP_gas(z), X_COMP_gas(, z))

END


#Constitutive Equations
#======================

FOR z := 0 TO 10 DO

  RHO_COMP_gas(, z) = CONC_COMP_gas(, z)*MW;

  RHO_COMP_gas(, z) = RHO_gas(z)*X_COMP_gas(, z);

  RHO_gas(z) = SIGMA(RHO_COMP_gas(, z));

  CONC_COMP_gas(, z) = CONC_gas(z)*Y_COMP_gas(, z);

  CONC_gas(z) = SIGMA(CONC_COMP_gas(, z));

  E_gas(z) = U_gas(z)+0.5*VELZ_gas(z)*VELZ_gas(z);

  ENTH_gas(z)*RHO_gas(z) = U_gas(z)*RHO_gas(z) + PRESS_gas(z);

  TOTENTH_gas(z) = ENTH_gas(z)+0.5*VELZ_gas(z)*VELZ_gas(z);

END

END #model reactorTube


MODEL FLOWSHEET

UNIT    RESERVOIR_R1    AS R1
UNIT    RESERVOIR_R2    AS R2
UNIT    RESERVOIR_R3    AS R3
UNIT    VESSEL_reactorTube      AS reactorTube

EQUATION

RESERVOIR_R1.TEMP = VESSEL_reactorTube.TEMP_gas(0);
```

```
RESERVOIR_R1.VEL = VESSEL_reactorTube.VELZ_gas(0);

RESERVOIR_R1.RHO_COMP = VESSEL_reactorTube.RHO_COMP_gas(, 0);

VESSEL_reactorTube.TEMP_gas(10) = RESERVOIR_R2.TEMP;

VESSEL_reactorTube.VELZ_gas(10) = RESERVOIR_R2.VEL;

VESSEL_reactorTube.RHO_COMP_gas(, 10) = RESERVOIR_R2.RHO_COMP;

FOR z := 0 TO 10 DO

  VESSEL_reactorTube.Q_wall(z) = VESSEL_reactorTube.ALPHA_wall(z)
                                 *(RESERVOIR_R3.TEMP-VESSEL_reactorTube.TEMP_gas(z));

END

END #model FLOWSHEET
```

```
DECLARE
  TYPE

        Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
        Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
        Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
        SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
        Temperature      = 300 : 0 : 1e4              UNIT = "K"
        Positive         = 10 : 0 :10e10              UNIT = "-"
        Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
        Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
        Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END


MODEL R1

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R1


MODEL R2

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R2


MODEL R3

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R3


MODEL R4

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
```

```
PRESS                          AS Pressure
RHO                            AS Density
TEMP                           AS Temperature
U                              AS SpecificEnergy
VEL                            AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R4


MODEL inner

PARAMETER

AREA_inlet                     AS REAL
VOL_inner                      AS REAL
MW                             AS REAL
AREA_wall                      AS REAL
AREA_outlet                    AS REAL


VARIABLE

ALPHA_wall                     AS Notype
CONC_cold                      AS Concentration
CONC_inlet                     AS Concentration
CONC_outlet                    AS Concentration
ENTH_cold                      AS SpecificEnergy
ENTH_inlet                     AS SpecificEnergy
ENTH_outlet                    AS SpecificEnergy
F                              AS Notype
PRESS_cold                     AS Pressure
PRESS_inlet                    AS Pressure
PRESS_outlet                   AS Pressure
Q_wall                         AS Notype
RHO                            AS Density
TEMP_cold                      AS Temperature
TEMP_inlet                     AS Temperature
TEMP_outlet                    AS Temperature
TEMP_wall                      AS Temperature
U_cold                         AS SpecificEnergy
U_inlet                        AS SpecificEnergy
U_outlet                       AS SpecificEnergy
VEL_inlet                      AS Velocity
VEL_outlet                     AS Velocity


EQUATION

#Lumped Mass Balances
#====================

0 =
        AREA_inlet*VEL_inlet
         - AREA_outlet*VEL_outlet;


#Lumped Energy Balance
#=====================

$(U_cold*VOL_inner*RHO) =
        RHO*VEL_inlet*AREA_inlet*(ENTH_inlet+0.5*VEL_inlet*VEL_inlet)
         - RHO*VEL_outlet*AREA_outlet*(ENTH_outlet+0.5*VEL_outlet*VEL_outlet)
         + AREA_wall*Q_wall;


#Lumped Momentum Balance
#=======================

0 =
        RHO*AREA_inlet*VEL_inlet*VEL_inlet
         + PRESS_inlet*AREA_inlet
         - RHO*AREA_outlet*VEL_outlet*VEL_outlet
         - PRESS_outlet*AREA_outlet
         - F;


#Equations of State
#==================

#ENTH_cold = ENTH_cold(TEMP_cold, PRESS_cold)

#ENTH_outlet = ENTH_outlet(TEMP_outlet, PRESS_outlet)

#ENTH_inlet = ENTH_inlet(TEMP_inlet, PRESS_inlet)


#Constitutive Equations
#======================

PRESS_cold = PRESS_outlet;

TEMP_cold = TEMP_wall;

TEMP_cold = TEMP_outlet;

RHO = CONC_cold*MW;

RHO = CONC_outlet*MW;

RHO = CONC_inlet*MW;

ENTH_cold*RHO = U_cold*RHO + PRESS_cold;

ENTH_outlet*RHO = U_outlet*RHO + PRESS_outlet;

ENTH_inlet*RHO = U_inlet*RHO + PRESS_inlet;

END #model inner


MODEL outer

PARAMETER

AREA_inlet                     AS REAL
VOL_outer                      AS REAL
MW                             AS REAL
AREA_wall                      AS REAL
```

```
AREA_outlet                    AS REAL

VARIABLE

ALPHA_wall                     AS Notype
CONC_hot                       AS Concentration
CONC_inlet                     AS Concentration
CONC_outlet                    AS Concentration
ENTH_hot                       AS SpecificEnergy
ENTH_inlet                     AS SpecificEnergy
ENTH_outlet                    AS SpecificEnergy
F                              AS Notype
PRESS_hot                      AS Pressure
PRESS_inlet                    AS Pressure
PRESS_outlet                   AS Pressure
Q_wall                         AS Notype
RHO                            AS Density
TEMP_hot                       AS Temperature
TEMP_inlet                     AS Temperature
TEMP_outlet                    AS Temperature
TEMP_wall                      AS Temperature
U_hot                          AS SpecificEnergy
U_inlet                        AS SpecificEnergy
U_outlet                       AS SpecificEnergy
VEL_inlet                      AS Velocity
VEL_outlet                     AS Velocity

EQUATION

#Lumped Mass Balances
#=====================

0 =
        AREA_inlet*VEL_inlet
         - AREA_outlet*VEL_outlet;

#Lumped Energy Balance
#=====================

$(U_hot*VOL_outer*RHO) =
        RHO*VEL_inlet*AREA_inlet*(ENTH_inlet+0.5*VEL_inlet*VEL_inlet)
         - RHO*VEL_outlet*AREA_outlet*(ENTH_outlet+0.5*VEL_outlet*VEL_outlet)
         + AREA_wall*Q_wall;

#Lumped Momentum Balance
#=======================

0 =
        RHO*AREA_inlet*VEL_inlet*VEL_inlet
         + PRESS_inlet*AREA_inlet
         - RHO*AREA_outlet*VEL_outlet*VEL_outlet
         - PRESS_outlet*AREA_outlet
         - F;

#Equations of State
#==================

#ENTH_hot = ENTH_hot(TEMP_hot, PRESS_hot)

#ENTH_outlet = ENTH_outlet(TEMP_outlet, PRESS_outlet)

#ENTH_inlet = ENTH_inlet(TEMP_inlet, PRESS_inlet)

#Constitutive Equations
#======================

PRESS_hot = PRESS_outlet;

TEMP_hot = TEMP_wall;

TEMP_hot = TEMP_outlet;

RHO = CONC_hot*MW;

RHO = CONC_outlet*MW;

RHO = CONC_inlet*MW;

ENTH_hot*RHO = U_hot*RHO + PRESS_hot;

ENTH_outlet*RHO = U_outlet*RHO + PRESS_outlet;

ENTH_inlet*RHO = U_inlet*RHO + PRESS_inlet;

END #model outer


MODEL FLOWSHEET

UNIT    RESERVOIR_R1    AS R1
UNIT    RESERVOIR_R2    AS R2
UNIT    RESERVOIR_R3    AS R3
UNIT    RESERVOIR_R4    AS R4
UNIT    VESSEL_outer    AS outer
UNIT    VESSEL_inner    AS inner

EQUATION

RESERVOIR_R1.PRESS = VESSEL_inner.PRESS_inlet;

RESERVOIR_R1.TEMP = VESSEL_inner.TEMP_inlet;

RESERVOIR_R1.VEL = VESSEL_inner.VEL_inlet;

RESERVOIR_R1.RHO = VESSEL_inner.RHO;

VESSEL_inner.PRESS_outlet = RESERVOIR_R2.PRESS;

VESSEL_inner.TEMP_outlet = RESERVOIR_R2.TEMP;

VESSEL_inner.VEL_outlet = RESERVOIR_R2.VEL;

VESSEL_inner.RHO = RESERVOIR_R2.RHO;

RESERVOIR_R3.PRESS = VESSEL_outer.PRESS_inlet;

RESERVOIR_R3.TEMP = VESSEL_outer.TEMP_inlet;

RESERVOIR_R3.VEL = VESSEL_outer.VEL_inlet;

RESERVOIR_R3.RHO = VESSEL_outer.RHO;

VESSEL_outer.PRESS_outlet = RESERVOIR_R4.PRESS;

VESSEL_outer.TEMP_outlet = RESERVOIR_R4.TEMP;

VESSEL_outer.VEL_outlet = RESERVOIR_R4.VEL;
```

```
VESSEL_outer.RHO = RESERVOIR_R4.RHO;

VESSEL_inner.Q_wall = (1/(1/VESSEL_inner.ALPHA_wall+1/VESSEL_outer.ALPHA_wall))
                      *(VESSEL_outer.TEMP_wall-VESSEL_inner.TEMP_wall);

VESSEL_inner.Q_wall = -VESSEL_outer.Q_wall;


END #model FLOWSHEET
```

E.4 Double Pipe Heat Exchanger - Case 2

```
DECLARE
  TYPE

        Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
        Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
        Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
        SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
        Temperature      = 300 : 0 : 1e4              UNIT = "K"
        Positive         = 10 : 0 :10e10              UNIT = "-"
        Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
        Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
        Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END

MODEL R1

 PARAMETER

 MW                              AS REAL

 VARIABLE

 CONC                            AS Concentration
 ENTH                            AS SpecificEnergy
 PRESS                           AS Pressure
 RHO                             AS Density
 TEMP                            AS Temperature
 U                               AS SpecificEnergy
 VEL                             AS Velocity

 EQUATION

 #Equations of State
 #===================

 #ENTH = ENTH(TEMP, PRESS)


 #Constitutive Equations
 #======================

 RHO = CONC*MW;

 ENTH*RHO = U*RHO + PRESS;

 END #model R1


 MODEL R2

 PARAMETER

 MW                              AS REAL

 VARIABLE

 CONC                            AS Concentration
 ENTH                            AS SpecificEnergy
 PRESS                           AS Pressure
 RHO                             AS Density
 TEMP                            AS Temperature
 U                               AS SpecificEnergy
 VEL                             AS Velocity

 EQUATION

 #Equations of State
 #===================

 #ENTH = ENTH(TEMP, PRESS)


 #Constitutive Equations
 #======================

 RHO = CONC*MW;

 ENTH*RHO = U*RHO + PRESS;

 END #model R2


 MODEL R3

 PARAMETER

 MW                              AS REAL

 VARIABLE

 CONC                            AS Concentration
 ENTH                            AS SpecificEnergy
 PRESS                           AS Pressure
 RHO                             AS Density
 TEMP                            AS Temperature
 U                               AS SpecificEnergy
 VEL                             AS Velocity

 EQUATION

 #Equations of State
 #===================

 #ENTH = ENTH(TEMP, PRESS)


 #Constitutive Equations
 #======================

 RHO = CONC*MW;

 ENTH*RHO = U*RHO + PRESS;

 END #model R3


 MODEL R4

 PARAMETER

 MW                              AS REAL

 VARIABLE

 CONC                            AS Concentration
 ENTH                            AS SpecificEnergy
```

```
PRESS                          AS Pressure
RHO                            AS Density
TEMP                           AS Temperature
U                              AS SpecificEnergy
VEL                            AS Velocity


EQUATION

#Equations of State
#===================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#=======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R4


MODEL outer

PARAMETER

AREA_inlet                     AS REAL
VOL_outer                      AS REAL
MW                             AS REAL
AREA_wall                      AS REAL
AREA_outlet                    AS REAL


VARIABLE

ALPHA_wall                     AS Notype
CONC_hot                       AS Concentration
CONC_inlet                     AS Concentration
CONC_outlet                    AS Concentration
ENTH_hot                       AS SpecificEnergy
ENTH_inlet                     AS SpecificEnergy
ENTH_outlet                    AS SpecificEnergy
F                              AS Notype
PRESS_hot                      AS Pressure
PRESS_inlet                    AS Pressure
PRESS_outlet                   AS Pressure
Q_wall                         AS Notype
RHO                            AS Density
TEMP_hot                       AS Temperature
TEMP_inlet                     AS Temperature
TEMP_outlet                    AS Temperature
TEMP_wall                      AS Temperature
U_hot                          AS SpecificEnergy
U_inlet                        AS SpecificEnergy
U_outlet                       AS SpecificEnergy
VEL_inlet                      AS Velocity
VEL_outlet                     AS Velocity
W                              AS Notype


EQUATION

#Lumped Mass Balances
#====================

0 =
        AREA_inlet*VEL_inlet
         - AREA_outlet*VEL_outlet;


#Lumped Energy Balance
#=====================

$(U_hot*VOL_outer*RHO) =
        RHO*VEL_inlet*AREA_inlet*(ENTH_inlet+0.5*VEL_inlet*VEL_inlet)
         - RHO*VEL_outlet*AREA_outlet*(ENTH_outlet+0.5*VEL_outlet*VEL_outlet)
         + AREA_wall*Q_wall
         - W;


#Lumped Momentum Balance
#=======================

0 =
        RHO*AREA_inlet*VEL_inlet*VEL_inlet
         + PRESS_inlet*AREA_inlet
         - RHO*AREA_outlet*VEL_outlet*VEL_outlet
         - PRESS_outlet*AREA_outlet
         - F;


#Equations of State
#==================

#ENTH_hot = ENTH_hot(TEMP_hot, PRESS_hot)

#ENTH_outlet = ENTH_outlet(TEMP_outlet, PRESS_outlet)

#ENTH_inlet = ENTH_inlet(TEMP_inlet, PRESS_inlet)


#Constitutive Equations
#=======================

RHO = CONC_hot*MW;

RHO = CONC_outlet*MW;

RHO = CONC_inlet*MW;

ENTH_hot*RHO = U_hot*RHO + PRESS_hot;

ENTH_outlet*RHO = U_outlet*RHO + PRESS_outlet;

ENTH_inlet*RHO = U_inlet*RHO + PRESS_inlet;

END #model outer


MODEL inner

PARAMETER

MW                             AS REAL
XAREA_inner                    AS REAL
P_wall                         AS REAL
P_inner                        AS REAL


DISTRIBUTION_DOMAIN
```

```
AXIALinner                    AS (0 : 36)

VARIABLE

ALPHA_wall                    AS DISTRIBUTION(AXIALinner) OF Notype
CONC_cold                     AS DISTRIBUTION(AXIALinner) OF Concentration
ENTH_cold                     AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
E_cold                        AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
PRESS_cold                    AS DISTRIBUTION(AXIALinner) OF Pressure
Q_wall                        AS DISTRIBUTION(AXIALinner) OF Notype
RHO                           AS Density
TAU0_cold                     AS DISTRIBUTION(AXIALinner) OF Notype
TEMP_cold                     AS DISTRIBUTION(AXIALinner) OF Temperature
TOTENTH_cold                  AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
U_cold                        AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
VELZ_cold                     AS DISTRIBUTION(AXIALinner) OF Velocity

 SET

 P_wall := 2*3.141592654*0.02105;

 XAREA_inner := 3.141592654*(0.02105*0.02105);

 P_inner := 2*3.141592654*(0.02105);

 EQUATION

 #1-D Mass Balances
 #==================

 FOR z := 0 TO 36 DO

        PARTIAL((VELZ_cold(z)), AXIALinner) = 0;

 END

 #1-D Energy Balance
 #==================

 FOR z := 0 TO 36 DO

   $(E_cold(z)*RHO)
          + PARTIAL((RHO*VELZ_cold(z)*TOTENTH_cold(z)), AXIALinner)
          - Q_wall(z)*(P_wall/XAREA_inner) = 0;

 END

 #1-D Momentum Balance
 #====================

 FOR z := 0 TO 36 DO

         PARTIAL((RHO*VELZ_cold(z)*VELZ_cold(z)), AXIALinner)
          + PARTIAL(PRESS_cold(z), AXIALinner)
          + (TAU0_cold(z)*(P_inner/XAREA_inner)) = 0;

 END


 #Equations of State
 #==================

 FOR z := 0 TO 36 DO

 #ENTH_cold(z) = ENTH_cold(z)(TEMP_cold(z), PRESS_cold(z))

 END


 #Constitutive Equations
 #======================

 FOR z := 0 TO 36 DO

   RHO = CONC_cold(z)*MW;

   E_cold(z) = U_cold(z)+0.5*VELZ_cold(z)*VELZ_cold(z);

   ENTH_cold(z)*RHO = U_cold(z)*RHO + PRESS_cold(z);

   TOTENTH_cold(z) = ENTH_cold(z)+0.5*VELZ_cold(z)*VELZ_cold(z);

 END

 END #model inner


 MODEL FLOWSHEET

 UNIT    RESERVOIR_R1    AS R1
 UNIT    RESERVOIR_R2    AS R2
 UNIT    RESERVOIR_R3    AS R3
 UNIT    RESERVOIR_R4    AS R4
 UNIT    VESSEL_outer    AS outer
 UNIT    VESSEL_inner    AS inner

 EQUATION

 RESERVOIR_R1.PRESS = VESSEL_inner.PRESS_cold(0);

 RESERVOIR_R1.TEMP = VESSEL_inner.TEMP_cold(0);

 RESERVOIR_R1.VEL = VESSEL_inner.VELZ_cold(0);

 RESERVOIR_R1.RHO = VESSEL_inner.RHO;

 VESSEL_inner.PRESS_cold(36) = RESERVOIR_R2.PRESS;

 VESSEL_inner.TEMP_cold(36) = RESERVOIR_R2.TEMP;

 VESSEL_inner.VELZ_cold(36) = RESERVOIR_R2.VEL;

 VESSEL_inner.RHO = RESERVOIR_R2.RHO;

 RESERVOIR_R3.PRESS = VESSEL_outer.PRESS_inlet;

 RESERVOIR_R3.TEMP = VESSEL_outer.TEMP_inlet;

 RESERVOIR_R3.VEL = VESSEL_outer.VEL_inlet;

 RESERVOIR_R3.RHO = VESSEL_outer.RHO;

 VESSEL_outer.PRESS_outlet = RESERVOIR_R4.PRESS;

 VESSEL_outer.TEMP_outlet = RESERVOIR_R4.TEMP;

 VESSEL_outer.VEL_outlet = RESERVOIR_R4.VEL;

 VESSEL_outer.RHO = RESERVOIR_R4.RHO;

 1/(36-0)*INTEGRAL(z := 0:36 ; VESSEL_inner.Q_wall(z)) = -VESSEL_outer.Q_wall;
```

```
FOR z := 0 TO 36 DO

  VESSEL_inner.Q_wall(z) = (1/(1/VESSEL_inner.ALPHA_wall(z)+1/VESSEL_outer.ALPHA_wall))
                           *(VESSEL_outer.TEMP_wall-VESSEL_inner.TEMP_cold(z));

END

END #model FLOWSHEET
```

E.5 Double Pipe Heat Exchanger - Case 3

```
DECLARE
  TYPE

        Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
        Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
        Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
        SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
        Temperature      = 300 : 0 : 1e4              UNIT = "K"
        Positive         = 10 : 0 :10e10              UNIT = "-"
        Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
        Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
        Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END


MODEL R1

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R1


MODEL R2

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R2


MODEL R3

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R3


MODEL R4

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
```

```
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#===================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R4


MODEL inner

PARAMETER

MW                              AS REAL
XAREA_inner                     AS REAL
P_wall                          AS REAL
P_inner                         AS REAL


DISTRIBUTION_DOMAIN

AXIALinner                      AS (0 : 36)


VARIABLE

ALPHA_wall                      AS DISTRIBUTION(AXIALinner) OF Notype
CONC_cold                       AS DISTRIBUTION(AXIALinner) OF Concentration
ENTH_cold                       AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
E_cold                          AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
PRESS_cold                      AS DISTRIBUTION(AXIALinner) OF Pressure
Q_wall                          AS DISTRIBUTION(AXIALinner) OF Notype
RHO                             AS Density
TAU0_cold                       AS DISTRIBUTION(AXIALinner) OF Notype
TEMP_cold                       AS DISTRIBUTION(AXIALinner) OF Temperature
TOTENTH_cold                    AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
U_cold                          AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
VELZ_cold                       AS DISTRIBUTION(AXIALinner) OF Velocity


SET

P_wall := 2*3.141592654*0.02105;

XAREA_inner := 3.141592654*(0.02105*0.02105);

P_inner := 2*3.141592654*(0.02105);


EQUATION

#1-D Mass Balances
#==================

FOR z := 0|+ TO 36 DO

        PARTIAL((VELZ_cold(z)), AXIALinner) = 0;

END

#1-D Energy Balance
#==================

FOR z := 0|+ TO 36 DO

  $(E_cold(z)*RHO)
         + PARTIAL((RHO*VELZ_cold(z)*TOTENTH_cold(z)), AXIALinner)
         - Q_wall(z)*(P_wall/XAREA_inner) = 0;

END

#1-D Momentum Balance
#====================

FOR z := 0|+ TO 36 DO

        PARTIAL((RHO*VELZ_cold(z)*VELZ_cold(z)), AXIALinner)
         + PARTIAL(PRESS_cold(z), AXIALinner)
         + (TAU0_cold(z)*(P_inner/XAREA_inner)) = 0;

END


#Equations of State
#==================

FOR z := 0 TO 36 DO

#ENTH_cold(z) = ENTH_cold(z)(TEMP_cold(z), PRESS_cold(z))

END


#Constitutive Equations
#======================

FOR z := 0 TO 36 DO

  RHO = CONC_cold(z)*MW;

  E_cold(z) = U_cold(z)+0.5*VELZ_cold(z)*VELZ_cold(z);

  ENTH_cold(z)*RHO = U_cold(z)*RHO + PRESS_cold(z);

  TOTENTH_cold(z) = ENTH_cold(z)+0.5*VELZ_cold(z)*VELZ_cold(z);

END

END #model inner


MODEL outer

PARAMETER

MW                              AS REAL
XAREA_outer                     AS REAL
P_wall                          AS REAL
```

```
P_outer                         AS REAL

DISTRIBUTION_DOMAIN

AXIALouter                      AS (0 : 36)

VARIABLE

ALPHA_wall                      AS DISTRIBUTION(AXIALouter) OF Notype
CONC_hot                        AS DISTRIBUTION(AXIALouter) OF Concentration
ENTH_hot                        AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
E_hot                           AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
PRESS_hot                       AS DISTRIBUTION(AXIALouter) OF Pressure
Q_wall                          AS DISTRIBUTION(AXIALouter) OF Notype
RHO                             AS Density
TAU0_hot                        AS DISTRIBUTION(AXIALouter) OF Notype
TEMP_hot                        AS DISTRIBUTION(AXIALouter) OF Temperature
TOTENTH_hot                     AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
U_hot                           AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
VELZ_hot                        AS DISTRIBUTION(AXIALouter) OF Velocity

 SET

 P_wall := 2*3.141592654*0.02105;

 XAREA_outer := 3.141592654*((0.02629*0.02629)-(0.02105*0.02105));

 P_outer := 2*3.141592654*(0.02105+0.02629);


 EQUATION

#1-D Mass Balances
#====================

FOR z := 0|+ TO 36 DO

        PARTIAL((VELZ_hot(z)), AXIALouter) = 0;

END

#1-D Energy Balance
#===================

FOR z := 0|+ TO 36 DO

  $(E_hot(z)*RHO)
         + PARTIAL((RHO*VELZ_hot(z)*TOTENTH_hot(z)), AXIALouter)
         - Q_wall(z)*(P_wall/XAREA_outer) = 0;

 END

#1-D Momentum Balance
#=====================

 FOR z := 0|+ TO 36 DO

         PARTIAL((RHO*VELZ_hot(z)*VELZ_hot(z)), AXIALouter)
          + PARTIAL(PRESS_hot(z), AXIALouter)
          + (TAU0_hot(z)*(P_outer/XAREA_outer)) = 0;

 END
```

```
#Equations of State
#===================

FOR z := 0 TO 36 DO

#ENTH_hot(z) = ENTH_hot(z)(TEMP_hot(z), PRESS_hot(z))

END


#Constitutive Equations
#======================

FOR z := 0 TO 36 DO

  RHO = CONC_hot(z)*MW;

  E_hot(z) = U_hot(z)+0.5*VELZ_hot(z)*VELZ_hot(z);

  ENTH_hot(z)*RHO = U_hot(z)*RHO + PRESS_hot(z);

  TOTENTH_hot(z) = ENTH_hot(z)+0.5*VELZ_hot(z)*VELZ_hot(z);

END

END #model outer


MODEL FLOWSHEET

UNIT    RESERVOIR_R1    AS R1
UNIT    RESERVOIR_R2    AS R2
UNIT    RESERVOIR_R3    AS R3
UNIT    RESERVOIR_R4    AS R4
UNIT    VESSEL_outer    AS outer
UNIT    VESSEL_inner    AS inner

EQUATION

RESERVOIR_R1.PRESS = VESSEL_inner.PRESS_cold(0);

RESERVOIR_R1.TEMP = VESSEL_inner.TEMP_cold(0);

RESERVOIR_R1.VEL = VESSEL_inner.VELZ_cold(0);

RESERVOIR_R1.RHO = VESSEL_inner.RHO;

VESSEL_inner.PRESS_cold(36) = RESERVOIR_R2.PRESS;

VESSEL_inner.TEMP_cold(36) = RESERVOIR_R2.TEMP;

VESSEL_inner.VELZ_cold(36) = RESERVOIR_R2.VEL;

VESSEL_inner.RHO = RESERVOIR_R2.RHO;

RESERVOIR_R3.PRESS = VESSEL_outer.PRESS_hot(0);

RESERVOIR_R3.TEMP = VESSEL_outer.TEMP_hot(0);

RESERVOIR_R3.VEL = VESSEL_outer.VELZ_hot(0);

RESERVOIR_R3.RHO = VESSEL_outer.RHO;

VESSEL_outer.PRESS_hot(36) = RESERVOIR_R4.PRESS;

VESSEL_outer.TEMP_hot(36) = RESERVOIR_R4.TEMP;
```

```
VESSEL_outer.VELZ_hot(36) = RESERVOIR_R4.VEL;

VESSEL_outer.RHO = RESERVOIR_R4.RHO;

FOR z := 0 TO 36 DO

  VESSEL_inner.Q_wall(z) = -VESSEL_outer.Q_wall(z);

END
FOR z := 0 TO 36 DO

  VESSEL_outer.Q_wall(z) = (1/(1/VESSEL_outer.ALPHA_wall(z)+1/VESSEL_inner.ALPHA_wall(z)))
                           *(VESSEL_inner.TEMP_cold(z)-VESSEL_outer.TEMP_hot(z));

END

END #model FLOWSHEET
```

E.6 Double Pipe Heat Exchanger - Case 4

```
DECLARE
  TYPE

       Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
       Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
       Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
       SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
       Temperature      = 300 : 0 : 1e4              UNIT = "K"
       Positive         = 10 : 0 :10e10              UNIT = "-"
       Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
       Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
       Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END


MODEL R1

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R1


MODEL R2

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R2


MODEL R3

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R3


MODEL R4

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
```

```
PRESS                          AS Pressure
RHO                            AS Density
TEMP                           AS Temperature
U                              AS SpecificEnergy
VEL                            AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R4


MODEL inner

PARAMETER

MW                             AS REAL
XAREA_inner                    AS REAL
P_wall                         AS REAL
P_inner                        AS REAL


DISTRIBUTION_DOMAIN

AXIALinner                     AS (0 : 36)


VARIABLE

ALPHA_wall                     AS DISTRIBUTION(AXIALinner) OF Notype
CONC_cold                      AS DISTRIBUTION(AXIALinner) OF Concentration
ENTH_cold                      AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
E_cold                         AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
PRESS_cold                     AS DISTRIBUTION(AXIALinner) OF Pressure
Q_wall                         AS DISTRIBUTION(AXIALinner) OF Notype
RHO                            AS Density
TAU0_cold                      AS DISTRIBUTION(AXIALinner) OF Notype
TEMP_cold                      AS DISTRIBUTION(AXIALinner) OF Temperature
TOTENTH_cold                   AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
U_cold                         AS DISTRIBUTION(AXIALinner) OF SpecificEnergy
VELZ_cold                      AS DISTRIBUTION(AXIALinner) OF Velocity


SET

P_wall := 2*3.141592654*0.02105;

XAREA_inner := 3.141592654*(0.02105*0.02105);

P_inner := 2*3.141592654*(0.02105);


EQUATION

#1-D Mass Balances
#==================

FOR z := 0 TO 36 DO

        PARTIAL((VELZ_cold(z)), AXIALinner) = 0;

END

#1-D Energy Balance
#==================

FOR z := 0 TO 36 DO

  $(E_cold(z)*RHO)
         + PARTIAL((RHO*VELZ_cold(z)*TOTENTH_cold(z)), AXIALinner)
         - Q_wall(z)*(P_wall/XAREA_inner) = 0;

END

#1-D Momentum Balance
#====================

FOR z := 0 TO 36 DO

        PARTIAL((RHO*VELZ_cold(z)*VELZ_cold(z)), AXIALinner)
         + PARTIAL(PRESS_cold(z), AXIALinner)
         + (TAU0_cold(z)*(P_inner/XAREA_inner)) = 0;

END


#Equations of State
#==================

FOR z := 0 TO 36 DO

#ENTH_cold(z) = ENTH_cold(z)(TEMP_cold(z), PRESS_cold(z))

END


#Constitutive Equations
#======================

FOR z := 0 TO 36 DO

  RHO = CONC_cold(z)*MW;

  E_cold(z) = U_cold(z)+0.5*VELZ_cold(z)*VELZ_cold(z);

  ENTH_cold(z)*RHO = U_cold(z)*RHO + PRESS_cold(z);

  TOTENTH_cold(z) = ENTH_cold(z)+0.5*VELZ_cold(z)*VELZ_cold(z);

END

END #model inner


MODEL outer

PARAMETER

MW                             AS REAL
XAREA_outer                    AS REAL
P_wall                         AS REAL
```

```
P_outer                         AS REAL

DISTRIBUTION_DOMAIN

AXIALouter                      AS (0 : 36)

VARIABLE

ALPHA_wall                      AS DISTRIBUTION(AXIALouter) OF Notype
CONC_hot                        AS DISTRIBUTION(AXIALouter) OF Concentration
ENTH_hot                        AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
E_hot                           AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
PRESS_hot                       AS DISTRIBUTION(AXIALouter) OF Pressure
Q_wall                          AS DISTRIBUTION(AXIALouter) OF Notype
RHO                             AS Density
TAU0_hot                        AS DISTRIBUTION(AXIALouter) OF Notype
TEMP_hot                        AS DISTRIBUTION(AXIALouter) OF Temperature
TOTENTH_hot                     AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
U_hot                           AS DISTRIBUTION(AXIALouter) OF SpecificEnergy
VELZ_hot                        AS DISTRIBUTION(AXIALouter) OF Velocity

 SET

 P_wall := 2*3.141592654*0.02105;

 XAREA_outer := 3.141592654*((0.02629*0.02629)-(0.02105*0.02105));

 P_outer := 2*3.141592654*(0.02105+0.02629);


 EQUATION

#1-D Mass Balances
#===================

FOR z := 0 TO 36 DO

        PARTIAL((VELZ_hot(z)), AXIALouter) = 0;

END

#1-D Energy Balance
#===================

FOR z := 0 TO 36 DO

  $(E_hot(z)*RHO)
         + PARTIAL((RHO*VELZ_hot(z)*TOTENTH_hot(z)), AXIALouter)
         - Q_wall(z)*(P_wall/XAREA_outer) = 0;

 END

 #1-D Momentum Balance
 #====================

 FOR z := 0 TO 36 DO

         PARTIAL((RHO*VELZ_hot(z)*VELZ_hot(z)), AXIALouter)
          + PARTIAL(PRESS_hot(z), AXIALouter)
          + (TAU0_hot(z)*(P_outer/XAREA_outer)) = 0;

 END
```

```
#Equations of State
#===================

FOR z := 0 TO 36 DO

#ENTH_hot(z) = ENTH_hot(z)(TEMP_hot(z), PRESS_hot(z))

END


#Constitutive Equations
#======================

FOR z := 0 TO 36 DO

  RHO = CONC_hot(z)*MW;

  E_hot(z) = U_hot(z)+0.5*VELZ_hot(z)*VELZ_hot(z);

  ENTH_hot(z)*RHO = U_hot(z)*RHO + PRESS_hot(z);

  TOTENTH_hot(z) = ENTH_hot(z)+0.5*VELZ_hot(z)*VELZ_hot(z);

END

END #model outer


MODEL FLOWSHEET

UNIT    RESERVOIR_R1    AS R1
UNIT    RESERVOIR_R2    AS R2
UNIT    RESERVOIR_R3    AS R3
UNIT    RESERVOIR_R4    AS R4
UNIT    VESSEL_outer    AS outer
UNIT    VESSEL_inner    AS inner

EQUATION

RESERVOIR_R1.PRESS = VESSEL_inner.PRESS_cold(0);

RESERVOIR_R1.TEMP = VESSEL_inner.TEMP_cold(0);

RESERVOIR_R1.VEL = VESSEL_inner.VELZ_cold(0);

RESERVOIR_R1.RHO = VESSEL_inner.RHO;

VESSEL_inner.PRESS_cold(36) = RESERVOIR_R2.PRESS;

VESSEL_inner.TEMP_cold(36) = RESERVOIR_R2.TEMP;

VESSEL_inner.VELZ_cold(36) = RESERVOIR_R2.VEL;

VESSEL_inner.RHO = RESERVOIR_R2.RHO;

RESERVOIR_R3.PRESS = VESSEL_outer.PRESS_hot(0);

RESERVOIR_R3.TEMP = VESSEL_outer.TEMP_hot(0);

RESERVOIR_R3.VEL = VESSEL_outer.VELZ_hot(0);

RESERVOIR_R3.RHO = VESSEL_outer.RHO;

VESSEL_outer.PRESS_hot(36) = RESERVOIR_R4.PRESS;

VESSEL_outer.TEMP_hot(36) = RESERVOIR_R4.TEMP;
```

```
VESSEL_outer.VELZ_hot(36) = RESERVOIR_R4.VEL;

VESSEL_outer.RHO = RESERVOIR_R4.RHO;

FOR z := 0 TO 36 DO

  VESSEL_inner.Q_wall(z) = -VESSEL_outer.Q_wall(36-z);

END
FOR z := 0 TO 36 DO

  VESSEL_outer.Q_wall(36-z) = (1/(
                              1/VESSEL_outer.ALPHA_wall(36-z)+1/VESSEL_inner.ALPHA_wall(z))
                              )*(VESSEL_inner.TEMP_cold(z)-VESSEL_outer.TEMP_hot(36-z));

END

END #model FLOWSHEET
```

E.7 Laminar Pipe Flow

```
DECLARE
  TYPE

        Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
        Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
        Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
        SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
        Temperature      = 300 : 0 : 1e4              UNIT = "K"
        Positive         = 10 : 0 :10e10              UNIT = "-"
        Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
        Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
        Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END


MODEL R1

PARAMETER

MW              AS REAL


VARIABLE

CONC            AS Concentration
ENTH            AS SpecificEnergy
PRESS           AS Pressure
RHO             AS Density
TEMP            AS Temperature
U               AS SpecificEnergy
VEL             AS Velocity


EQUATION

#Equations of State
#===================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R1


MODEL R2

PARAMETER

MW                  AS REAL


VARIABLE

CONC                AS Concentration
ENTH                AS SpecificEnergy
PRESS               AS Pressure
RHO                 AS Density
TEMP                AS Temperature
U                   AS SpecificEnergy
VEL                 AS Velocity


EQUATION

#Equations of State
#===================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R2


MODEL pipe

PARAMETER

MW                          AS REAL


DISTRIBUTION_DOMAIN

AXIALpipe                   AS (0 : 0.06)
RADIALpipe                  AS (0 : 0.02)


VARIABLE

CONC_glycerol               AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Concentration
ENTH_glycerol               AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
E_glycerol                  AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
PRESS_glycerol              AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Pressure
RHO                         AS Density
TAU00_glycerol              AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURR_glycerol              AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURZ_glycerol              AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAUZZ_glycerol              AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TEMP                        AS Temperature
TOTENTH_glycerol            AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
U_glycerol                  AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
VELR_glycerol               AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Velocity
VELZ_glycerol               AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Velocity
VISC_glycerol               AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype


BOUNDARY

#Symmetry and Wall Boundary Conditions
#=====================================

FOR z := 0 TO 0.06 DO

    VELZ_glycerol(z, 0.02) = 0;

END
FOR z := 0 TO 0.06 DO

    VELR_glycerol(z, 0.02) = 0;

END
```

```
FOR z := 0 TO 0.06 DO

    PARTIAL(PRESS_glycerol(z, 0), RADIALpipe) = 0;

END
FOR z := 0 TO 0.06 DO

    PARTIAL(VELZ_glycerol(z, 0), RADIALpipe) = 0;

END
FOR z := 0 TO 0.06 DO

    VELR_glycerol(z, 0) = 0;

END

EQUATION

#2-D Mass Balances
#===================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

        PARTIAL((VELZ_glycerol(z, r)), AXIALpipe)
         + 1/r*PARTIAL((r*VELR_glycerol(z, r)), RADIALpipe) = 0;

  END
END

#2-D Momentum Balances
#======================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

       PARTIAL((RHO*VELZ_glycerol(z, r)*VELR_glycerol(z, r)), AXIALpipe)
        + (1/r*PARTIAL((r*RHO*VELR_glycerol(z, r)*VELR_glycerol(z, r)), RADIALpipe))
        + PARTIAL(PRESS_glycerol(z, r), RADIALpipe)
        - PARTIAL(TAURZ_glycerol(z, r), AXIALpipe)
        - 1/r*PARTIAL((r*TAURR_glycerol(z, r)), RADIALpipe)
        - (-1/r*TAU00_glycerol(z, r)) = 0;

  END
END

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

       PARTIAL((RHO*VELZ_glycerol(z, r)*VELZ_glycerol(z, r)), AXIALpipe)
        + (1/r*PARTIAL((r*RHO*VELR_glycerol(z, r)*VELZ_glycerol(z, r)), RADIALpipe))
        + PARTIAL(PRESS_glycerol(z, r), AXIALpipe)
        - PARTIAL(TAUZZ_glycerol(z, r), AXIALpipe)
        - 1/r*PARTIAL((r*TAURZ_glycerol(z, r)), RADIALpipe) = 0;

  END
END


#Equations of State
#==================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

  #ENTH_glycerol(z, r) = ENTH_glycerol(z, r)(TEMP, PRESS_glycerol(z, r))
  END
END


#Constitutive Equations
#======================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

  RHO = CONC_glycerol(z, r)*MW;

  E_glycerol(z, r) = U_glycerol(z, r)+0.5*(VELZ_glycerol(z, r)*VELZ_glycerol(z, r)
                     +VELR_glycerol(z, r)*VELR_glycerol(z, r));

  ENTH_glycerol(z, r)*RHO = U_glycerol(z, r)*RHO + PRESS_glycerol(z, r);

  TOTENTH_glycerol(z, r) = ENTH_glycerol(z, r)+0.5*(VELZ_glycerol(z, r)*VELZ_glycerol(z, r)
                           +VELR_glycerol(z, r)*VELR_glycerol(z, r));

  TAUZZ_glycerol(z, r) = 2*VISC_glycerol(z, r)*PARTIAL(VELZ_glycerol(z, r), AXIALpipe);

  TAURZ_glycerol(z, r) = VISC_glycerol(z, r)*(PARTIAL(VELZ_glycerol(z, r), RADIALpipe)
                         +PARTIAL(VELR_glycerol(z, r), AXIALpipe));

  TAURR_glycerol(z, r) = 2*VISC_glycerol(z, r)*PARTIAL(VELR_glycerol(z, r), RADIALpipe);

  TAU00_glycerol(z, r) = 2*VISC_glycerol(z, r)*1/r*VELR_glycerol(z, r);

  END
END

END #model pipe


MODEL FLOWSHEET

UNIT    RESERVOIR_R1    AS R1
UNIT    RESERVOIR_R2    AS R2
UNIT    VESSEL_pipe     AS pipe

EQUATION

FOR r := 0 TO 0.02 DO

  RESERVOIR_R1.PRESS = VESSEL_pipe.PRESS_glycerol(0,r);

END
RESERVOIR_R1.TEMP = VESSEL_pipe.TEMP;

FOR r := 0 TO 0.02 DO

  VESSEL_pipe.VELR_glycerol(0,r) = 0;

END
FOR r := 0 TO 0.02 DO

  RESERVOIR_R1.VEL = VESSEL_pipe.VELZ_glycerol(0,r);

END
RESERVOIR_R1.RHO = VESSEL_pipe.RHO;

2/((0.02-0)*(0.02-0))*INTEGRAL(r := 0:0.02 ;
                               r*VESSEL_pipe.PRESS_glycerol(0.06,r)) = RESERVOIR_R2.PRESS;

VESSEL_pipe.TEMP = RESERVOIR_R2.TEMP;
```

```
2/((0.02-0)*(0.02-0))*INTEGRAL(r := 0:0.02 ;
                               r*VESSEL_pipe.VELZ_glycerol(0.06,r)) = RESERVOIR_R2.VEL;

VESSEL_pipe.RHO = RESERVOIR_R2.RHO;

FOR r := 0 TO 0.02 DO

  PARTIAL(VESSEL_pipe.VELR_glycerol(0.06,r), AXIALpipe) = 0;

END
FOR r := 0 TO 0.02 DO

  PARTIAL(VESSEL_pipe.VELZ_glycerol(0.06,r), AXIALpipe) = 0;

END

END #model FLOWSHEET
```

E.8 Turbulent Pipe Flow

```
DECLARE
  TYPE

        Density          = 10 : 0 : 10e6              UNIT = "kg/m3"
        Concentration    = 10 : 0 : 10e6              UNIT = "mole/m3"
        Velocity         = 1 : -5e3 : 5e3             UNIT = "m/s2"
        SpecificEnergy   = 1 : -10e10 : 10e10         UNIT = "J/kg"
        Temperature      = 300 : 0 : 1e4              UNIT = "K"
        Positive         = 10 : 0 :10e10              UNIT = "-"
        Pressure         = 10 : 1 : 1e10              UNIT = "N/m2"
        Normalised       = 0.5 : -0.0001 : 1.001      UNIT = "-"
        Notype           = 1 : -10e10 : 10e10         UNIT = "-"
END


MODEL R1

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R1


MODEL R2

PARAMETER

MW                              AS REAL


VARIABLE

CONC                            AS Concentration
ENTH                            AS SpecificEnergy
PRESS                           AS Pressure
RHO                             AS Density
TEMP                            AS Temperature
U                               AS SpecificEnergy
VEL                             AS Velocity


EQUATION

#Equations of State
#==================

#ENTH = ENTH(TEMP, PRESS)


#Constitutive Equations
#======================

RHO = CONC*MW;

ENTH*RHO = U*RHO + PRESS;

END #model R2


MODEL pipe

PARAMETER

MW                              AS REAL


DISTRIBUTION_DOMAIN

AXIALpipe                       AS (0 : 0.06)
RADIALpipe                      AS (0 : 0.02)


VARIABLE

CONC_liquid                     AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Concentration
ENTH_liquid                     AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
E_liquid                        AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
PRESS_liquid                    AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Pressure
RHO                             AS Density
TAU00_liquid                    AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAU00eff_liquid                 AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAU00turb_liquid                AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURR_liquid                    AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURReff_liquid                 AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURRturb_liquid                AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURZ_liquid                    AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURZeff_liquid                 AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAURZturb_liquid                AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAUZZ_liquid                    AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAUZZeff_liquid                 AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TAUZZturb_liquid                AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
TEMP                            AS Temperature
TOTENTH_liquid                  AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
U_liquid                        AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy
VELR_liquid                     AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Velocity
VELZ_liquid                     AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Velocity
VISC_liquid                     AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
VISCturb_liquid                 AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF Notype
k_liquid                        AS DISTRIBUTION(AXIALpipe, RADIALpipe) OF SpecificEnergy


BOUNDARY

#Symmetry and Wall Boundary Conditions
#=====================================
```

```
FOR z := 0 TO 0.06 DO

    VELZ_liquid(z, 0.02) = 0;

END
FOR z := 0 TO 0.06 DO

    VELR_liquid(z, 0.02) = 0;

END
FOR z := 0 TO 0.06 DO

    PARTIAL(PRESS_liquid(z, 0), RADIALpipe) = 0;

END
FOR z := 0 TO 0.06 DO

    PARTIAL(VELZ_liquid(z, 0), RADIALpipe) = 0;

END
FOR z := 0 TO 0.06 DO

    VELR_liquid(z, 0) = 0;

END

EQUATION

#2-D Mass Balances
#===================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

       PARTIAL((VELZ_liquid(z, r)), AXIALpipe)
        + 1/r*PARTIAL((r*VELR_liquid(z, r)), RADIALpipe) = 0;

  END
END

#2-D Momentum Balances
#======================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

       PARTIAL((RHO*VELZ_liquid(z, r)*VELR_liquid(z, r)), AXIALpipe)
        + (1/r*PARTIAL((r*RHO*VELR_liquid(z, r)*VELR_liquid(z, r)), RADIALpipe))
        + PARTIAL(PRESS_liquid(z, r), RADIALpipe)
        - PARTIAL(TAURZeff_liquid(z, r), AXIALpipe)
        - 1/r*PARTIAL((r*TAURReff_liquid(z, r)), RADIALpipe)
        - (-1/r*TAU00eff_liquid(z, r)) = 0;

  END
END

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

       PARTIAL((RHO*VELZ_liquid(z, r)*VELZ_liquid(z, r)), AXIALpipe)
        + (1/r*PARTIAL((r*RHO*VELR_liquid(z, r)*VELZ_liquid(z, r)), RADIALpipe))
        + PARTIAL(PRESS_liquid(z, r), AXIALpipe)
        - PARTIAL(TAUZZeff_liquid(z, r), AXIALpipe)
        - 1/r*PARTIAL((r*TAURZeff_liquid(z, r)), RADIALpipe) = 0;

  END
END
```

```
#Equations of State
#===================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

  #ENTH_liquid(z, r) = ENTH_liquid(z, r)(TEMP, PRESS_liquid(z, r))

  END
END


#Constitutive Equations
#=======================

FOR z := 0 TO 0.06 DO
  FOR r := 0 TO 0.02 DO

  RHO = CONC_liquid(z, r)*MW;

  E_liquid(z, r) = U_liquid(z, r)+0.5*(VELZ_liquid(z, r)*VELZ_liquid(z, r)
                   +VELR_liquid(z, r)*VELR_liquid(z, r))+k_liquid(z, r);

  ENTH_liquid(z, r)*RHO = U_liquid(z, r)*RHO + PRESS_liquid(z, r);

  TOTENTH_liquid(z, r) = ENTH_liquid(z, r)+0.5*(VELZ_liquid(z, r)*VELZ_liquid(z, r)
                         +VELR_liquid(z, r)*VELR_liquid(z, r))+k_liquid(z, r);

  TAUZZ_liquid(z, r) = 2*VISC_liquid(z, r)*PARTIAL(VELZ_liquid(z, r), AXIALpipe);

  TAUZZturb_liquid(z, r) = 2*VISCturb_liquid(z, r)*PARTIAL(VELZ_liquid(z, r), AXIALpipe)
                           + (-2/3*RHO*k_liquid(z, r));

  TAUZZeff_liquid(z, r) = TAUZZ_liquid(z, r)+TAUZZturb_liquid(z, r);

  TAURZ_liquid(z, r) = VISC_liquid(z, r)*(PARTIAL(VELZ_liquid(z, r), RADIALpipe)
                                          +PARTIAL(VELR_liquid(z, r), AXIALpipe));

  TAURZturb_liquid(z, r) = VISCturb_liquid(z, r)*(PARTIAL(VELZ_liquid(z, r), RADIALpipe)
                                                  +PARTIAL(VELR_liquid(z, r), AXIALpipe));

  TAURZeff_liquid(z, r) = TAURZ_liquid(z, r)+TAURZturb_liquid(z, r);

  TAURR_liquid(z, r) = 2*VISC_liquid(z, r)*PARTIAL(VELR_liquid(z, r), RADIALpipe);

  TAURRturb_liquid(z, r) = 2*VISCturb_liquid(z, r)*PARTIAL(VELR_liquid(z, r), RADIALpipe)
                           + (-2/3*RHO*k_liquid(z, r));

  TAURReff_liquid(z, r) = TAURR_liquid(z, r)+TAURRturb_liquid(z, r);

  TAU00_liquid(z, r) = 2*VISC_liquid(z, r)*1/r*VELR_liquid(z, r);

  TAU00turb_liquid(z, r) = 2*VISCturb_liquid(z, r)*1/r*VELR_liquid(z, r)
                           + (-2/3*RHO*k_liquid(z, r));

  TAU00eff_liquid(z, r) = TAU00_liquid(z, r)+TAU00turb_liquid(z, r);

  END
END

END #model pipe


MODEL FLOWSHEET
```

```
UNIT    RESERVOIR_R1    AS R1
UNIT    RESERVOIR_R2    AS R2
UNIT    VESSEL_pipe     AS pipe

EQUATION

FOR r := 0 TO 0.02 DO

  RESERVOIR_R1.PRESS = VESSEL_pipe.PRESS_liquid(0,r);

END
RESERVOIR_R1.TEMP = VESSEL_pipe.TEMP;

FOR r := 0 TO 0.02 DO

  VESSEL_pipe.VELR_liquid(0,r) = 0;

END
FOR r := 0 TO 0.02 DO

  RESERVOIR_R1.VEL = VESSEL_pipe.VELZ_liquid(0,r);

END
RESERVOIR_R1.RHO = VESSEL_pipe.RHO;

2/((0.02-0)*(0.02-0))*INTEGRAL(r := 0:0.02 ;
                               r*VESSEL_pipe.PRESS_liquid(0.06,r)) = RESERVOIR_R2.PRESS;

VESSEL_pipe.TEMP = RESERVOIR_R2.TEMP;

2/((0.02-0)*(0.02-0))*INTEGRAL(r := 0:0.02 ;
                               r*VESSEL_pipe.VELZ_liquid(0.06,r)) = RESERVOIR_R2.VEL;

VESSEL_pipe.RHO = RESERVOIR_R2.RHO;

FOR r := 0 TO 0.02 DO

  PARTIAL(VESSEL_pipe.VELR_liquid(0.06,r), AXIALpipe) = 0;

END
FOR r := 0 TO 0.02 DO

  PARTIAL(VESSEL_pipe.VELZ_liquid(0.06,r), AXIALpipe) = 0;

END

END #model FLOWSHEET
```

CONTAINS DISKETTE

UNABLE TO COPY

CONTACT UNIVERSITY

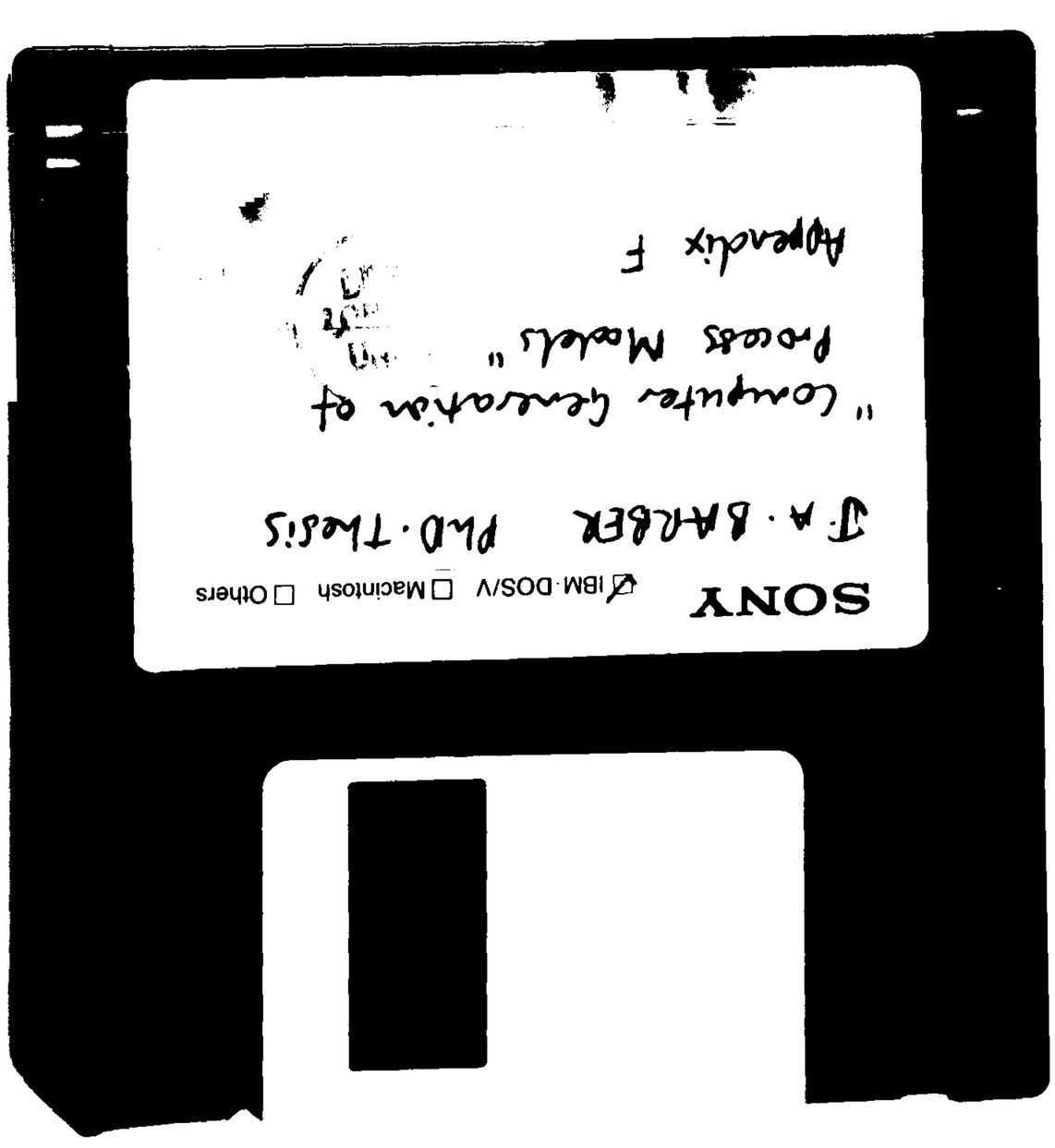
IF YOU WISH TO SEE

THIS MATERIAL

Appendix F

gPROMS Files For Examples of Chapter 6

The attached floppy disk contains the complete *gPROMS* input files for the modelling examples presented in chapter 6. These were produced by manually editing the files given in appendix E.



BRL LONDON UNIV.